

IRIS
0.1.1

Generated by Doxygen 1.5.8

Wed Jun 16 00:55:11 2010

Contents

1 Main Page	1
1.1 General	1
1.2 License	1
1.3 Author	1
2 Directory Hierarchy	3
2.1 Directories	3
3 Class Index	5
3.1 Class Hierarchy	5
4 Class Index	9
4.1 Class List	9
5 File Index	13
5.1 File List	13
6 Directory Documentation	17
6.1 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/components/impl/backup/ Directory Reference	17
6.2 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/components/ Directory Reference	18
6.3 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/data_- types/ Directory Reference	19
6.4 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/frontend/ Directory Reference	20
6.5 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/frontend/impl/ Directory Reference	21
6.6 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/components/impl/ Directory Reference	22
6.7 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/data_- types/impl/ Directory Reference	23

6.8	/home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/components/interfaces/ Directory Reference	24
6.9	/home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/kernel/ Directory Reference	25
6.10	/home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/MemCtrl/ Directory Reference	26
6.11	/home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/ Directory Reference	27
7	Class Documentation	29
7.1	AddressDecoder Class Reference	29
7.1.1	Detailed Description	29
7.1.2	Constructor & Destructor Documentation	30
7.1.2.1	AddressDecoder	30
7.1.3	Member Function Documentation	30
7.1.3.1	get_putput_port	30
7.1.3.2	get_vc	30
7.1.3.3	is_channel_empty	30
7.1.3.4	is_empty	30
7.1.3.5	no_channels	30
7.1.3.6	pull	30
7.1.3.7	push	30
7.1.3.8	set_no_channels	30
7.1.3.9	speculate_channel	30
7.1.3.10	speculate_port	30
7.2	AddrMap Class Reference	31
7.2.1	Detailed Description	31
7.2.2	Constructor & Destructor Documentation	31
7.2.2.1	AddrMap	31
7.2.2.2	~AddrMap	31
7.2.3	Member Function Documentation	31
7.2.3.1	map_addr	31
7.2.3.2	process_event	32
7.2.3.3	toString	32
7.2.4	Member Data Documentation	32
7.2.4.1	parent	32
7.3	Arbiter Class Reference	33
7.3.1	Detailed Description	33

7.3.2	Constructor & Destructor Documentation	33
7.3.2.1	Arbiter	33
7.4	BankHandler Class Reference	34
7.4.1	Detailed Description	35
7.4.2	Constructor & Destructor Documentation	35
7.4.2.1	BankHandler	35
7.4.2.2	~BankHandler	35
7.4.3	Member Function Documentation	35
7.4.3.1	CanUpperStart	35
7.4.3.2	FCFS	35
7.4.3.3	FindHighest	36
7.4.3.4	FindRank	36
7.4.3.5	FRFCFS	36
7.4.3.6	HasWork	36
7.4.3.7	HighestRankedFirst	36
7.4.3.8	IsBufferFull	37
7.4.3.9	MainScheduler	37
7.4.3.10	OldestFirst	37
7.4.3.11	PARBS	38
7.4.3.12	process_event	38
7.4.3.13	ReadsFirst	38
7.4.3.14	RestorePrevState	39
7.4.3.15	RowHitFirst	39
7.4.3.16	ScheduleUnmarked	39
7.4.3.17	SetBypasses	40
7.4.3.18	SetPrevState	40
7.4.3.19	SetReadsOWrite	40
7.4.3.20	StopUpper	40
7.4.3.21	toString	41
7.4.4	Member Data Documentation	41
7.4.4.1	address	41
7.4.4.2	bankId	41
7.4.4.3	bankTag	41
7.4.4.4	bufferFull	41
7.4.4.5	bufferFullCounter	41
7.4.4.6	bufferId	41

7.4.4.7	bufferOccupancy	42
7.4.4.8	bufferSize	42
7.4.4.9	bypassReq	42
7.4.4.10	generated	42
7.4.4.11	myChannel	42
7.4.4.12	myRank	42
7.4.4.13	parent	42
7.4.4.14	prevBufferIndex	42
7.4.4.15	prevBypassReq	43
7.4.4.16	prevRowOpen	43
7.4.4.17	rowBufferIndex	43
7.4.4.18	rowOpen	43
7.4.4.19	stoppedUpper	43
7.4.4.20	stopSignal	43
7.5	BankState Struct Reference	44
7.5.1	Detailed Description	44
7.5.2	Constructor & Destructor Documentation	44
7.5.2.1	BankState	44
7.5.3	Member Data Documentation	44
7.5.3.1	prevCmd	44
7.5.3.2	prevCmdTime	44
7.5.3.3	prevRow	44
7.6	BodyFlit Class Reference	45
7.6.1	Detailed Description	45
7.6.2	Constructor & Destructor Documentation	45
7.6.2.1	BodyFlit	45
7.6.2.2	~BodyFlit	45
7.6.3	Member Function Documentation	45
7.6.3.1	populate_body_flit	45
7.6.3.2	toString	46
7.6.4	Member Data Documentation	46
7.6.4.1	bf_data	46
7.7	Buffer Class Reference	47
7.7.1	Detailed Description	47
7.7.2	Constructor & Destructor Documentation	47
7.7.2.1	Buffer	47

7.7.2.2	~Buffer	47
7.7.3	Member Function Documentation	47
7.7.3.1	get_occupancy	47
7.7.3.2	pull	47
7.7.3.3	push	47
7.8	Bus Class Reference	48
7.8.1	Detailed Description	48
7.8.2	Constructor & Destructor Documentation	48
7.8.2.1	Bus	48
7.8.2.2	~Bus	48
7.8.3	Member Function Documentation	48
7.8.3.1	process_event	48
7.8.3.2	SetLinks	49
7.8.3.3	toString	49
7.8.4	Member Data Documentation	49
7.8.4.1	child1	49
7.8.4.2	child2	49
7.8.4.3	cmdBus	49
7.8.4.4	dataBus	49
7.8.4.5	mc	49
7.8.4.6	parent	50
7.9	BusHandler Class Reference	51
7.9.1	Detailed Description	51
7.9.2	Constructor & Destructor Documentation	51
7.9.2.1	BusHandler	51
7.9.2.2	~BusHandler	51
7.9.3	Member Function Documentation	52
7.9.3.1	BankRankFree	52
7.9.3.2	IsFull	52
7.9.3.3	LowLevelCmdGen	52
7.9.3.4	SetIfFull	52
7.9.3.5	toString	52
7.9.4	Member Data Documentation	52
7.9.4.1	cmdIssuer	52
7.9.4.2	cmdQueue	53
7.9.4.3	full	53

7.9.4.4	linkBusy	53
7.9.4.5	oneReq	53
7.9.4.6	parent	53
7.9.4.7	stopSignal	53
7.10	ChannelHandler Class Reference	54
7.10.1	Detailed Description	54
7.10.2	Constructor & Destructor Documentation	54
7.10.2.1	ChannelHandler	54
7.10.2.2	\sim ChannelHandler	54
7.10.3	Member Data Documentation	54
7.10.3.1	chanId	54
7.10.3.2	rank	54
7.11	ChannelState Struct Reference	55
7.11.1	Detailed Description	55
7.11.2	Member Data Documentation	55
7.11.2.1	cmd	55
7.11.2.2	prevCmdBank	55
7.11.2.3	prevCmdRank	55
7.11.2.4	prevCmdTime	55
7.11.2.5	rank	56
7.12	CmdBusHandler Class Reference	57
7.12.1	Detailed Description	57
7.12.2	Constructor & Destructor Documentation	57
7.12.2.1	CmdBusHandler	57
7.12.2.2	\sim CmdBusHandler	57
7.12.3	Member Function Documentation	57
7.12.3.1	process_event	57
7.12.3.2	toString	58
7.12.4	Member Data Documentation	58
7.12.4.1	child	58
7.12.4.2	parent	58
7.13	CmdIssuer Class Reference	59
7.13.1	Detailed Description	59
7.13.2	Constructor & Destructor Documentation	60
7.13.2.1	CmdIssuer	60
7.13.2.2	\sim CmdIssuer	60

7.13.3 Member Function Documentation	60
7.13.3.1 BankNotBusy	60
7.13.3.2 BanksFirstReq	60
7.13.3.3 BusNotBusy	61
7.13.3.4 CalculateBurstL	61
7.13.3.5 CalculateBusyTime	61
7.13.3.6 CalculateDataDelay	62
7.13.3.7 CanSchedule	62
7.13.3.8 CmdDelay	62
7.13.3.9 HasWork	62
7.13.3.10 IssueCmd	62
7.13.3.11 Max	63
7.13.3.12 process_event	63
7.13.3.13 SetBusBusyTime	63
7.13.3.14 SetPrevState	64
7.13.3.15 toString	64
7.13.4 Member Data Documentation	64
7.13.4.1 bufferId	64
7.13.4.2 busBusy	64
7.13.4.3 cmdBus	64
7.13.4.4 dataBus	65
7.13.4.5 generated_start_event	65
7.13.4.6 Id	65
7.13.4.7 parent	65
7.13.4.8 prevBusCmd	65
7.13.4.9 prevState	65
7.14 Component Class Reference	66
7.14.1 Detailed Description	66
7.14.2 Constructor & Destructor Documentation	66
7.14.2.1 Component	66
7.14.2.2 Component	67
7.14.2.3 ~Component	67
7.14.3 Member Function Documentation	67
7.14.3.1 addInputLink	67
7.14.3.2 addOutputLink	67
7.14.3.3 myId	67

7.14.3.4	setComponentId	67
7.14.4	Member Data Documentation	68
7.14.4.1	componentId	68
7.14.4.2	inLinks	68
7.14.4.3	outLinks	68
7.15	ComponentDescription Class Reference	69
7.15.1	Detailed Description	69
7.15.2	Constructor & Destructor Documentation	69
7.15.2.1	ComponentDescription	69
7.15.3	Member Data Documentation	69
7.15.3.1	lpId	69
7.15.3.2	ptr	69
7.16	Crossbar Class Reference	70
7.16.1	Detailed Description	70
7.16.2	Constructor & Destructor Documentation	70
7.16.2.1	Crossbar	70
7.16.2.2	\sim Crossbar	70
7.16.3	Member Function Documentation	70
7.16.3.1	configure_crossbar	70
7.16.3.2	get_map	70
7.16.3.3	get_no_channels	71
7.16.3.4	get_no_input_ports	71
7.16.3.5	get_no_output_ports	71
7.16.3.6	is_empty	71
7.16.3.7	is_full	71
7.16.3.8	set_input_ports	71
7.16.3.9	set_no_virtual_channels	71
7.16.3.10	set_output_ports	71
7.17	Data Struct Reference	72
7.17.1	Detailed Description	72
7.17.2	Member Data Documentation	72
7.17.2.1	size	72
7.17.2.2	value	72
7.18	DataBusHandler Class Reference	73
7.18.1	Detailed Description	73
7.18.2	Constructor & Destructor Documentation	73

7.18.2.1	DataBusHandler	73
7.18.2.2	~DataBusHandler	73
7.18.3	Member Function Documentation	73
7.18.3.1	process_event	73
7.18.3.2	toString	74
7.18.4	Member Data Documentation	74
7.18.4.1	busBusyTime	74
7.18.4.2	child1	74
7.18.4.3	child2	74
7.18.4.4	parent	74
7.18.4.5	prevTime	74
7.19	DRAM Class Reference	76
7.19.1	Detailed Description	76
7.19.2	Constructor & Destructor Documentation	76
7.19.2.1	DRAM	76
7.19.2.2	~DRAM	76
7.19.3	Member Function Documentation	76
7.19.3.1	process_event	76
7.19.3.2	SetLinks	77
7.19.3.3	toString	77
7.19.4	Member Data Documentation	77
7.19.4.1	dc	77
7.19.4.2	mc	77
7.19.4.3	parent	77
7.20	DRAMChannel Class Reference	78
7.20.1	Detailed Description	78
7.20.2	Constructor & Destructor Documentation	78
7.20.2.1	DRAMChannel	78
7.20.2.2	~DRAMChannel	78
7.20.3	Member Function Documentation	79
7.20.3.1	process_event	79
7.20.3.2	toString	79
7.20.4	Member Data Documentation	79
7.20.4.1	child	79
7.20.4.2	dramBankBusyCycles	79
7.20.4.3	dramBankBusyTime	79

7.20.4.4	dramBusyCycles	79
7.20.4.5	dramBusyTime	80
7.20.4.6	dramReadCycles	80
7.20.4.7	dramWriteCycles	80
7.20.4.8	mc	80
7.20.4.9	parent	80
7.21	DRAMCmdState Struct Reference	81
7.21.1	Detailed Description	81
7.21.2	Constructor & Destructor Documentation	81
7.21.2.1	DRAMCmdState	81
7.21.2.2	\sim DRAMCmdState	81
7.21.3	Member Function Documentation	81
7.21.3.1	set	81
7.21.4	Member Data Documentation	82
7.21.4.1	burst	82
7.21.4.2	cmd	82
7.21.4.3	req	82
7.22	Event0< T, OBJ > Class Template Reference	83
7.22.1	Detailed Description	83
7.22.2	Constructor & Destructor Documentation	83
7.22.2.1	Event0	83
7.22.3	Member Function Documentation	83
7.22.3.1	CallHandler	83
7.22.4	Member Data Documentation	84
7.22.4.1	handler	84
7.22.4.2	obj	84
7.23	Event0Stat Class Reference	85
7.23.1	Detailed Description	85
7.23.2	Constructor & Destructor Documentation	85
7.23.2.1	Event0Stat	85
7.23.3	Member Function Documentation	85
7.23.3.1	CallHandler	85
7.23.4	Member Data Documentation	85
7.23.4.1	handler	85
7.24	Event1< T, OBJ, U1, T1 > Class Template Reference	86
7.24.1	Detailed Description	86

7.24.2	Constructor & Destructor Documentation	86
7.24.2.1	Event1	86
7.24.3	Member Function Documentation	86
7.24.3.1	CallHandler	86
7.24.4	Member Data Documentation	87
7.24.4.1	handler	87
7.24.4.2	obj	87
7.24.4.3	t1	87
7.25	Event1Stat< U1, T1 > Class Template Reference	88
7.25.1	Detailed Description	88
7.25.2	Constructor & Destructor Documentation	88
7.25.2.1	Event1Stat	88
7.25.3	Member Function Documentation	88
7.25.3.1	CallHandler	88
7.25.4	Member Data Documentation	89
7.25.4.1	handler	89
7.25.4.2	t1	89
7.26	Event2< T, OBJ, U1, T1, U2, T2 > Class Template Reference	90
7.26.1	Detailed Description	90
7.26.2	Constructor & Destructor Documentation	90
7.26.2.1	Event2	90
7.26.3	Member Function Documentation	90
7.26.3.1	CallHandler	90
7.26.4	Member Data Documentation	91
7.26.4.1	handler	91
7.26.4.2	obj	91
7.26.4.3	t1	91
7.26.4.4	t2	91
7.27	Event2Stat< U1, T1, U2, T2 > Class Template Reference	92
7.27.1	Detailed Description	92
7.27.2	Constructor & Destructor Documentation	92
7.27.2.1	Event2Stat	92
7.27.3	Member Function Documentation	92
7.27.3.1	CallHandler	92
7.27.4	Member Data Documentation	93
7.27.4.1	handler	93

7.27.4.2	t1	93
7.27.4.3	t2	93
7.28	Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >	Class Template Reference	94
7.28.1	Detailed Description	94
7.28.2	Constructor & Destructor Documentation	94
7.28.2.1	Event3	94
7.28.3	Member Function Documentation	95
7.28.3.1	CallHandler	95
7.28.4	Member Data Documentation	95
7.28.4.1	handler	95
7.28.4.2	obj	95
7.28.4.3	t1	95
7.28.4.4	t2	95
7.28.4.5	t3	95
7.29	Event3Stat< U1, T1, U2, T2, U3, T3 >	Class Template Reference	97
7.29.1	Detailed Description	97
7.29.2	Constructor & Destructor Documentation	97
7.29.2.1	Event3Stat	97
7.29.3	Member Function Documentation	97
7.29.3.1	CallHandler	97
7.29.4	Member Data Documentation	98
7.29.4.1	handler	98
7.29.4.2	t1	98
7.29.4.3	t2	98
7.29.4.4	t3	98
7.30	Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >	Class Template Reference	99
7.30.1	Detailed Description	99
7.30.2	Constructor & Destructor Documentation	99
7.30.2.1	Event4	99
7.30.3	Member Function Documentation	100
7.30.3.1	CallHandler	100
7.30.4	Member Data Documentation	100
7.30.4.1	handler	100
7.30.4.2	obj	100
7.30.4.3	t1	100
7.30.4.4	t2	100

7.30.4.5	t3	101
7.30.4.6	t4	101
7.31	Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 > Class Template Reference	102
7.31.1	Detailed Description	102
7.31.2	Constructor & Destructor Documentation	102
7.31.2.1	Event4Stat	102
7.31.3	Member Function Documentation	103
7.31.3.1	CallHandler	103
7.31.4	Member Data Documentation	103
7.31.4.1	handler	103
7.31.4.2	t1	103
7.31.4.3	t2	103
7.31.4.4	t3	103
7.31.4.5	t4	103
7.32	event_less Class Reference	105
7.32.1	Detailed Description	105
7.32.2	Constructor & Destructor Documentation	105
7.32.2.1	event_less	105
7.32.3	Member Function Documentation	105
7.32.3.1	operator()	105
7.33	EventBase Class Reference	106
7.33.1	Detailed Description	106
7.33.2	Constructor & Destructor Documentation	106
7.33.2.1	EventBase	106
7.33.2.2	EventBase	106
7.33.3	Member Function Documentation	106
7.33.3.1	CallHandler	106
7.33.4	Member Data Documentation	107
7.33.4.1	nextUID	107
7.33.4.2	time	107
7.33.4.3	uid	107
7.34	EventId Class Reference	108
7.34.1	Detailed Description	108
7.34.2	Constructor & Destructor Documentation	108
7.34.2.1	EventId	108
7.34.3	Member Function Documentation	108

7.34.3.1	CallHandler	108
7.35	Flit Class Reference	109
7.35.1	Detailed Description	109
7.35.2	Constructor & Destructor Documentation	109
7.35.2.1	Flit	109
7.35.2.2	~Flit	109
7.35.3	Member Function Documentation	109
7.35.3.1	populate_phit_data	109
7.35.3.2	toString	110
7.35.4	Member Data Documentation	110
7.35.4.1	phits	110
7.35.4.2	type	110
7.35.4.3	vc	110
7.36	GenericArbiter Class Reference	111
7.36.1	Detailed Description	111
7.36.2	Constructor & Destructor Documentation	111
7.36.2.1	GenericArbiter	111
7.36.2.2	~GenericArbiter	111
7.36.3	Member Function Documentation	112
7.36.3.1	clear_winner	112
7.36.3.2	get_no_requestors	112
7.36.3.3	is_empty	112
7.36.3.4	is_requested	112
7.36.3.5	pick_winner	112
7.36.3.6	request	113
7.36.3.7	set_no_requestors	113
7.36.3.8	toString	113
7.36.4	Member Data Documentation	113
7.36.4.1	address	113
7.36.4.2	done	113
7.36.4.3	last_winner	113
7.36.4.4	name	114
7.36.4.5	node_ip	114
7.36.4.6	requests	114
7.36.4.7	vcs	114
7.37	GenericBuffer Class Reference	115

7.37.1	Detailed Description	116
7.37.2	Constructor & Destructor Documentation	116
7.37.2.1	GenericBuffer	116
7.37.2.2	\sim GenericBuffer	116
7.37.3	Member Function Documentation	116
7.37.3.1	change_pull_channel	116
7.37.3.2	change_push_channel	116
7.37.3.3	get_no_credits	116
7.37.3.4	get_no_vcs	116
7.37.3.5	get_occupancy	117
7.37.3.6	get_pull_channel	117
7.37.3.7	get_push_channel	117
7.37.3.8	got_credit	117
7.37.3.9	is_channel_full	117
7.37.3.10	is_empty	117
7.37.3.11	peek	117
7.37.3.12	pull	118
7.37.3.13	push	118
7.37.3.14	resize	118
7.37.3.15	set_no_credits	118
7.37.3.16	toString	118
7.37.4	Member Data Documentation	119
7.37.4.1	buffer_size	119
7.37.4.2	buffers	119
7.37.4.3	credits	119
7.37.4.4	max_credits	119
7.37.4.5	next_port	119
7.37.4.6	pull_channel	119
7.37.4.7	push_channel	119
7.37.4.8	vcs	119
7.38	GenericCrossbar Class Reference	120
7.38.1	Detailed Description	120
7.38.2	Constructor & Destructor Documentation	121
7.38.2.1	GenericCrossbar	121
7.38.2.2	\sim GenericCrossbar	121
7.38.3	Member Function Documentation	121

7.38.3.1	clear	121
7.38.3.2	configure_crossbar	121
7.38.3.3	get_map	121
7.38.3.4	get_no_channels	121
7.38.3.5	get_no_input_ports	121
7.38.3.6	get_no_output_ports	122
7.38.3.7	is_empty	122
7.38.3.8	is_full	122
7.38.3.9	pull	122
7.38.3.10	push	122
7.38.3.11	set_input_ports	123
7.38.3.12	set_no_virtual_channels	123
7.38.3.13	set_output_ports	123
7.38.3.14	toString	123
7.38.4	Member Data Documentation	124
7.38.4.1	busy	124
7.38.4.2	input_ports	124
7.38.4.3	map	124
7.38.4.4	output_ports	124
7.39	GenericCrossbar::GenericCrossbar::CrossbarUnit Class Reference	125
7.39.1	Detailed Description	125
7.39.2	Member Data Documentation	125
7.39.2.1	data	125
7.39.2.2	valid	125
7.40	GenericFlatMc Class Reference	126
7.40.1	Detailed Description	127
7.40.2	Constructor & Destructor Documentation	127
7.40.2.1	GenericFlatMc	127
7.40.2.2	~GenericFlatMc	127
7.40.3	Member Function Documentation	127
7.40.3.1	convertFromBitStream	127
7.40.3.2	finish	127
7.40.3.3	handle_new_packet_event	127
7.40.3.4	handle_out_pull_event	128
7.40.3.5	handle_ready_event	128
7.40.3.6	print_stats	129

7.40.3.7	process_event	129
7.40.3.8	set_output_path	129
7.40.3.9	setup	129
7.40.3.10	toString	129
7.40.4	Member Data Documentation	130
7.40.4.1	max_sim_time	130
7.40.4.2	max_time	130
7.40.4.3	mc_node_ip	130
7.40.4.4	min_pkt_latency	130
7.40.4.5	no_nodes	130
7.40.4.6	out_file	130
7.40.4.7	out_filename	130
7.40.4.8	out_packets	130
7.40.4.9	outstanding_hlp	130
7.40.4.10	packets	131
7.40.4.11	packets_pending	131
7.40.4.12	pending_packets	131
7.40.4.13	pending_packets_time	131
7.40.4.14	ready	131
7.40.4.15	sending	131
7.40.4.16	vcs	131
7.40.4.17	waiting_at_injection	131
7.41	GenericInterface Class Reference	133
7.41.1	Detailed Description	134
7.41.2	Constructor & Destructor Documentation	134
7.41.2.1	GenericInterface	134
7.41.2.2	~GenericInterface	134
7.41.3	Member Function Documentation	134
7.41.3.1	get_flits_out	134
7.41.3.2	get_no_credits	134
7.41.3.3	get_packets	134
7.41.3.4	get_packets_out	134
7.41.3.5	handle_link_arrival	135
7.41.3.6	handle_new_packet_event	135
7.41.3.7	handle_ready_event	135
7.41.3.8	handle_tick_event	136

7.41.3.9	print_stats	136
7.41.3.10	process_event	136
7.41.3.11	set_buffer_size	137
7.41.3.12	set_no_credits	137
7.41.3.13	set_no_vcs	137
7.41.3.14	setup	137
7.41.3.15	toString	138
7.41.4	Member Data Documentation	138
7.41.4.1	buffer_size	138
7.41.4.2	credits	138
7.41.4.3	downstream_credits	138
7.41.4.4	flits_in	138
7.41.4.5	flits_out	138
7.41.4.6	in_buffer	138
7.41.4.7	in_packet_cleared	139
7.41.4.8	in_packet_complete	139
7.41.4.9	in_packets	139
7.41.4.10	in_packets_flit_index	139
7.41.4.11	in_packets_valid	139
7.41.4.12	in_ready	139
7.41.4.13	out_buffer	139
7.41.4.14	out_packet_flit_index	139
7.41.4.15	out_packets	139
7.41.4.16	packets_in	140
7.41.4.17	packets_out	140
7.41.4.18	ticking	140
7.41.4.19	total_packets_in_time	140
7.41.4.20	vcs	140
7.42	GenericLink Class Reference	141
7.42.1	Detailed Description	141
7.42.2	Constructor & Destructor Documentation	141
7.42.2.1	GenericLink	141
7.42.2.2	~GenericLink	142
7.42.3	Member Function Documentation	142
7.42.3.1	get_flits_utilization	142
7.42.3.2	handle_link_arrival_event	142

7.42.3.3	print_stats	142
7.42.3.4	process_event	142
7.42.3.5	setup	143
7.42.3.6	toString	143
7.42.4	Member Data Documentation	143
7.42.4.1	credits_passed	143
7.42.4.2	cycles	143
7.42.4.3	flits_passed	143
7.42.4.4	node_ip	143
7.42.4.5	stages	144
7.43	GenericPortArbiter Class Reference	145
7.43.1	Detailed Description	145
7.43.2	Constructor & Destructor Documentation	146
7.43.2.1	GenericPortArbiter	146
7.43.2.2	~GenericPortArbiter	146
7.43.3	Member Function Documentation	146
7.43.3.1	clear	146
7.43.3.2	clear_winner	146
7.43.3.3	destroy	146
7.43.3.4	flush_all_requests	146
7.43.3.5	get_no_requests	146
7.43.3.6	get_requests	146
7.43.3.7	is_empty	146
7.43.3.8	is_requested	146
7.43.3.9	pick_winner	146
7.43.3.10	pull_winner	147
7.43.3.11	request	147
7.43.3.12	resize	147
7.43.3.13	set_req_queue_size	147
7.43.3.14	toString	147
7.43.4	Member Data Documentation	147
7.43.4.1	done	147
7.43.4.2	flits	147
7.43.4.3	last_winner	147
7.43.4.4	locked	147
7.43.4.5	ports	148

7.43.4.6 requests	148
7.43.4.7 write_time	148
7.44 GenericRC Class Reference	149
7.44.1 Detailed Description	150
7.44.2 Constructor & Destructor Documentation	150
7.44.2.1 GenericRC	150
7.44.2.2 ~GenericRC	150
7.44.3 Member Function Documentation	150
7.44.3.1 get_no_channels	150
7.44.3.2 get_output_port	150
7.44.3.3 get_virtual_channel	150
7.44.3.4 is_empty	150
7.44.3.5 push	150
7.44.3.6 resize	151
7.44.3.7 route_negative_first	151
7.44.3.8 route_north_last	151
7.44.3.9 route_north_last_non_minimal	151
7.44.3.10 route_valiant	152
7.44.3.11 route_west_first	152
7.44.3.12 route_x_y	152
7.44.3.13 speculate_channel	152
7.44.3.14 speculate_port	152
7.44.3.15 toString	152
7.44.4 Member Data Documentation	152
7.44.4.1 address	152
7.44.4.2 addresses	153
7.44.4.3 grid_xloc	153
7.44.4.4 grid_yloc	153
7.44.4.5 name	153
7.44.4.6 node_ip	153
7.44.4.7 possible_out_ports	153
7.45 GenericRC::GenericRC::Address Class Reference	154
7.45.1 Detailed Description	154
7.45.2 Member Data Documentation	154
7.45.2.1 channel	154
7.45.2.2 last_adaptive_port	154

7.45.2.3	out_port	154
7.45.2.4	possible_out_ports	154
7.45.2.5	route_valid	154
7.46	GenericRouterAdaptive Class Reference	155
7.46.1	Detailed Description	156
7.46.2	Constructor & Destructor Documentation	156
7.46.2.1	GenericRouterAdaptive	156
7.46.2.2	~GenericRouterAdaptive	156
7.46.3	Member Function Documentation	156
7.46.3.1	do_switch_allocation	156
7.46.3.2	do_switch_traversal	157
7.46.3.3	handle_link_arrival_event	157
7.46.3.4	handle_tick_event	157
7.46.3.5	init	158
7.46.3.6	print_stats	158
7.46.3.7	process_event	158
7.46.3.8	send_credit_back	159
7.46.3.9	set_edge_links	159
7.46.3.10	set_grid_x_location	159
7.46.3.11	set_grid_y_location	159
7.46.3.12	set_no_nodes	159
7.46.3.13	toString	160
7.46.4	Member Data Documentation	160
7.46.4.1	available_ports	160
7.46.4.2	decoders	160
7.46.4.3	downstream_credits	160
7.46.4.4	flits	160
7.46.4.5	in_buffers	160
7.46.4.6	input_buffer_state	161
7.46.4.7	last_flit_out_cycle	161
7.46.4.8	packets	161
7.46.4.9	stat_flit_out	161
7.46.4.10	stat_packet_out	161
7.46.4.11	stat_sim_total_time	161
7.46.4.12	swa	161
7.46.4.13	ticking	161

7.46.4.14 total_packet_latency	162
7.46.4.15 xbar	162
7.47 GenericRouterNoVcs Class Reference	163
7.47.1 Detailed Description	164
7.47.2 Constructor & Destructor Documentation	164
7.47.2.1 GenericRouterNoVcs	164
7.47.2.2 ~GenericRouterNoVcs	164
7.47.2.3 GenericRouterNoVcs	164
7.47.2.4 ~GenericRouterNoVcs	164
7.47.3 Member Function Documentation	164
7.47.3.1 do_switch_allocation	164
7.47.3.2 do_switch_traversal	165
7.47.3.3 handle_link_arrival_event	165
7.47.3.4 handle_link_arrival_event	165
7.47.3.5 handle_tick_event	165
7.47.3.6 handle_tick_event	165
7.47.3.7 init	166
7.47.3.8 init	166
7.47.3.9 print_stats	166
7.47.3.10 print_stats	167
7.47.3.11 process_event	167
7.47.3.12 process_event	167
7.47.3.13 send_credit_back	167
7.47.3.14 send_credit_back	167
7.47.3.15 set_grid_x_location	167
7.47.3.16 set_grid_x_location	167
7.47.3.17 set_grid_y_location	168
7.47.3.18 set_grid_y_location	168
7.47.3.19 set_no_nodes	168
7.47.3.20 set_no_nodes	168
7.47.3.21 toString	168
7.47.3.22 toString	168
7.47.4 Member Data Documentation	168
7.47.4.1 decoders	168
7.47.4.2 downstream_credits	169
7.47.4.3 flits	169

7.47.4.4	in_buffers	169
7.47.4.5	input_buffer_state	169
7.47.4.6	last_flit_out_cycle	169
7.47.4.7	mstate	169
7.47.4.8	out_arbiters	169
7.47.4.9	packets	169
7.47.4.10	swa	169
7.47.4.11	swa	170
7.47.4.12	ticking	170
7.47.4.13	total_packet_latency	170
7.47.4.14	xbar	170
7.48	GenericRouterVct Class Reference	171
7.48.1	Detailed Description	172
7.48.2	Constructor & Destructor Documentation	172
7.48.2.1	GenericRouterVct	172
7.48.2.2	\sim GenericRouterVct	172
7.48.3	Member Function Documentation	172
7.48.3.1	do_switch_allocation	172
7.48.3.2	do_switch_traversal	172
7.48.3.3	handle_link_arrival_event	173
7.48.3.4	handle_tick_event	173
7.48.3.5	init	174
7.48.3.6	print_stats	174
7.48.3.7	process_event	174
7.48.3.8	send_credit_back	175
7.48.3.9	set_grid_x_location	175
7.48.3.10	set_grid_y_location	175
7.48.3.11	set_no_nodes	175
7.48.3.12	toString	175
7.48.4	Member Data Documentation	176
7.48.4.1	decoders	176
7.48.4.2	downstream_credits	176
7.48.4.3	flits	176
7.48.4.4	in_buffers	176
7.48.4.5	input_buffer_state	176
7.48.4.6	last_flit_out_cycle	176

7.48.4.7	packets	176
7.48.4.8	stat_flit_out	177
7.48.4.9	stat_packet_out	177
7.48.4.10	stat_sim_total_time	177
7.48.4.11	swa	177
7.48.4.12	ticking	177
7.48.4.13	total_packet_latency	177
7.48.4.14	xbar	177
7.49	GenericRPG Class Reference	178
7.49.1	Detailed Description	179
7.49.2	Constructor & Destructor Documentation	179
7.49.2.1	GenericRPG	179
7.49.2.2	~GenericRPG	180
7.49.2.3	GenericRPG	180
7.49.2.4	~GenericRPG	180
7.49.3	Member Function Documentation	180
7.49.3.1	compare	180
7.49.3.2	compare	180
7.49.3.3	finish	180
7.49.3.4	finish	180
7.49.3.5	get_all_recv	180
7.49.3.6	get_all_recv	180
7.49.3.7	get_all_sent	180
7.49.3.8	get_all_sent	180
7.49.3.9	handle_new_packet_event	180
7.49.3.10	handle_new_packet_event	180
7.49.3.11	handle_out_pull_event	181
7.49.3.12	handle_out_pull_event	181
7.49.3.13	handle_ready_event	181
7.49.3.14	handle_ready_event	181
7.49.3.15	idle	181
7.49.3.16	idle	181
7.49.3.17	init	181
7.49.3.18	init	182
7.49.3.19	init_generator	182
7.49.3.20	init_generator	182

7.49.3.21 post_tick	182
7.49.3.22 post_tick	182
7.49.3.23 pre_tick	182
7.49.3.24 pre_tick	182
7.49.3.25 process_event	182
7.49.3.26 process_event	182
7.49.3.27 set_no_vcs	182
7.49.3.28 set_no_vcs	182
7.49.3.29 setup	183
7.49.3.30 setup	183
7.49.3.31 toString	183
7.49.3.32 toString	183
7.49.4 Member Data Documentation	183
7.49.4.1 address	183
7.49.4.2 delay_type	183
7.49.4.3 destination_type	184
7.49.4.4 generator	184
7.49.4.5 hot_spots	184
7.49.4.6 lamda	184
7.49.4.7 last_vc	184
7.49.4.8 length_type	184
7.49.4.9 max_address	184
7.49.4.10 max_delay	184
7.49.4.11 max_length	184
7.49.4.12 max_sim_time	185
7.49.4.13 max_time	185
7.49.4.14 min_delay	185
7.49.4.15 min_length	185
7.49.4.16 only_sink	185
7.49.4.17 out_file	185
7.49.4.18 out_filename	185
7.49.4.19 out_packets	185
7.49.4.20 packets	185
7.49.4.21 ready	185
7.49.4.22 seed	186
7.49.4.23 sending	186

7.49.4.24 sent_packets	186
7.49.4.25 vcs	186
7.50 GenericSink Class Reference	187
7.50.1 Detailed Description	187
7.50.2 Constructor & Destructor Documentation	188
7.50.2.1 GenericSink	188
7.50.3 Member Function Documentation	188
7.50.3.1 handle_new_packet_event	188
7.50.3.2 handle_outpull_event	188
7.50.3.3 handle_ready_event	188
7.50.3.4 process_event	189
7.50.3.5 setup	189
7.50.3.6 toString	189
7.50.4 Member Data Documentation	189
7.50.4.1 address	189
7.50.4.2 last_vc	190
7.50.4.3 max_sim_time	190
7.50.4.4 no_nodes	190
7.50.4.5 out_file	190
7.50.4.6 out_filename	190
7.50.4.7 out_packets	190
7.50.4.8 packets	190
7.50.4.9 ready	190
7.50.4.10 sending	191
7.50.4.11 vcs	191
7.51 GenericTPG Class Reference	192
7.51.1 Detailed Description	193
7.51.2 Constructor & Destructor Documentation	193
7.51.2.1 GenericTPG	193
7.51.2.2 ~GenericTPG	193
7.51.3 Member Function Documentation	194
7.51.3.1 compare	194
7.51.3.2 convertFromBitStream	194
7.51.3.3 convertToBitStream	194
7.51.3.4 finish	194
7.51.3.5 GetNewRequest	194

7.51.3.6	GetNextRequest	195
7.51.3.7	GetRequest	195
7.51.3.8	handle_new_packet_event	195
7.51.3.9	handle_out_pull_event	196
7.51.3.10	handle_ready_event	196
7.51.3.11	print_stats	197
7.51.3.12	process_event	197
7.51.3.13	set_no_vcs	197
7.51.3.14	set_output_path	197
7.51.3.15	set_trace_filename	197
7.51.3.16	setup	198
7.51.3.17	toString	198
7.51.4	Member Data Documentation	198
7.51.4.1	fwd_path_delay	198
7.51.4.2	last_packet_out_cycle	198
7.51.4.3	last_vc	198
7.51.4.4	max_sim_time	198
7.51.4.5	max_time	199
7.51.4.6	mc_node_ip	199
7.51.4.7	min_pkt_latency	199
7.51.4.8	mshrHandler	199
7.51.4.9	no_nodes	199
7.51.4.10	no_outstanding	199
7.51.4.11	out_file	199
7.51.4.12	out_filename	199
7.51.4.13	out_packets	199
7.51.4.14	packets	199
7.51.4.15	packets_in	200
7.51.4.16	ready	200
7.51.4.17	roundTripLat	200
7.51.4.18	sending	200
7.51.4.19	sent_packets	200
7.51.4.20	stat_round_trip_hop_count	200
7.51.4.21	stat_round_trip_memory_latency	200
7.51.4.22	stat_round_trip_network_latency	200
7.51.4.23	stat_waiting_in_ni	200

7.51.4.24 TotalBLP	201
7.51.4.25 trace_filename	201
7.51.4.26 trace_name	201
7.51.4.27 vcs	201
7.52 GenericVcArbiter Class Reference	202
7.52.1 Detailed Description	202
7.52.2 Constructor & Destructor Documentation	203
7.52.2.1 GenericVcArbiter	203
7.52.2.2 ~GenericVcArbiter	203
7.52.3 Member Function Documentation	203
7.52.3.1 clear_winner	203
7.52.3.2 empty	203
7.52.3.3 empty	203
7.52.3.4 get_no_requests	203
7.52.3.5 is_requested	203
7.52.3.6 pick_winner	203
7.52.3.7 pick_winner	204
7.52.3.8 pull_winner	204
7.52.3.9 request	204
7.52.3.10 set_req_queue_size	204
7.52.3.11 toString	204
7.52.4 Member Data Documentation	204
7.52.4.1 address	204
7.52.4.2 arb	204
7.52.4.3 done	205
7.52.4.4 flits	205
7.52.4.5 last_winner	205
7.52.4.6 name	205
7.52.4.7 next_port	205
7.52.4.8 node_ip	205
7.52.4.9 requests	205
7.52.4.10 write_time	205
7.53 GenericVcsInterface Class Reference	206
7.53.1 Detailed Description	207
7.53.2 Constructor & Destructor Documentation	207
7.53.2.1 GenericVcsInterface	207

7.53.2.2 ~GenericVcsInterface	207
7.53.3 Member Function Documentation	207
7.53.3.1 get_flits_out	207
7.53.3.2 get_no_credits	207
7.53.3.3 get_packets	207
7.53.3.4 get_packets_out	208
7.53.3.5 handle_link_arrival	208
7.53.3.6 handle_new_packet_event	208
7.53.3.7 handle_ready_event	209
7.53.3.8 handle_tick_event	209
7.53.3.9 print_stats	209
7.53.3.10 process_event	210
7.53.3.11 set_buffer_size	210
7.53.3.12 set_no_credits	210
7.53.3.13 set_no_vcs	210
7.53.3.14 setup	211
7.53.3.15 toString	211
7.53.4 Member Data Documentation	211
7.53.4.1 buffer_size	211
7.53.4.2 credits	211
7.53.4.3 downstream_credits	211
7.53.4.4 flits_in	211
7.53.4.5 flits_out	212
7.53.4.6 in_buffer	212
7.53.4.7 in_packet_cleared	212
7.53.4.8 in_packet_complete	212
7.53.4.9 in_packets	212
7.53.4.10 in_packets_flit_index	212
7.53.4.11 in_packets_valid	212
7.53.4.12 in_ready	212
7.53.4.13 out_arbiter	212
7.53.4.14 out_buffer	213
7.53.4.15 out_packet_flit_index	213
7.53.4.16 out_packets	213
7.53.4.17 packets_in	213
7.53.4.18 packets_out	213

7.53.4.19	ticking	213
7.53.4.20	total_packets_in_time	213
7.53.4.21	vcs	213
7.54	HeadFlit Class Reference	215
7.54.1	Detailed Description	215
7.54.2	Constructor & Destructor Documentation	215
7.54.2.1	HeadFlit	215
7.54.2.2	~HeadFlit	216
7.54.3	Member Function Documentation	216
7.54.3.1	next	216
7.54.3.2	populate_head_flit	216
7.54.3.3	route	216
7.54.3.4	toString	216
7.54.4	Member Data Documentation	216
7.54.4.1	addr	216
7.54.4.2	avg_network_latency	216
7.54.4.3	control_bits	217
7.54.4.4	dst_address	217
7.54.4.5	hop_count	217
7.54.4.6	import	217
7.54.4.7	length	217
7.54.4.8	msg_class	217
7.54.4.9	packet_originated_time	217
7.54.4.10	payload	218
7.54.4.11	req_start_time	218
7.54.4.12	src_address	218
7.54.4.13	stat_memory_serviced_time	218
7.54.4.14	transaction_id	218
7.54.4.15	waiting_in_ni	218
7.55	HighLevelPacket Class Reference	219
7.55.1	Detailed Description	219
7.55.2	Constructor & Destructor Documentation	219
7.55.2.1	HighLevelPacket	219
7.55.2.2	~HighLevelPacket	220
7.55.3	Member Function Documentation	220
7.55.3.1	from_low_level_packet	220

7.55.3.2	get_transit_time	220
7.55.3.3	operator==	220
7.55.3.4	to_low_level_packet	220
7.55.3.5	toString	221
7.55.4	Member Data Documentation	221
7.55.4.1	addr	221
7.55.4.2	avg_network_latency	221
7.55.4.3	data	222
7.55.4.4	data_payload_length	222
7.55.4.5	destination	222
7.55.4.6	hop_count	222
7.55.4.7	msg_class	222
7.55.4.8	recv_time	222
7.55.4.9	req_start_time	223
7.55.4.10	sent_time	223
7.55.4.11	source	223
7.55.4.12	stat_memory_serviced_time	223
7.55.4.13	transaction_id	223
7.55.4.14	virtual_channel	223
7.55.4.15	vn	224
7.55.4.16	waiting_in_ni	224
7.56	InputBuffer Class Reference	225
7.56.1	Detailed Description	225
7.56.2	Constructor & Destructor Documentation	225
7.56.2.1	InputBuffer	225
7.56.2.2	~InputBuffer	225
7.56.3	Member Function Documentation	226
7.56.3.1	change_pull_channel	226
7.56.3.2	change_push_channel	226
7.56.3.3	get_no_vc	226
7.56.3.4	get_pull_channel	226
7.56.3.5	get_push_channel	226
7.56.3.6	is_channel_full	226
7.56.3.7	is_empty	226
7.56.3.8	set_no_vc	226
7.57	InputBufferState Class Reference	227

7.57.1	Detailed Description	227
7.57.2	Constructor & Destructor Documentation	227
7.57.2.1	InputBufferState	227
7.57.2.2	~InputBufferState	227
7.57.3	Member Function Documentation	228
7.57.3.1	toString	228
7.57.4	Member Data Documentation	228
7.57.4.1	arrival_time	228
7.57.4.2	clear_message	228
7.57.4.3	credits_sent	228
7.57.4.4	flits_in_ib	228
7.57.4.5	input_channel	228
7.57.4.6	input_port	228
7.57.4.7	length	228
7.57.4.8	msg_class	228
7.57.4.9	output_channel	229
7.57.4.10	output_port	229
7.57.4.11	pipe_stage	229
7.57.4.12	possible_oports	229
7.57.4.13	possible_ovcs	229
7.57.4.14	stat_pkt_intime	229
7.58	Interface Class Reference	230
7.58.1	Detailed Description	230
7.58.2	Constructor & Destructor Documentation	230
7.58.2.1	Interface	230
7.58.2.2	~Interface	230
7.58.3	Member Function Documentation	231
7.58.3.1	get_flits_out	231
7.58.3.2	get_packets	231
7.58.3.3	get_packets_out	231
7.58.3.4	print_stats	231
7.58.3.5	process_event	231
7.58.3.6	set_buffer_size	231
7.58.3.7	set_no_credits	231
7.58.3.8	set_no_vcs	231
7.58.3.9	setup	231

7.58.3.10	toString	231
7.58.4	Member Data Documentation	232
7.58.4.1	input_connection	232
7.58.4.2	output_connection	232
7.58.4.3	processor_connection	232
7.59	IrisEvent Class Reference	233
7.59.1	Detailed Description	233
7.59.2	Constructor & Destructor Documentation	233
7.59.2.1	IrisEvent	233
7.59.2.2	~IrisEvent	233
7.59.3	Member Function Documentation	233
7.59.3.1	toString	233
7.59.4	Member Data Documentation	234
7.59.4.1	dst	234
7.59.4.2	dst_id	234
7.59.4.3	event_data	234
7.59.4.4	src	234
7.59.4.5	src_id	234
7.59.4.6	time	235
7.59.4.7	type	235
7.59.4.8	vc	235
7.60	IrisLink Class Reference	236
7.60.1	Detailed Description	236
7.60.2	Constructor & Destructor Documentation	236
7.60.2.1	IrisLink	236
7.60.2.2	~IrisLink	236
7.60.3	Member Function Documentation	236
7.60.3.1	process_event	236
7.60.3.2	toString	237
7.60.4	Member Data Documentation	237
7.60.4.1	input_connection	237
7.60.4.2	output_connection	237
7.61	Link Class Reference	238
7.61.1	Detailed Description	238
7.61.2	Constructor & Destructor Documentation	238
7.61.2.1	Link	238

7.61.3	Member Function Documentation	238
7.61.3.1	Send	238
7.61.4	Member Data Documentation	239
7.61.4.1	outputs	239
7.61.4.2	src	239
7.61.4.3	width	239
7.62	LinkArrivalData Class Reference	240
7.62.1	Detailed Description	240
7.62.2	Constructor & Destructor Documentation	240
7.62.2.1	LinkArrivalData	240
7.62.2.2	~LinkArrivalData	240
7.62.3	Member Data Documentation	240
7.62.3.1	ptr	240
7.62.3.2	type	241
7.62.3.3	valid	241
7.62.3.4	vc	241
7.63	LowLevelPacket Class Reference	242
7.63.1	Detailed Description	242
7.63.2	Constructor & Destructor Documentation	242
7.63.2.1	LowLevelPacket	242
7.63.2.2	~LowLevelPacket	243
7.63.3	Member Function Documentation	243
7.63.3.1	add	243
7.63.3.2	at	243
7.63.3.3	clear	243
7.63.3.4	get_next_flit	243
7.63.3.5	operator=	243
7.63.3.6	size	243
7.63.3.7	toString	244
7.63.3.8	valid_packet	244
7.63.4	Member Data Documentation	244
7.63.4.1	addr	244
7.63.4.2	avg_network_latency	244
7.63.4.3	control_bits	244
7.63.4.4	destination	244
7.63.4.5	flits	244

7.63.4.6 hop_count	244
7.63.4.7 length	245
7.63.4.8 msg_class	245
7.63.4.9 payload	245
7.63.4.10 req_start_time	245
7.63.4.11 sent_time	245
7.63.4.12 source	245
7.63.4.13 stat_memory_serviced_time	245
7.63.4.14 transaction_id	246
7.63.4.15 virtual_channel	246
7.63.4.16 waiting_in_ni	246
7.64 MC Class Reference	247
7.64.1 Detailed Description	247
7.64.2 Constructor & Destructor Documentation	247
7.64.2.1 ~MC	247
7.64.3 Member Function Documentation	247
7.64.3.1 Init	247
7.64.3.2 StartRefresh	248
7.64.4 Member Data Documentation	248
7.64.4.1 bus	248
7.64.4.2 doneOnce	248
7.64.4.3 dram	248
7.64.4.4 id	248
7.64.4.5 ni	248
7.64.4.6 parent	248
7.64.4.7 refMgr	249
7.64.4.8 reqH	249
7.64.4.9 responseH	249
7.64.4.10 stats	249
7.65 MCFrontEnd Class Reference	250
7.65.1 Detailed Description	251
7.65.2 Constructor & Destructor Documentation	251
7.65.2.1 MCFrontEnd	251
7.65.2.2 ~MCFrontEnd	251
7.65.3 Member Function Documentation	251
7.65.3.1 check_input_conditions	251

7.65.3.2	check_tick	252
7.65.3.3	convertFromBitStream	252
7.65.3.4	get_no_credits	252
7.65.3.5	handle_flit_out_event	252
7.65.3.6	handle_in_arbitrate_event	253
7.65.3.7	handle_in_push_event	253
7.65.3.8	handle_link_arrival	253
7.65.3.9	handle_new_packet_event	254
7.65.3.10	handle_out_arbitrate_event	254
7.65.3.11	handle_ready_event	254
7.65.3.12	handle_tick_event	255
7.65.3.13	print_stats	255
7.65.3.14	process_event	255
7.65.3.15	set_no_credits	255
7.65.3.16	setup	255
7.65.3.17	toString	256
7.65.4	Member Data Documentation	256
7.65.4.1	flits_in	256
7.65.4.2	flits_out	256
7.65.4.3	in_arbiter	256
7.65.4.4	in_buffer	256
7.65.4.5	in_packets	257
7.65.4.6	in_ready	257
7.65.4.7	last_flit_out_cycle	257
7.65.4.8	last_in_arbitrate_cycle	257
7.65.4.9	last_out_arbitrate_cycle	257
7.65.4.10	node_ip	257
7.65.4.11	out_arbiter	257
7.65.4.12	out_buffer	257
7.65.4.13	out_packet_index	258
7.65.4.14	out_packets	258
7.65.4.15	packets_in	258
7.65.4.16	packets_out	258
7.65.4.17	ticking	258
7.65.4.18	total_packets_in_time	258
7.66	Mesh Class Reference	259

7.66.1	Detailed Description	259
7.66.2	Constructor & Destructor Documentation	259
7.66.2.1	Mesh	259
7.66.2.2	~Mesh	260
7.66.3	Member Function Documentation	260
7.66.3.1	connect_interface_processor	260
7.66.3.2	connect_interface_routers	260
7.66.3.3	connect_routers	260
7.66.3.4	init	260
7.66.3.5	print_stats	261
7.66.3.6	set_max_phy_link_bits	261
7.66.3.7	setup	261
7.66.4	Member Data Documentation	261
7.66.4.1	buffer_size	261
7.66.4.2	credits	261
7.66.4.3	grid_size	261
7.66.4.4	interfaces	261
7.66.4.5	link_a	262
7.66.4.6	link_b	262
7.66.4.7	links	262
7.66.4.8	max_sim_time	262
7.66.4.9	no_nodes	262
7.66.4.10	ports	262
7.66.4.11	processors	262
7.66.4.12	routers	262
7.66.4.13	vcs	263
7.67	MessageState Class Reference	264
7.67.1	Detailed Description	264
7.67.2	Constructor & Destructor Documentation	264
7.67.2.1	MessageState	264
7.67.2.2	~MessageState	264
7.67.3	Member Function Documentation	265
7.67.3.1	toString	265
7.67.4	Member Data Documentation	265
7.67.4.1	arrival_time	265
7.67.4.2	clear_message	265

7.67.4.3 credits_sent	265
7.67.4.4 flits_in_ib	265
7.67.4.5 input_channel	265
7.67.4.6 input_port	265
7.67.4.7 length	265
7.67.4.8 output_channel	265
7.67.4.9 output_port	266
7.67.4.10 pipe_stage	266
7.67.4.11 possible_oports	266
7.67.4.12 possible_ovcs	266
7.68 MSHR_H Class Reference	267
7.68.1 Detailed Description	267
7.68.2 Constructor & Destructor Documentation	268
7.68.2.1 MSHR_H	268
7.68.2.2 ~MSHR_H	268
7.68.3 Member Function Documentation	268
7.68.3.1 countBLP	268
7.68.3.2 DeleteInMSHR	268
7.68.3.3 demap_addr	268
7.68.3.4 GlobalAddrMap	269
7.68.3.5 local_map_addr	269
7.68.3.6 map_addr	269
7.68.3.7 process_event	269
7.68.3.8 toString	270
7.68.4 Member Data Documentation	270
7.68.4.1 child	270
7.68.4.2 countOnce	270
7.68.4.3 done	270
7.68.4.4 filename	270
7.68.4.5 globalUnSink	270
7.68.4.6 id	270
7.68.4.7 lastFinishTime	270
7.68.4.8 lastFullTime	271
7.68.4.9 lastScheduledIndex	271
7.68.4.10 mshr	271
7.68.4.11 nextReq	271

7.68.4.12 parent	271
7.68.4.13 trace_filename	271
7.68.4.14 unsink	271
7.68.4.15 waiting	271
7.68.4.16 waitingForMSHR	272
7.68.4.17 writeQueue	272
7.69 NetworkComponent Class Reference	273
7.69.1 Detailed Description	273
7.69.2 Member Enumeration Documentation	273
7.69.2.1 types	273
7.69.3 Constructor & Destructor Documentation	274
7.69.3.1 NetworkComponent	274
7.69.3.2 ~NetworkComponent	274
7.69.4 Member Function Documentation	274
7.69.4.1 process_event	274
7.69.4.2 toString	275
7.69.5 Member Data Documentation	275
7.69.5.1 address	275
7.69.5.2 name	275
7.69.5.3 node_ip	276
7.69.5.4 type	276
7.70 NI Class Reference	277
7.70.1 Detailed Description	278
7.70.2 Constructor & Destructor Documentation	278
7.70.2.1 NI	278
7.70.2.2 ~NI	278
7.70.3 Member Function Documentation	278
7.70.3.1 compare	278
7.70.3.2 convertFromBitStream	278
7.70.3.3 convertToBitStream	278
7.70.3.4 finish	279
7.70.3.5 GetFromNIQueue	279
7.70.3.6 handle_new_packet_event	279
7.70.3.7 handle_old_packet_event	279
7.70.3.8 handle_out_pull_event	279
7.70.3.9 handle_ready_event	280

7.70.3.10 print_stats	280
7.70.3.11 process_event	280
7.70.3.12 set_no_vcs	281
7.70.3.13 set_output_path	281
7.70.3.14 setup	281
7.70.3.15 toString	281
7.70.4 Member Data Documentation	281
7.70.4.1 last_pkt_out_cycle	281
7.70.4.2 last_vc	282
7.70.4.3 max_sim_time	282
7.70.4.4 mc	282
7.70.4.5 missed_time	282
7.70.4.6 niQueue	282
7.70.4.7 no_nodes	282
7.70.4.8 node_ip	282
7.70.4.9 out_file	282
7.70.4.10 out_filename	282
7.70.4.11 out_packets	283
7.70.4.12 outstanding_hlp	283
7.70.4.13 packets	283
7.70.4.14 packets_out	283
7.70.4.15 ready	283
7.70.4.16 sending	283
7.70.4.17 sent_packets	283
7.70.4.18 total_backward_time	283
7.70.4.19 total_missed_time	283
7.70.4.20 trace_filename	283
7.70.4.21 trace_name	284
7.70.4.22 vcs	284
7.71 Output0< OBJ > Class Template Reference	285
7.71.1 Detailed Description	285
7.71.2 Constructor & Destructor Documentation	285
7.71.2.1 Output0	285
7.71.3 Member Function Documentation	285
7.71.3.1 CallHandler	285
7.71.4 Member Data Documentation	285

7.71.4.1	handler	285
7.71.4.2	obj	286
7.72	OutputBase Class Reference	287
7.72.1	Detailed Description	287
7.72.2	Constructor & Destructor Documentation	287
7.72.2.1	OutputBase	287
7.72.3	Member Function Documentation	287
7.72.3.1	CallHandler	287
7.72.4	Member Data Documentation	287
7.72.4.1	componentId	287
7.72.4.2	latency	287
7.73	OutputBuffer Class Reference	288
7.73.1	Detailed Description	288
7.73.2	Constructor & Destructor Documentation	288
7.73.2.1	OutputBuffer	288
7.73.2.2	~OutputBuffer	288
7.73.3	Member Function Documentation	289
7.73.3.1	change_pull_channel	289
7.73.3.2	change_push_channel	289
7.73.3.3	get_no_vcs	289
7.73.3.4	get_pull_channel	289
7.73.3.5	get_push_channel	289
7.73.3.6	is_channel_full	289
7.73.3.7	is_empty	289
7.74	Phit Class Reference	290
7.74.1	Detailed Description	290
7.74.2	Constructor & Destructor Documentation	290
7.74.2.1	Phit	290
7.74.2.2	~Phit	290
7.74.3	Member Function Documentation	290
7.74.3.1	toString	290
7.74.4	Member Data Documentation	290
7.74.4.1	data	290
7.75	PortArbiter Class Reference	291
7.75.1	Detailed Description	291
7.75.2	Constructor & Destructor Documentation	291

7.75.2.1	PortArbiter	291
7.75.2.2	~PortArbiter	291
7.75.3	Member Data Documentation	291
7.75.3.1	name	291
7.76	PowerStats Class Reference	292
7.76.1	Detailed Description	292
7.76.2	Constructor & Destructor Documentation	292
7.76.2.1	PowerStats	292
7.76.2.2	~PowerStats	293
7.76.3	Member Function Documentation	293
7.76.3.1	CalcPower	293
7.76.4	Member Data Documentation	293
7.76.4.1	Pow_ACT_PDN	293
7.76.4.2	Pow_ACT_PRE	293
7.76.4.3	Pow_ACT_STBY	293
7.76.4.4	Pow_AP	293
7.76.4.5	Pow_Bkg	293
7.76.4.6	Pow_DQ	293
7.76.4.7	Pow_PRE_PDN	294
7.76.4.8	Pow_PRE_STBY	294
7.76.4.9	Pow_RD	294
7.76.4.10	Pow_RD_WR_Tr	294
7.76.4.11	Pow_REF	294
7.76.4.12	Pow_Term	294
7.76.4.13	Pow_termRDoth	294
7.76.4.14	Pow_termWR	294
7.76.4.15	Pow_termWRoth	294
7.76.4.16	Pow_Total	294
7.76.4.17	Pow_WR	295
7.77	Processor Class Reference	296
7.77.1	Detailed Description	296
7.77.2	Constructor & Destructor Documentation	296
7.77.2.1	Processor	296
7.77.2.2	~Processor	296
7.77.3	Member Function Documentation	296
7.77.3.1	init	296

7.77.3.2	print_stats	296
7.77.3.3	process_event	297
7.77.3.4	set_output_path	297
7.77.3.5	setup	297
7.77.3.6	toString	297
7.77.4	Member Data Documentation	297
7.77.4.1	interface_connections	297
7.78	PToPSwitchArbiter Class Reference	298
7.78.1	Detailed Description	298
7.78.2	Constructor & Destructor Documentation	298
7.78.2.1	PToPSwitchArbiter	298
7.78.2.2	~PToPSwitchArbiter	299
7.78.3	Member Function Documentation	299
7.78.3.1	clear_requested	299
7.78.3.2	clear_winner	299
7.78.3.3	do_fcfs_arbitration	299
7.78.3.4	do_priority_round_robin_arbitration	300
7.78.3.5	do_round_robin_arbitration	300
7.78.3.6	is_empty	300
7.78.3.7	is_requested	300
7.78.3.8	pick_winner	301
7.78.3.9	request	301
7.78.3.10	resize	301
7.78.3.11	toString	301
7.78.3.12	update_msg_class	302
7.78.4	Member Data Documentation	302
7.78.4.1	address	302
7.78.4.2	CHANNELS	302
7.78.4.3	done	302
7.78.4.4	last_port_winner	302
7.78.4.5	last_winner	302
7.78.4.6	locked	302
7.78.4.7	name	302
7.78.4.8	node_ip	303
7.78.4.9	port_locked	303
7.78.4.10	PORTS	303

7.78.4.11 priority_reqs	303
7.78.4.12 requested	303
7.78.4.13 requesting_inputs	303
7.79 PVToPV_swa Class Reference	304
7.79.1 Detailed Description	304
7.79.2 Constructor & Destructor Documentation	304
7.79.2.1 PVToPV_swa	304
7.79.2.2 ~PVToPV_swa	304
7.79.3 Member Function Documentation	305
7.79.3.1 clear_winner	305
7.79.3.2 is_empty	305
7.79.3.3 is_empty_for_ch	305
7.79.3.4 is_requested	305
7.79.3.5 no_requests_ch	305
7.79.3.6 pick_winner	305
7.79.3.7 request	305
7.79.3.8 resize	305
7.79.3.9 toString	305
7.79.4 Member Data Documentation	306
7.79.4.1 address	306
7.79.4.2 CHANNELS	306
7.79.4.3 done	306
7.79.4.4 last_ch_winner	306
7.79.4.5 last_port_winner	306
7.79.4.6 last_winner	306
7.79.4.7 locked	306
7.79.4.8 name	306
7.79.4.9 node_ip	306
7.79.4.10 port_locked	306
7.79.4.11 PORTS	307
7.79.4.12 requested	307
7.79.4.13 requesting_inputs	307
7.80 RankHandler Class Reference	308
7.80.1 Detailed Description	308
7.80.2 Constructor & Destructor Documentation	308
7.80.2.1 RankHandler	308

7.80.2.2	~RankHandler	308
7.80.3	Member Data Documentation	308
7.80.3.1	bank	308
7.80.3.2	prevReadsOWrite	308
7.80.3.3	rankId	309
7.80.3.4	rbuffer	309
7.80.3.5	readsOWrite	309
7.81	RankState Struct Reference	310
7.81.1	Detailed Description	310
7.81.2	Member Data Documentation	310
7.81.2.1	bank	310
7.82	RefreshMgr Class Reference	311
7.82.1	Detailed Description	311
7.82.2	Constructor & Destructor Documentation	311
7.82.2.1	RefreshMgr	311
7.82.2.2	~RefreshMgr	311
7.82.3	Member Function Documentation	311
7.82.3.1	process_event	311
7.82.3.2	toString	312
7.82.4	Member Data Documentation	312
7.82.4.1	currentRankNo	312
7.82.4.2	currentRowNo	312
7.82.4.3	mc	312
7.82.4.4	nextRefresh	312
7.82.4.5	reqPtr	312
7.83	Request Class Reference	313
7.83.1	Detailed Description	313
7.83.2	Constructor & Destructor Documentation	313
7.83.2.1	Request	313
7.83.2.2	~Request	314
7.83.3	Member Data Documentation	314
7.83.3.1	address	314
7.83.3.2	arrivalTime	314
7.83.3.3	avg_network_latency	314
7.83.3.4	bankNo	314
7.83.3.5	busInsertionTime	314

7.83.3.6 cbufferInsertionTime	315
7.83.3.7 channelNo	315
7.83.3.8 cmdType	315
7.83.3.9 columnNo	315
7.83.3.10 data	315
7.83.3.11 dimmNo	315
7.83.3.12 hop_count	315
7.83.3.13 local	316
7.83.3.14 lowerBits	316
7.83.3.15 mark	316
7.83.3.16 mcNo	316
7.83.3.17 rankNo	316
7.83.3.18 rbufferInsertionTime	316
7.83.3.19 retireTime	316
7.83.3.20 rowNo	316
7.83.3.21 scheduledInMSHR	317
7.83.3.22 serviced	317
7.83.3.23 startTime	317
7.83.3.24 status	317
7.83.3.25 tag	317
7.83.3.26 threadId	317
7.83.3.27 throttleTime	317
7.84 RequestHandler Class Reference	318
7.84.1 Detailed Description	318
7.84.2 Constructor & Destructor Documentation	318
7.84.2.1 RequestHandler	318
7.84.2.2 ~RequestHandler	319
7.84.3 Member Function Documentation	319
7.84.3.1 FormBatch	319
7.84.3.2 MarkAll	319
7.84.3.3 MarkBatchOnly	319
7.84.3.4 process_event	319
7.84.3.5 PushPipeline	320
7.84.3.6 SetLinks	320
7.84.3.7 toString	320
7.84.4 Member Data Documentation	320

7.84.4.1	addrMap	320
7.84.4.2	busHandler	320
7.84.4.3	chan	321
7.84.4.4	child	321
7.84.4.5	lastBatchFormTime	321
7.84.4.6	mc	321
7.84.4.7	oneBufferFull	321
7.84.4.8	parent	321
7.84.4.9	pipeline	321
7.84.4.10	pipelineFilled	321
7.84.4.11	reqTag	321
7.84.4.12	resPtr	322
7.85	ResponseHandler Class Reference	323
7.85.1	Detailed Description	323
7.85.2	Constructor & Destructor Documentation	323
7.85.2.1	ResponseHandler	323
7.85.2.2	~ResponseHandler	324
7.85.2.3	ResponseHandler	324
7.85.2.4	~ResponseHandler	324
7.85.3	Member Function Documentation	324
7.85.3.1	CanStart	324
7.85.3.2	CanStart	324
7.85.3.3	IsBufferFull	324
7.85.3.4	IsBufferFull	324
7.85.3.5	process_event	324
7.85.3.6	process_event	324
7.85.3.7	SearchBuffer	325
7.85.3.8	SearchBuffer	325
7.85.3.9	SendServiced	325
7.85.3.10	toString	325
7.85.3.11	toString	325
7.85.4	Member Data Documentation	325
7.85.4.1	bufferFull	325
7.85.4.2	child	325
7.85.4.3	mc	325
7.85.4.4	out_pull_scheduled	325

7.85.4.5 parent	326
7.85.4.6 reqPtr	326
7.85.4.7 responseBuffer	326
7.85.4.8 serviced	326
7.85.4.9 stoppedQueue	326
7.86 RouteEntry Class Reference	327
7.86.1 Detailed Description	327
7.86.2 Constructor & Destructor Documentation	327
7.86.2.1 RouteEntry	327
7.86.3 Member Data Documentation	327
7.86.3.1 channels	327
7.86.3.2 destination	327
7.86.3.3 ports	327
7.87 Router Class Reference	328
7.87.1 Detailed Description	328
7.87.2 Constructor & Destructor Documentation	328
7.87.2.1 Router	328
7.87.2.2 ~Router	328
7.87.3 Member Function Documentation	328
7.87.3.1 init	328
7.87.3.2 print_stats	329
7.87.3.3 set_no_nodes	329
7.87.3.4 toString	329
7.87.4 Member Data Documentation	329
7.87.4.1 buffer_size	329
7.87.4.2 credits	329
7.87.4.3 input_connections	329
7.87.4.4 output_connections	330
7.87.4.5 ports	330
7.87.4.6 vcs	330
7.88 RouterFourStageVcs Class Reference	331
7.88.1 Detailed Description	332
7.88.2 Constructor & Destructor Documentation	332
7.88.2.1 RouterFourStageVcs	332
7.88.2.2 ~RouterFourStageVcs	332
7.88.3 Member Function Documentation	332

7.88.3.1	check_all_conditions	332
7.88.3.2	check_tick	332
7.88.3.3	handle_link_arrival_event	332
7.88.3.4	handle_tick_event	332
7.88.3.5	init	333
7.88.3.6	print_stats	333
7.88.3.7	process_event	333
7.88.3.8	set_buffer_size	333
7.88.3.9	set_grid_x_location	333
7.88.3.10	set_grid_y_location	334
7.88.3.11	set_no_credits	334
7.88.3.12	set_no_nodes	334
7.88.3.13	set_no_ports	334
7.88.3.14	set_no_vcs	334
7.88.3.15	toString	334
7.88.4	Member Data Documentation	334
7.88.4.1	buffer_size	334
7.88.4.2	credits	335
7.88.4.3	decoders	335
7.88.4.4	flits	335
7.88.4.5	in_arbiters	335
7.88.4.6	in_buffers	335
7.88.4.7	packets	335
7.88.4.8	port_arbiters	335
7.88.4.9	ports	335
7.88.4.10	ticking	336
7.88.4.11	total_packet_latency	336
7.88.4.12	vcs	336
7.88.4.13	xbar	336
7.89	SA_unit Class Reference	337
7.89.1	Detailed Description	337
7.89.2	Constructor & Destructor Documentation	337
7.89.2.1	SA_unit	337
7.89.3	Member Data Documentation	337
7.89.3.1	ch	337
7.89.3.2	in_time	337

7.89.3.3	port	337
7.90	Simulator Class Reference	338
7.90.1	Detailed Description	339
7.90.2	Member Function Documentation	339
7.90.2.1	Cancel	339
7.90.2.2	getComponentDesc	339
7.90.2.3	GetEarliestEvent	339
7.90.2.4	MyRank	339
7.90.2.5	Now	339
7.90.2.6	Peek	340
7.90.2.7	registerComponent	340
7.90.2.8	Run	340
7.90.2.9	Schedule	341
7.90.2.10	Schedule	341
7.90.2.11	Schedule	341
7.90.2.12	Schedule	341
7.90.2.13	Schedule	341
7.90.2.14	Schedule	342
7.90.2.15	Schedule	342
7.90.2.16	Schedule	342
7.90.2.17	Schedule	342
7.90.2.18	Schedule	342
7.90.2.19	Stop	343
7.90.2.20	StopAt	343
7.90.3	Member Data Documentation	343
7.90.3.1	components	343
7.90.3.2	events	344
7.90.3.3	halted	344
7.90.3.4	nextComponentID	344
7.90.3.5	rank	344
7.90.3.6	simTime	344
7.91	Statistic Class Reference	345
7.91.1	Detailed Description	346
7.91.2	Member Function Documentation	346
7.91.2.1	CalculateAggregateStats	346
7.91.2.2	CollectStatsPerCycle	347

7.91.2.3	CollectStatsPerRequest	347
7.91.2.4	InitStats	347
7.91.2.5	PrintAggregateStats	348
7.91.3	Member Data Documentation	348
7.91.3.1	avgBusMemDelayPerBank	348
7.91.3.2	avgBusMemDelayPerThread	348
7.91.3.3	avgCBufDelayPerBank	349
7.91.3.4	avgCBufDelayPerThread	349
7.91.3.5	avgFrontSideTimePerBank	349
7.91.3.6	avgFrontSideTimePerThread	349
7.91.3.7	avgLatPerBank	349
7.91.3.8	avgLatPerThread	349
7.91.3.9	avgLatPerThreadPerBank	349
7.91.3.10	avgQueueDelayPerBank	349
7.91.3.11	avgQueueDelayPerThread	349
7.91.3.12	avgThrottlePerBank	350
7.91.3.13	avgThrottlePerThread	350
7.91.3.14	avgThrottlePerThreadPerBank	350
7.91.3.15	BLPPerThread	350
7.91.3.16	BNK_PRE	350
7.91.3.17	bufferFullCycles	350
7.91.3.18	CLK_LO_ACT	350
7.91.3.19	CLK_LO_PRE	350
7.91.3.20	doneOnce	350
7.91.3.21	emptyCycles	351
7.91.3.22	hitRatePerBank	351
7.91.3.23	hitRatePerThread	351
7.91.3.24	hitRatePerThreadPerBank	351
7.91.3.25	hitsPerBank	351
7.91.3.26	hitsPerThread	351
7.91.3.27	hitsPerThreadPerBank	351
7.91.3.28	latPerBank	351
7.91.3.29	latPerThread	351
7.91.3.30	latPerThreadPerBank	352
7.91.3.31	mc	352
7.91.3.32	PH	352

7.91.3.33 pwr	352
7.91.3.34 rbufferOccupancy	352
7.91.3.35 rbufferOccupancyRatio	352
7.91.3.36 RDsch	352
7.91.3.37 reqPerBank	352
7.91.3.38 reqPerThread	352
7.91.3.39 reqPerThreadPerBank	353
7.91.3.40 RRDsch	353
7.91.3.41 termRDsch	353
7.91.3.42 termWRsch	353
7.91.3.43 throttlePerBank	353
7.91.3.44 throttlePerThread	353
7.91.3.45 throttlePerThreadPerBank	353
7.91.3.46 TotalBusMemDelayPerBank	353
7.91.3.47 TotalBusMemDelayPerThread	353
7.91.3.48 TotalBusMemDelayPerThreadPerBank	354
7.91.3.49 TotalCBufDelayPerBank	354
7.91.3.50 TotalCBufDelayPerThread	354
7.91.3.51 TotalCBufDelayPerThreadPerBank	354
7.91.3.52 TotalFrontSideTimePerBank	354
7.91.3.53 TotalFrontSideTimePerThread	354
7.91.3.54 TotalFrontSideTimePerThreadPerBank	354
7.91.3.55 TotalQueueDelayPerBank	354
7.91.3.56 TotalQueueDelayPerThread	355
7.91.3.57 TotalQueueDelayPerThreadPerBank	355
7.91.3.58 WRsch	355
7.92 TailFlit Class Reference	356
7.92.1 Detailed Description	356
7.92.2 Constructor & Destructor Documentation	356
7.92.2.1 TailFlit	356
7.92.2.2 ~TailFlit	356
7.92.3 Member Function Documentation	357
7.92.3.1 populate_tail_flit	357
7.92.3.2 toString	357
7.92.4 Member Data Documentation	357
7.92.4.1 avg_network_latency	357

7.92.4.2	hop_count	357
7.92.4.3	packet_originated_time	357
7.92.4.4	scratch_pad_time	357
7.93	VirtualChannelArbiter Class Reference	358
7.93.1	Detailed Description	358
7.93.2	Constructor & Destructor Documentation	358
7.93.2.1	VirtualChannelArbiter	358
7.93.3	Member Function Documentation	358
7.93.3.1	clear	358
7.93.3.2	clear_winner	358
7.93.3.3	get_requests	358
7.93.3.4	pick_winner	358
7.93.3.5	pull_winner	358
7.93.3.6	ready	358
7.93.3.7	request	358
7.93.3.8	resize	358
7.93.3.9	size	358
7.94	VirtualChannelDescription Class Reference	359
7.94.1	Detailed Description	359
7.94.2	Constructor & Destructor Documentation	359
7.94.2.1	VirtualChannelDescription	359
7.94.3	Member Data Documentation	359
7.94.3.1	port	359
7.94.3.2	vc	359
8	File Documentation	361
8.1	addr_map.cc File Reference	361
8.1.1	Variable Documentation	361
8.1.1.1	BANK_BITS	361
8.2	addr_map.h File Reference	362
8.3	addressDecoder.h File Reference	363
8.4	arbiter.h File Reference	364
8.5	bank_handler.cc File Reference	365
8.6	bank_handler.h File Reference	366
8.6.1	Enumeration Type Documentation	366
8.6.1.1	CallerType	366
8.7	buffer.cc File Reference	367

8.8	buffer.h File Reference	368
8.9	bus.cc File Reference	369
8.10	bus.h File Reference	370
8.11	bus_handler.cc File Reference	371
8.12	bus_handler.h File Reference	372
8.13	channel_handler.cc File Reference	373
8.14	channel_handler.h File Reference	374
8.15	cmd_bus_handler.cc File Reference	375
8.16	cmd_bus_handler.h File Reference	376
8.17	cmd_issuer.cc File Reference	377
8.18	cmd_issuer.h File Reference	378
8.18.1	Typedef Documentation	379
8.18.1.1	BankState	379
8.18.1.2	BurstLength	379
8.18.1.3	ChannelState	379
8.18.1.4	CmdQueue	379
8.18.1.5	DRAMCmd	379
8.18.1.6	DRAMCmdState	379
8.18.1.7	RankState	379
8.18.2	Enumeration Type Documentation	379
8.18.2.1	BurstLength	379
8.18.2.2	DRAMCmd	379
8.18.2.3	RankBankComb	380
8.19	component.cc File Reference	381
8.20	component.h File Reference	382
8.21	constants.h File Reference	383
8.21.1	Define Documentation	384
8.21.1.1	BATCH_FORM_TIME	384
8.21.1.2	BLOCKS_PER_ROW	384
8.21.1.3	BUS_CYCLE	384
8.21.1.4	BUS_SPEED	384
8.21.1.5	CACHE_BLOCK_SIZE	384
8.21.1.6	COLUMN_SIZE	385
8.21.1.7	CORE_SPEED	385
8.21.1.8	CYCLE_2_NS	385
8.21.1.9	DDR_BUS_WIDTH	385

8.21.1.10 DRAM_SIZE	385
8.21.1.11 FR_FCFS	385
8.21.1.12 MAX_BATCH_SIZE	385
8.21.1.13 MAX_BUFFER_SIZE	385
8.21.1.14 MAX_CMD_BUFFER_SIZE	385
8.21.1.15 MAX_READ_OV_WRITE	385
8.21.1.16 MEM_CYCLE	386
8.21.1.17 MEM_SPEED	386
8.21.1.18 MSHR_SIZE	386
8.21.1.19 NETWORK_ADDRESS_BITS	386
8.21.1.20 NETWORK_COMMAND_BITS	386
8.21.1.21 NETWORK_THREADID_BITS	386
8.21.1.22 NO_OF_BANKS	386
8.21.1.23 NO_OF_BUFFERS	386
8.21.1.24 NO_OF_CHANNELS	387
8.21.1.25 NO_OF_COLUMNS	387
8.21.1.26 NO_OF_RANKS	387
8.21.1.27 NO_OF_ROWS	387
8.21.1.28 NO_OF_THREADS	387
8.21.1.29 OPEN_PAGE_POLICY	387
8.21.1.30 PAGE_INTERLEAVING	387
8.21.1.31 PREFETCH_SIZE	388
8.21.1.32 READ_SIZE	388
8.21.1.33 REFRESH_INC	388
8.21.1.34 REFRESH_PERIOD	388
8.21.1.35 RESPONSE_BUFFER_SIZE	388
8.21.1.36 ROW_SIZE	388
8.21.1.37 t_AL	388
8.21.1.38 t_CAS	388
8.21.1.39 t_CCD	388
8.21.1.40 t_CMD	389
8.21.1.41 t_CWD	389
8.21.1.42 t_OST	389
8.21.1.43 t_RAS	389
8.21.1.44 t_RC	389
8.21.1.45 t_RCD	389

8.21.1.46 t_RFC	389
8.21.1.47 t_RP	389
8.21.1.48 t_RRD	389
8.21.1.49 t RTP	390
8.21.1.50 t_RTRS	390
8.21.1.51 t_WR	390
8.21.1.52 t_WTR	390
8.21.1.53 TAG_BITS	390
8.21.1.54 tRAS	390
8.21.1.55 tRC	390
8.21.1.56 tREFI	390
8.21.1.57 tRFC	390
8.21.1.58 USE_MSHR	390
8.21.1.59 WRITE_SIZE	391
8.21.1.60 WRITEBACK_SIZE	391
8.21.2 Typedef Documentation	391
8.21.2.1 Addr_t	391
8.21.2.2 Time	391
8.21.2.3 UInt	391
8.22 crossbar.h File Reference	392
8.23 data_bus_handler.cc File Reference	393
8.24 data_bus_handler.h File Reference	394
8.25 dram.cc File Reference	395
8.26 dram.h File Reference	396
8.27 flit.cc File Reference	397
8.28 flit.h File Reference	398
8.28.1 Enumeration Type Documentation	398
8.28.1.1 flit_type	398
8.28.2 Variable Documentation	398
8.28.2.1 max_phy_link_bits	398
8.29 genericArbiter.cc File Reference	399
8.30 genericArbiter.h File Reference	400
8.31 genericBuffer.cc File Reference	401
8.32 genericBuffer.h File Reference	402
8.32.1 Define Documentation	402
8.32.1.1 BUFFER_SIZE	402

8.33 genericComponentHeader.h File Reference	403
8.33.1 Define Documentation	404
8.33.1.1 _DBG	404
8.33.1.2 _DBG_NOARG	404
8.33.1.3 CREDIT_ID	404
8.33.1.4 DEFAULT_ADDRESS	405
8.33.1.5 DEFAULT_CONVERT_PACKET_CYCLES	405
8.33.1.6 FLIT_ID	405
8.33.1.7 LOC	405
8.33.1.8 NO_DATA	405
8.33.2 Typedef Documentation	405
8.33.2.1 simTime	405
8.33.2.2 uint	405
8.33.2.3 ullint	405
8.33.2.4 uniqueId	405
8.33.3 Enumeration Type Documentation	406
8.33.3.1 message_class	406
8.33.4 Variable Documentation	406
8.33.4.1 max_control_bits	406
8.33.4.2 max_network_node_bits	406
8.33.4.3 max_tail_length_bits	406
8.33.4.4 max_transaction_id_bits	406
8.34 genericCrossbar.cc File Reference	407
8.35 genericCrossbar.h File Reference	408
8.36 genericData.cc File Reference	409
8.37 genericData.h File Reference	410
8.37.1 Enumeration Type Documentation	410
8.37.1.1 RouterPipeStage	410
8.37.1.2 ROUTING_SCHEME	411
8.37.1.3 SW_ARBITRATION	411
8.38 genericEvents.h File Reference	412
8.38.1 Define Documentation	412
8.38.1.1 ADDRESS_DECODE_EVENT	412
8.38.1.2 CHECK_IN_ARBITER_EVENT	412
8.38.1.3 CHECK_OUT_ARBITER_EVENT	412
8.38.1.4 CONFIGURE_CROSSBAR_EVENT	413

8.38.1.5 CONTINUE	413
8.38.1.6 CREDIT_EVENT	413
8.38.1.7 FLIT_OUT_EVENT	413
8.38.1.8 IN_ARBITRATE_EVENT	413
8.38.1.9 IN_BUFFER_EVENT	413
8.38.1.10 IN_PULL_EVENT	413
8.38.1.11 IN_PUSH_EVENT	413
8.38.1.12 LINK_ARRIVAL_EVENT	413
8.38.1.13 MSHR_DELETE	414
8.38.1.14 NEW_PACKET_EVENT	414
8.38.1.15 OLD_PACKET_EVENT	414
8.38.1.16 OUT_ARBITRATE_EVENT	414
8.38.1.17 OUT_PULL_EVENT	414
8.38.1.18 OUT_PUSH_EVENT	415
8.38.1.19 PORT_ARBITRATE_EVENT	415
8.38.1.20 PUSH_BUFFER	415
8.38.1.21 READY_EVENT	415
8.38.1.22 REPLY	415
8.38.1.23 SEND_TO_NI	415
8.38.1.24 START	415
8.38.1.25 START_CMD_QUEUE	415
8.38.1.26 START_READ	416
8.38.1.27 START_SUBCOMPONENT	416
8.38.1.28 START_WRITE	416
8.38.1.29 STOP	416
8.38.1.30 STOP_CMD_QUEUE	416
8.38.1.31 STOP_SUBCOMPONENT	416
8.38.1.32 SWAP_VC_EVENT	416
8.38.1.33 TICK_EVENT	416
8.38.1.34 TRAVERSE_CROSSBAR_EVENT	417
8.38.1.35 VC_ARBITRATE_EVENT	417
8.39 genericFlatMc.cc File Reference	418
8.40 genericFlatMc.h File Reference	419
8.40.1 Define Documentation	419
8.40.1.1 DEFAULT_RAN_MAX_TIME	419
8.40.1.2 MAX	419

8.40.1.3	MAX_ADDRESS	419
8.40.1.4	MIN	419
8.41	genericInterface.cc File Reference	420
8.42	genericInterface.h File Reference	421
8.42.1	Define Documentation	421
8.42.1.1	DEFAULT_NO_OF_CREDITS	421
8.42.2	Variable Documentation	422
8.42.2.1	do_two_stage_router	422
8.43	genericLink.cc File Reference	423
8.44	genericLink.h File Reference	424
8.44.1	Variable Documentation	424
8.44.1.1	max_sim_time	424
8.45	genericPortArbiter.cc File Reference	425
8.46	genericPortArbiter.h File Reference	426
8.47	genericRC.cc File Reference	427
8.48	genericRC.h File Reference	428
8.48.1	Variable Documentation	428
8.48.1.1	grid_size	428
8.48.1.2	no_nodes	428
8.48.1.3	rc_method	428
8.49	genericRouterAdaptive.cc File Reference	429
8.50	genericRouterAdaptive.h File Reference	430
8.50.1	Variable Documentation	430
8.50.1.1	do_two_stage_router	430
8.50.1.2	send_early_credit	430
8.51	genericRouterNoVcs.cc File Reference	431
8.52	genericRouterNoVcs.cc File Reference	432
8.53	genericRouterNoVcs.h File Reference	433
8.53.1	Enumeration Type Documentation	433
8.53.1.1	GenericRouterNoVcsPipeStage	433
8.54	genericRouterNoVcs.h File Reference	434
8.54.1	Variable Documentation	434
8.54.1.1	send_early_credit	434
8.55	genericRouterVct.cc File Reference	435
8.56	genericRouterVct.h File Reference	436
8.56.1	Variable Documentation	436

8.56.1.1	do_two_stage_router	436
8.56.1.2	send_early_credit	436
8.57	genericRPG.cc File Reference	437
8.58	genericRPG.cc File Reference	438
8.58.1	Define Documentation	438
8.58.1.1	MAX_SIM_TIME	438
8.59	genericRPG.h File Reference	439
8.59.1	Define Documentation	439
8.59.1.1	DEFAULT_RAN_ADDRESS	439
8.59.1.2	DEFAULT_RAN_DESTINATION_TYPE	440
8.59.1.3	DEFAULT_RAN_LAMDA	440
8.59.1.4	DEFAULT_RAN_LENGTH_TYPE	440
8.59.1.5	DEFAULT_RAN_MAX_TIME	440
8.59.1.6	DEFAULT_RAN_MAX_VC	440
8.59.1.7	DEFAULT_RAN_SEED	440
8.59.1.8	DEFAULT_RAN_TRACE_FILE_NAME	440
8.59.1.9	HOT_SPOTS	440
8.59.1.10	IN_OUT_MISMATCH	440
8.59.1.11	MAX	440
8.59.1.12	MAX_ADDRESS	440
8.59.1.13	MAX_DELAY	440
8.59.1.14	MAX_LENGTH	441
8.59.1.15	MIN	441
8.59.1.16	MIN_DELAY	441
8.59.1.17	MIN_LENGTH	441
8.59.1.18	REPORT_BASE	441
8.59.1.19	SIM_SUCCESS	441
8.59.2	Variable Documentation	441
8.59.2.1	run_destination_type	441
8.60	genericRPG.h File Reference	442
8.60.1	Define Documentation	442
8.60.1.1	DEFAULT_RAN_ADDRESS	442
8.60.1.2	DEFAULT_RAN_DESTINATION_TYPE	443
8.60.1.3	DEFAULT_RAN_LAMDA	443
8.60.1.4	DEFAULT_RAN_LENGTH_TYPE	443
8.60.1.5	DEFAULT_RAN_MAX_TIME	443

8.60.1.6	DEFAULT_RAN_MAX_VC	443
8.60.1.7	DEFAULT_RAN_SEED	443
8.60.1.8	DEFAULT_RAN_TRACE_FILE_NAME	443
8.60.1.9	HOT_SPOTS	443
8.60.1.10	IN_OUT_MISMATCH	443
8.60.1.11	MAX	443
8.60.1.12	MAX_ADDRESS	443
8.60.1.13	MAX_DELAY	443
8.60.1.14	MAX_LENGTH	444
8.60.1.15	MIN	444
8.60.1.16	MIN_DELAY	444
8.60.1.17	MIN_LENGTH	444
8.60.1.18	REPORT_BASE	444
8.60.1.19	SIM_SUCCESS	444
8.60.2	Variable Documentation	444
8.60.2.1	run_destination_type	444
8.61	genericSink.cc File Reference	445
8.62	genericSink.h File Reference	446
8.63	genericTPG.cc File Reference	447
8.63.1	Variable Documentation	447
8.63.1.1	MC_ADDR_BITS	447
8.64	genericTPG.h File Reference	448
8.64.1	Define Documentation	448
8.64.1.1	DEFAULT_RAN_MAX_TIME	448
8.64.1.2	MAX	448
8.64.1.3	MAX_ADDRESS	449
8.64.1.4	MIN	449
8.64.2	Variable Documentation	449
8.64.2.1	no_mcs	449
8.64.2.2	no_nodes	449
8.65	genericTPG_temp.cc File Reference	450
8.66	genericVcArbiter.cc File Reference	451
8.67	genericVcArbiter.h File Reference	452
8.68	genericVcsInterface.cc File Reference	453
8.69	genericVcsInterface.h File Reference	454
8.69.1	Define Documentation	454

8.69.1.1	DEFAULT_NO_OF_CREDITS	454
8.69.2	Variable Documentation	454
8.69.2.1	do_two_stage_router	454
8.70	highLevelPacket.cc File Reference	455
8.71	highLevelPacket.h File Reference	456
8.71.1	Define Documentation	456
8.71.1.1	CONTROL_VECTOR_SIZE	456
8.71.2	Enumeration Type Documentation	456
8.71.2.1	virtual_network	456
8.72	inputBuffer.h File Reference	457
8.72.1	Typedef Documentation	457
8.72.1.1	uint	457
8.73	interface.cc File Reference	458
8.74	interface.h File Reference	459
8.75	irisEvent.cc File Reference	460
8.76	irisEvent.h File Reference	461
8.77	irisLink.cc File Reference	462
8.78	irisLink.h File Reference	463
8.79	link.cc File Reference	464
8.80	link.h File Reference	465
8.80.1	Function Documentation	465
8.80.1.1	addOutput	465
8.81	lowLevelPacket.cc File Reference	466
8.82	lowLevelPacket.h File Reference	467
8.83	main.cc File Reference	468
8.83.1	Function Documentation	468
8.83.1.1	GetNextRequest	468
8.83.1.2	main	469
8.84	main.dox File Reference	470
8.85	MC.cc File Reference	471
8.86	MC.h File Reference	472
8.87	mcFrontEnd.cc File Reference	473
8.88	mcFrontEnd.h File Reference	474
8.89	mesh.cc File Reference	475
8.90	mesh.h File Reference	476
8.91	mshr.cc File Reference	477

8.91.1 Variable Documentation	477
8.91.1.1 THREAD_BITS_POSITION	477
8.92 mshr.h File Reference	478
8.92.1 Typedef Documentation	478
8.92.1.1 BoolVector	478
8.92.2 Variable Documentation	478
8.92.2.1 max_sim_time	478
8.92.2.2 MC_ADDR_BITS	479
8.92.2.3 no_mcs	479
8.93 networkComponent.cc File Reference	480
8.94 networkComponent.h File Reference	481
8.95 NI.cc File Reference	482
8.96 NI.h File Reference	483
8.96.1 Define Documentation	483
8.96.1.1 DEFAULT_RAN_MAX_TIME	483
8.97 outputBuffer.h File Reference	484
8.97.1 Typedef Documentation	484
8.97.1.1 uint	484
8.98 portArbiter.h File Reference	485
8.99 processor.cc File Reference	486
8.100processor.h File Reference	487
8.101ptop_swa.cc File Reference	488
8.102ptop_swa.h File Reference	489
8.102.1 Variable Documentation	489
8.102.1.1 sw_arbitration	489
8.103pvtopv_swa.cc File Reference	490
8.104pvtopv_swa.h File Reference	491
8.105rank_handler.cc File Reference	492
8.106rank_handler.h File Reference	493
8.106.1 Typedef Documentation	493
8.106.1.1 ReqBuffer	493
8.107refresh_manager.cc File Reference	494
8.108refresh_manager.h File Reference	495
8.109request.cc File Reference	496
8.110request.h File Reference	497
8.110.1 Typedef Documentation	497

8.110.1.1 Data	497
8.110.2 Enumeration Type Documentation	497
8.110.2.1 Command_t	497
8.110.2.2 CStatus	498
8.111request_handler.cc File Reference	499
8.112request_handler.h File Reference	500
8.113response_handler.cc File Reference	501
8.114response_handler.h File Reference	502
8.115response_handler_temp.cc File Reference	503
8.116response_handler_temp.h File Reference	504
8.117router.cc File Reference	505
8.118router.h File Reference	506
8.119routerFourStageVcs.cc File Reference	507
8.120routerFourStageVcs.h File Reference	508
8.121simMc2Mesh.cc File Reference	509
8.121.1 Function Documentation	509
8.121.1.1 main	509
8.121.2 Variable Documentation	510
8.121.2.1 BANK_BITS	510
8.121.2.2 do_two_stage_router	510
8.121.2.3 grid_size	510
8.121.2.4 max_phy_link_bits	510
8.121.2.5 max_sim_time	510
8.121.2.6 MC_ADDR_BITS	510
8.121.2.7 no_mcs	510
8.121.2.8 no_nodes	510
8.121.2.9 print_setup	511
8.121.2.10 rc_method	511
8.121.2.11 lsw_arbitration	511
8.121.2.12 THREAD_BITS_POSITION	511
8.122simulator.cc File Reference	512
8.123simulator.h File Reference	513
8.123.1 Typedef Documentation	513
8.123.1.1 ComponentMap_t	513
8.123.1.2 EventSet_t	513
8.124stats.cc File Reference	514

8.125stats.h File Reference	515
8.125.1 Define Documentation	515
8.125.1.1 BL	515
8.125.1.2 DM	515
8.125.1.3 DQS	516
8.125.1.4 IDD0	516
8.125.1.5 IDD2N	516
8.125.1.6 IDD2PF	516
8.125.1.7 IDD2PS	516
8.125.1.8 IDD3N	516
8.125.1.9 IDD3P	516
8.125.1.10 IDD4R	516
8.125.1.11 IDD4W	516
8.125.1.12 IDD5A	516
8.125.1.13 MaxVcc	517
8.125.1.14 MinVcc	517
8.125.1.15 PdqRD	517
8.125.1.16 PdqRDoth	517
8.125.1.17 PdqWR	517
8.125.1.18 PdqWRoth	517
8.125.1.19 SysClk	517
8.125.1.20 SysVdd	517
8.125.1.21 t_CK	517
8.126util.cc File Reference	518
8.126.1 Function Documentation	518
8.126.1.1 timed_cout	518
8.127util.h File Reference	519
8.127.1 Function Documentation	519
8.127.1.1 timed_cout	519
8.128virtualChannelArbiter.h File Reference	520

Chapter 1

Main Page

1.1 General

Iris is a simple network simulator for on-chip networks. It is developed at the Computer Architecture and Systems Laboratory, Georgia Institute of Technology.

Iris uses the Discrete Event Simulation methodology. Some of the models it provides:

- An on chip memory controller model with supporting DDR2 and DDR3 delay models.
- Switch Arbitor Models.
- Routing Techniques.<
 - >

Topologies Supported:

- **Mesh** (p. 259).<
 - > <>

1.2 License

Iris is released under the **GNU Lesser General Public License**.

1.3 Author

Syed Minhaj Hassan. Michelle Rasquinha.

Chapter 2

Directory Hierarchy

2.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

source	27
components	18
impl	22
backup	17
interfaces	24
data_types	19
impl	23
frontend	20
impl	21
kernel	25
MemCtrl	26

Chapter 3

Class Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AddressDecoder	29
Arbiter	33
GenericVcArbiter	202
BankState	44
Buffer	47
GenericBuffer	115
InputBuffer	225
OutputBuffer	288
BusHandler	51
ChannelHandler	54
ChannelState	55
Component	66
AddrMap	31
BankHandler	34
Bus	48
CmdBusHandler	57
CmdIssuer	59
DataBusHandler	73
DRAM	76
DRAMChannel	78
MSHR_H	267
NetworkComponent	273
Interface	230
GenericInterface	133
GenericVcsInterface	206
MCFrontEnd	250
IrisLink	236
GenericLink	141
Processor	296
GenericFlatMc	126
GenericRPG	178
GenericRPG	178

GenericSink	187
GenericTPG	192
NI	277
Router	328
GenericRouterAdaptive	155
GenericRouterNoVcs	163
GenericRouterNoVcs	163
GenericRouterVct	171
RouterFourStageVcs	331
RefreshMgr	311
RequestHandler	318
ResponseHandler	323
ResponseHandler	323
ComponentDescription	69
Crossbar	70
GenericCrossbar	120
Data	72
DRAMCmdState	81
event_less	105
EventBase	106
Event0< T, OBJ >	83
Event0Stat	85
Event1< T, OBJ, U1, T1 >	86
Event1Stat< U1, T1 >	88
Event2< T, OBJ, U1, T1, U2, T2 >	90
Event2Stat< U1, T1, U2, T2 >	92
Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >	94
Event3Stat< U1, T1, U2, T2, U3, T3 >	97
Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >	99
Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >	102
EventId	108
Flit	109
BodyFlit	45
HeadFlit	215
TailFlit	356
GenericArbiter	111
GenericCrossbar::GenericCrossbar::CrossbarUnit	125
GenericRC	149
GenericRC::GenericRC::Address	154
HighLevelPacket	219
InputBufferState	227
IrisEvent	233
Link	238
LinkArrivalData	240
LowLevelPacket	242
MC	247
Mesh	259
MessageState	264
OutputBase	287
Output0< OBJ >	285
Phit	290
PortArbiter	291

GenericPortArbiter	145
PToPSwitchArbiter	298
PVToPV_swa	304
RankHandler	308
RankState	310
Request	313
RouteEntry	327
SA_unit	337
Simulator	338
Statistic	345
PowerStats	292
VirtualChannelArbiter	358
VirtualChannelDescription	359

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

AddressDecoder	29
AddrMap	31
Arbiter	33
BankHandler	34
BankState	44
BodyFlit	45
Buffer	47
Bus	48
BusHandler	51
ChannelHandler	54
ChannelState	55
CmdBusHandler	57
CmdIssuer	59
Component	66
ComponentDescription	69
Crossbar	70
Data	72
DataBusHandler	73
DRAM	76
DRAMChannel	78
DRAMCmdState	81
Event0< T, OBJ >	83
Event0Stat	85
Event1< T, OBJ, U1, T1 >	86
Event1Stat< U1, T1 >	88
Event2< T, OBJ, U1, T1, U2, T2 >	90
Event2Stat< U1, T1, U2, T2 >	92
Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >	94
Event3Stat< U1, T1, U2, T2, U3, T3 >	97
Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >	99
Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >	102
event_less	105
EventBase	106

EventId	108
Flit	109
GenericArbiter	111
GenericBuffer	115
GenericCrossbar	120
GenericCrossbar::GenericCrossbar::CrossbarUnit	125
GenericFlatMc	126
GenericInterface	133
GenericLink	141
GenericPortArbiter	145
GenericRC	149
GenericRC::GenericRC::Address	154
GenericRouterAdaptive	155
GenericRouterNoVcs	163
GenericRouterVct	171
GenericRPG	178
GenericSink	187
GenericTPG	192
GenericVcArbiter	202
GenericVcsInterface	206
HeadFlit	215
HighLevelPacket	219
InputBuffer	225
InputBufferState	227
Interface	230
IrisEvent	233
IrisLink	236
Link	238
LinkArrivalData	240
LowLevelPacket	242
MC	247
MCFrontEnd	250
Mesh	259
MessageState	264
MSHR_H	267
NetworkComponent	273
NI	277
Output0< OBJ >	285
OutputBase	287
OutputBuffer	288
Phit	290
PortArbiter	291
PowerStats	292
Processor	296
PToPSwitchArbiter	298
PVToPV_swa	304
RankHandler	308
RankState	310
RefreshMgr	311
Request	313
RequestHandler	318
ResponseHandler	323
RouteEntry	327
Router	328

RouterFourStageVcs	331
SA_unit	337
Simulator	338
Statistic	345
TailFlit	356
VirtualChannelArbiter	358
VirtualChannelDescription	359

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

addr_map.cc	361
addr_map.h	362
addressDecoder.h	363
arbiter.h	364
bank_handler.cc	365
bank_handler.h	366
buffer.cc	367
buffer.h	368
bus.cc	369
bus.h	370
bus_handler.cc	371
bus_handler.h	372
channel_handler.cc	373
channel_handler.h	374
cmd_bus_handler.cc	375
cmd_bus_handler.h	376
cmd_issuer.cc	377
cmd_issuer.h	378
component.cc	381
component.h	382
constants.h	383
crossbar.h	392
data_bus_handler.cc	393
data_bus_handler.h	394
dram.cc	395
dram.h	396
flit.cc	397
flit.h	398
genericArbiter.cc	399
genericArbiter.h	400
genericBuffer.cc	401
genericBuffer.h	402
genericComponentHeader.h	403

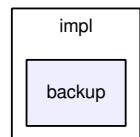
genericCrossbar.cc	407
genericCrossbar.h	408
genericData.cc	409
genericData.h	410
genericEvents.h	412
genericFlatMc.cc	418
genericFlatMc.h	419
genericInterface.cc	420
genericInterface.h	421
genericLink.cc	423
genericLink.h	424
genericPortArbiter.cc	425
genericPortArbiter.h	426
genericRC.cc	427
genericRC.h	428
genericRouterAdaptive.cc	429
genericRouterAdaptive.h	430
backup/genericRouterNoVcs.cc	431
genericRouterNoVcs.cc	432
backup/genericRouterNoVcs.h	433
genericRouterNoVcs.h	434
genericRouterVct.cc	435
genericRouterVct.h	436
backup/genericRPG.cc	437
genericRPG.cc	438
backup/genericRPG.h	439
genericRPG.h	442
genericSink.cc	445
genericSink.h	446
genericTPG.cc	447
genericTPG.h	448
genericTPG_temp.cc	450
genericVcArbiter.cc	451
genericVcArbiter.h	452
genericVcsInterface.cc	453
genericVcsInterface.h	454
highLevelPacket.cc	455
highLevelPacket.h	456
inputBuffer.h	457
interface.cc	458
interface.h	459
irisEvent.cc	460
irisEvent.h	461
irisLink.cc	462
irisLink.h	463
link.cc	464
link.h	465
lowLevelPacket.cc	466
lowLevelPacket.h	467
main.cc	468
MC.cc	471
MC.h	472
mcFrontEnd.cc	473
mcFrontEnd.h	474

mesh.cc	475
mesh.h	476
mshr.cc	477
mshr.h	478
networkComponent.cc	480
networkComponent.h	481
NI.cc	482
NI.h	483
outputBuffer.h	484
portArbiter.h	485
processor.cc	486
processor.h	487
ptop_swa.cc	488
ptop_swa.h	489
pvtopy_swa.cc	490
pvtopy_swa.h	491
rank_handler.cc	492
rank_handler.h	493
refresh_manager.cc	494
refresh_manager.h	495
request.cc	496
request.h	497
request_handler.cc	499
request_handler.h	500
response_handler.cc	501
response_handler.h	502
response_handler_temp.cc	503
response_handler_temp.h	504
router.cc	505
router.h	506
routerFourStageVcs.cc	507
routerFourStageVcs.h	508
simMc2Mesh.cc	509
simulator.cc	512
simulator.h	513
stats.cc	514
stats.h	515
util.cc	518
util.h	519
virtualChannelArbiter.h	520

Chapter 6

Directory Documentation

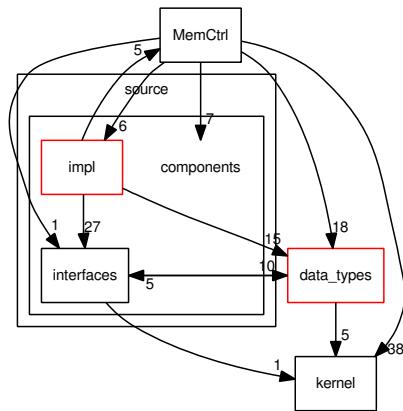
6.1 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_-prealpha/source/components/impl/backup/ Directory Reference



Files

- file **genericPortArbiter.cc**
- file **genericPortArbiter.h**
- file **backup/genericRouterNoVcs.cc**
- file **backup/genericRouterNoVcs.h**
- file **backup/genericRPG.cc**
- file **backup/genericRPG.h**
- file **mcFrontEnd.cc**
- file **mcFrontEnd.h**
- file **routerFourStageVcs.cc**
- file **routerFourStageVcs.h**

6.2 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/components/ Directory Reference

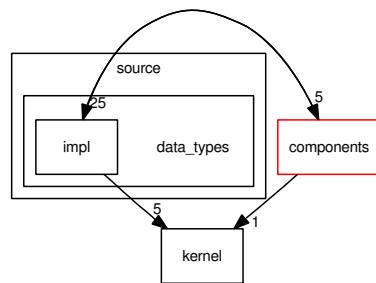


Directories

- directory **impl**
- directory **interfaces**

6.3 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/data_types/ Directory

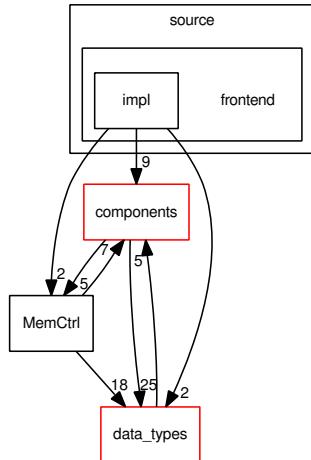
Reference
6.3 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/data_types/ Directory Reference¹⁹



Directories

- directory **impl**

6.4 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/frontend/ Directory Reference

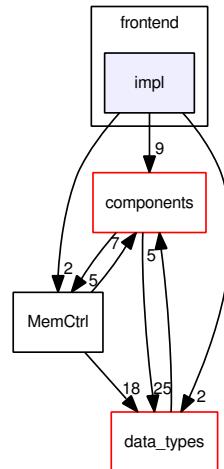


Directories

- directory **impl**

6.5 /home/mitch/workspace/MIRCO/k1230504_srcs/iris_-
prealpha/source/frontend/impl/ Directory

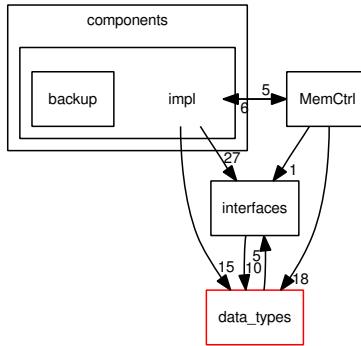
Reference
6.5 /home/mitch/workspace/MIRCO/k1230504_srcs/iris_-
prealpha/source/frontend/impl/ Directory Reference ²¹



Files

- file **mesh.cc**
- file **mesh.h**
- file **simMc2Mesh.cc**

6.6 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/components/impl/ Directory Reference



Directories

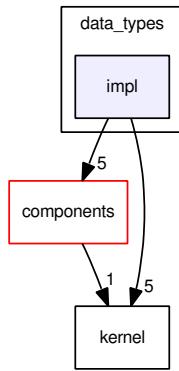
- directory **backup**

Files

- file **genericArbiter.cc**
- file **genericArbiter.h**
- file **genericBuffer.cc**
- file **genericBuffer.h**
- file **genericCrossbar.cc**
- file **genericCrossbar.h**
- file **genericData.cc**
- file **genericData.h**
- file **genericEvents.h**
- file **genericFlatMc.cc**
- file **genericFlatMc.h**
- file **genericInterface.cc**
- file **genericInterface.h**
- file **genericLink.cc**
- file **genericLink.h**
- file **genericRC.cc**
- file **genericRC.h**
- file **genericRouterAdaptive.cc**
- file **genericRouterAdaptive.h**
- file **genericRouterNoVcs.cc**
- file **genericRouterNoVcs.h**
- file **genericRouterVct.cc**
- file **genericRouterVct.h**
- file **genericRPG.cc**
- file **genericRPG.h**

-
- file **genericSink.cc**
 - file **genericSink.h**
 - file **genericTPG.cc**
 - file **genericTPG.h**
 - file **genericTPG_temp.cc**
 - file **genericVcArbiter.cc**
 - file **genericVcArbiter.h**
 - file **genericVcsInterface.cc**
 - file **genericVcsInterface.h**
 - file **ptop_swa.cc**
 - file **ptop_swa.h**
 - file **pvttopv_swa.cc**
 - file **pvttopv_swa.h**

6.7 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/data_types/impl/ Directory Reference

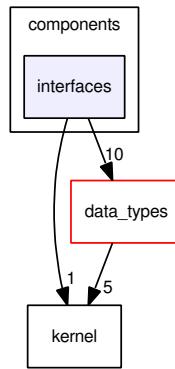


Files

- file `flit.cc`
- file `flit.h`
- file `highLevelPacket.cc`
- file `highLevelPacket.h`
- file `irisEvent.cc`
- file `irisEvent.h`
- file `lowLevelPacket.cc`
- file `lowLevelPacket.h`
- file `util.cc`
- file `util.h`

6.8 /home/mitch/workspace/MIRCO/k1230504_srcs/iris_-
prealpha/source/components/interfaces/ Directory

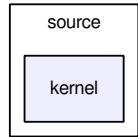
Reference
6.8 /home/mitch/workspace/MIRCO/k1230504_srcs/iris_-
prealpha/source/components/interfaces/ Directory Reference ²⁵



Files

- file **addressDecoder.h**
- file **arbiter.h**
- file **buffer.cc**
- file **buffer.h**
- file **crossbar.h**
- file **genericComponentHeader.h**
- file **inputBuffer.h**
- file **interface.cc**
- file **interface.h**
- file **irisLink.cc**
- file **irisLink.h**
- file **networkComponent.cc**
- file **networkComponent.h**
- file **outputBuffer.h**
- file **portArbiter.h**
- file **processor.cc**
- file **processor.h**
- file **router.cc**
- file **router.h**
- file **virtualChannelArbiter.h**

6.9 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_- prealpha/source/kernel/ Directory Reference



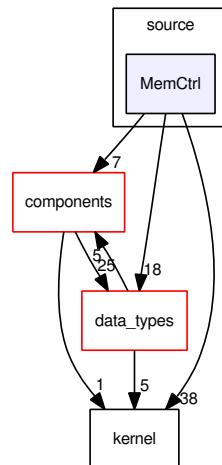
Files

- file **component.cc**
- file **component.h**
- file **link.cc**
- file **link.h**
- file **simulator.cc**
- file **simulator.h**

6.10

/home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/MemCtrl/
Directory Reference 27

6.10 /home/mitch/workspace/MIRCO/kl230504_- srcs/iris_prealpha/source/MemCtrl/ Directory Refer- ence

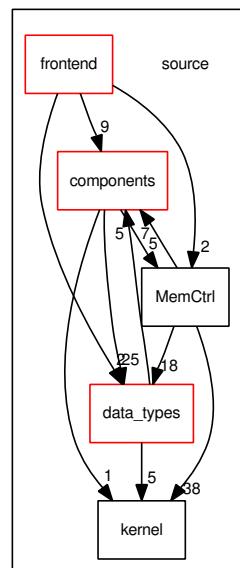


Files

- file **addr_map.cc**
- file **addr_map.h**
- file **bank_handler.cc**
- file **bank_handler.h**
- file **bus.cc**
- file **bus.h**
- file **bus_handler.cc**
- file **bus_handler.h**
- file **channel_handler.cc**
- file **channel_handler.h**
- file **cmd_bus_handler.cc**
- file **cmd_bus_handler.h**
- file **cmd_issuer.cc**
- file **cmd_issuer.h**
- file **constants.h**
- file **data_bus_handler.cc**
- file **data_bus_handler.h**
- file **dram.cc**
- file **dram.h**
- file **main.cc**
- file **MC.cc**
- file **MC.h**
- file **mshr.cc**
- file **mshr.h**
- file **NI.cc**
- file **NI.h**

- file **rank_handler.cc**
- file **rank_handler.h**
- file **refresh_manager.cc**
- file **refresh_manager.h**
- file **request.cc**
- file **request.h**
- file **request_handler.cc**
- file **request_handler.h**
- file **response_handler.cc**
- file **response_handler.h**
- file **response_handler_temp.cc**
- file **response_handler_temp.h**
- file **stats.cc**
- file **stats.h**

6.11 /home/mitch/workspace/MIRCO/kl230504_srcs/iris_prealpha/source/ Directory Reference



Directories

- directory **components**
- directory **data_types**
- directory **frontend**
- directory **kernel**
- directory **MemCtrl**

Chapter 7

Class Documentation

7.1 AddressDecoder Class Reference

```
#include <addressDecoder.h>
```

Public Member Functions

- **AddressDecoder ()**
- virtual void **push** (**Flit** *f, unsigned int vc)=0
- virtual **Flit** * **pull** (unsigned int vc)=0
- virtual unsigned int **get_putput_port** (unsigned int channel)=0
- virtual unsigned int **get_vc** (unsigned int channel)=0
- virtual void **no_channels** (unsigned int channels)=0
- virtual unsigned int **set_no_channels** ()=0
- virtual bool **is_channel_empty** (unsigned int channel)=0
- virtual bool **is_empty** ()=0
- virtual unsigned int **speculate_port** (**Flit** *f, unsigned int port)=0
- virtual unsigned int **speculate_channel** (**Flit** *f, unsigned int channel)=0

7.1.1 Detailed Description

Definition at line 31 of file addressDecoder.h.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 `AddressDecoder::AddressDecoder ()`

7.1.3 Member Function Documentation

7.1.3.1 `virtual unsigned int AddressDecoder::get_putput_port (unsigned int channel) [pure virtual]`

7.1.3.2 `virtual unsigned int AddressDecoder::get_vc (unsigned int channel) [pure virtual]`

7.1.3.3 `virtual bool AddressDecoder::is_channel_empty (unsigned int channel) [pure virtual]`

7.1.3.4 `virtual bool AddressDecoder::is_empty () [pure virtual]`

7.1.3.5 `virtual void AddressDecoder::no_channels (unsigned int channels) [pure virtual]`

7.1.3.6 `virtual Flit* AddressDecoder::pull (unsigned int vc) [pure virtual]`

7.1.3.7 `virtual void AddressDecoder::push (Flit * f, unsigned int vc) [pure virtual]`

7.1.3.8 `virtual unsigned int AddressDecoder::set_no_channels () [pure virtual]`

7.1.3.9 `virtual unsigned int AddressDecoder::speculate_channel (Flit * f, unsigned int channel) [pure virtual]`

7.1.3.10 `virtual unsigned int AddressDecoder::speculate_port (Flit * f, unsigned int port) [pure virtual]`

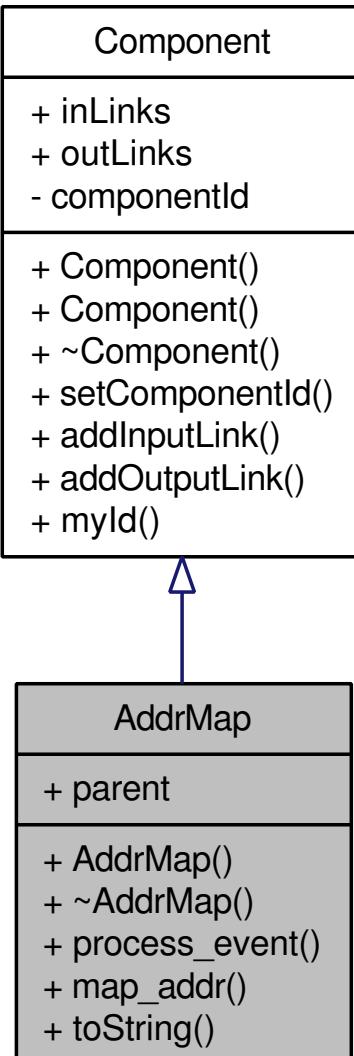
The documentation for this class was generated from the following file:

- `addressDecoder.h`

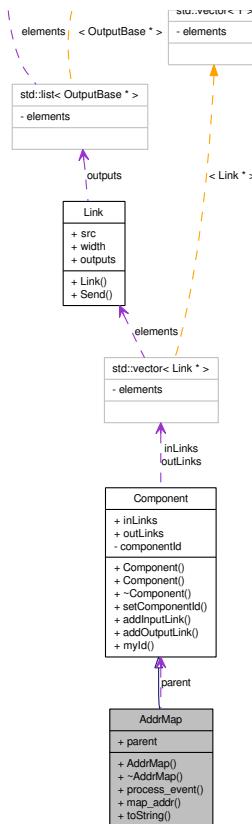
7.2 AddrMap Class Reference

```
#include <addr_map.h>
```

Inheritance diagram for AddrMap:



Collaboration diagram for AddrMap:



Public Member Functions

- **AddrMap ()**
- **~AddrMap ()**
- **void process_event (IrisEvent *e)**
- **void map_addr (Request *req)**
- **std::string toString ()**

Public Attributes

- **Component * parent**

7.2.1 Detailed Description

Definition at line 45 of file addr_map.h.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 AddrMap::AddrMap ()

Definition at line 34 of file addr_map.cc.

7.2.2.2 AddrMap::~AddrMap ()

Definition at line 46 of file addr_map.cc.

7.2.3 Member Function Documentation

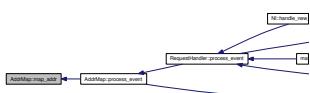
7.2.3.1 void AddrMap::map_addr (Request * *req*)

Definition at line 118 of file addr_map.cc.

References Request::address, BANK_BITS, Request::bankNo, BLOCKS_PER_ROW, CACHE_BLOCK_SIZE, Request::channelNo, COLUMN_SIZE, Request::columnNo, DRAM_SIZE, Request::lowerBits, NO_OF_BANKS, NO_OF_CHANNELS, NO_OF_COLUMNS, NO_OF_RANKS, NO_OF_ROWS, Request::rankNo, ROW_SIZE, Request::rowNo, and TAG_BITS.

Referenced by process_event().

Here is the caller graph for this function:



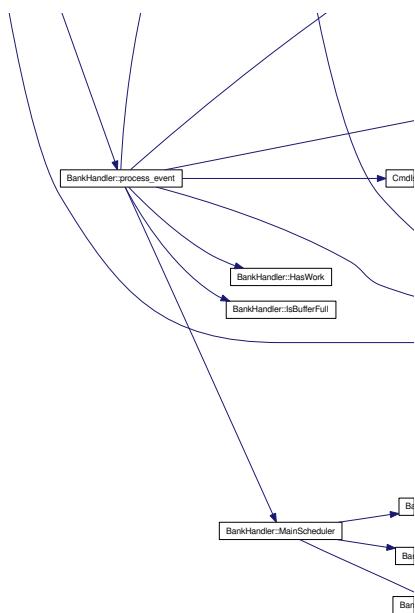
7.2.3.2 void AddrMap::process_event (IrisEvent * *e*)

Definition at line 58 of file addr_map.cc.

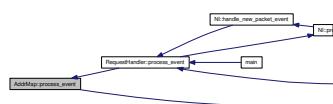
References Request::address, Request::bankNo, Request::channelNo, Request::columnNo, map_addr(), Simulator::Now(), parent, BankHandler::process_event(), Request::rankNo, Request::rbufferInsertionTime, Request::rowNo, Simulator::Schedule(), and START.

Referenced by RequestHandler::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.2.3.3 std::string AddrMap::toString (void)

Definition at line 113 of file `addr_map.cc`.

7.2.4 Member Data Documentation

7.2.4.1 Component* AddrMap::parent

Definition at line 50 of file addr_map.h.

Referenced by process_event(), and RequestHandler::RequestHandler().

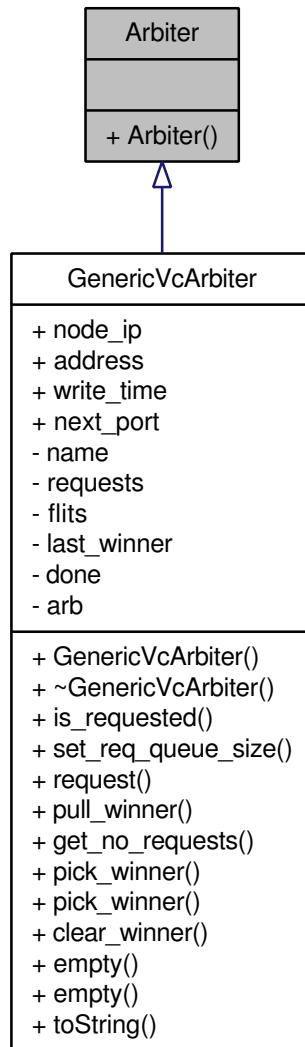
The documentation for this class was generated from the following files:

- [addr_map.h](#)
- [addr_map.cc](#)

7.3 Arbiter Class Reference

```
#include <arbiter.h>
```

Inheritance diagram for Arbiter:



Public Member Functions

- **Arbiter ()**

7.3.1 Detailed Description

Definition at line 32 of file arbiter.h.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Arbiter::Arbiter () [inline]

Definition at line 35 of file arbiter.h.

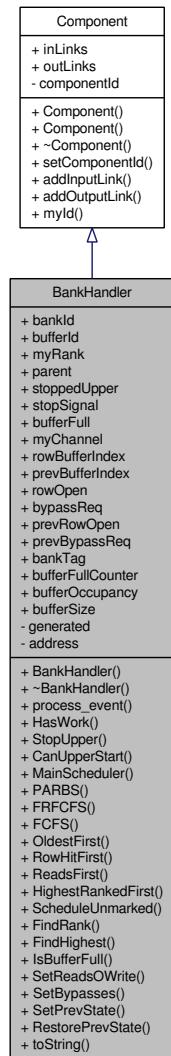
The documentation for this class was generated from the following file:

- **arbiter.h**

7.4 BankHandler Class Reference

```
#include <bank_handler.h>
```

Inheritance diagram for BankHandler:



Collaboration diagram for BankHandler:



Public Member Functions

- **BankHandler ()**
- **~BankHandler ()**
- void **process _ event (IrisEvent *e)**
- bool **HasWork ()**
- bool **StopUpper ()**
- bool **CanUpperStart ()**
- bool **MainScheduler (Request *req, int *index)**
- bool **PARBS (Request *req, int *index)**
- bool **FRFCFS (Request *req, int *index)**
- bool **FCFS (Request *req, int *index)**
- bool **OldestFirst (Request *req, int *index)**
- bool **RowHitFirst (Request *req, int *index)**
- bool **ReadsFirst (Request *req, int *index)**

- bool **HighestRankedFirst** (Request *req, UInt highest, int *index)
- bool **ScheduleUnmarked** (Request *req, int *index)
- void **FindRank** (int priority[])
- UInt **FindHighest** ()
- bool **IsBufferFull** ()
- void **SetReadsOWrite** (CallerType caller, Command_t cmdType)
- void **SetBypasses** (unsigned int index)
- void **SetPrevState** ()
- void **RestorePrevState** ()
- std::string **toString** ()

Public Attributes

- UInt **bankId**
- UInt **bufferId**
- void * **myRank**
- Component * **parent**
- bool **stoppedUpper**
- bool **stopSignal**
- bool **bufferFull**
- UInt **myChannel**
- UInt **rowBufferIndex**
- UInt **prevBufferIndex**
- bool **rowOpen**
- unsigned int **bypassReq**
- bool **prevRowOpen**
- unsigned int **prevBypassReq**
- unsigned int **bankTag**
- unsigned long long int **bufferFullCounter**
- unsigned long long int **bufferOccupancy**
- unsigned long long int **bufferSize**

Private Attributes

- bool **generated**
- uint **address**

7.4.1 Detailed Description

Definition at line 48 of file bank_handler.h.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 BankHandler::BankHandler ()

Definition at line 30 of file bank_handler.cc.

References address, bufferFull, bufferFullCounter, bufferOccupancy, bufferSize, bypassReq, generated, prevBufferIndex, prevBypassReq, prevRowOpen, rowBufferIndex, rowOpen, stoppedUpper, and stopSignal.

7.4.2.2 BankHandler::~BankHandler ()

Definition at line 57 of file bank_handler.cc.

7.4.3 Member Function Documentation

7.4.3.1 bool BankHandler::CanUpperStart ()

Definition at line 258 of file bank_handler.cc.

References bankId, MAX_BUFFER_SIZE, and myRank.

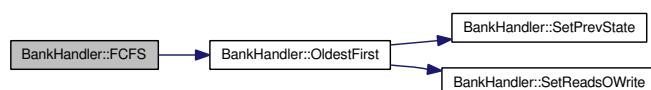
7.4.3.2 bool BankHandler::FCFS (Request * *req*, int * *index*)

Definition at line 344 of file bank_handler.cc.

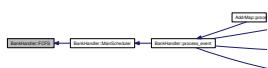
References bufferId, myRank, OldestFirst(), and RankHandler::rbuffer.

Referenced by MainScheduler().

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.3 UInt BankHandler::FindHighest ()

Definition at line 536 of file bank_handler.cc.

References MAX_BUFFER_SIZE, myRank, NO_OF_BUFFERS, NO_OF_THREADS, and RankHandler::rbuffer.

Referenced by PARBS().

Here is the caller graph for this function:



7.4.3.4 void BankHandler::FindRank (int *priority*[])

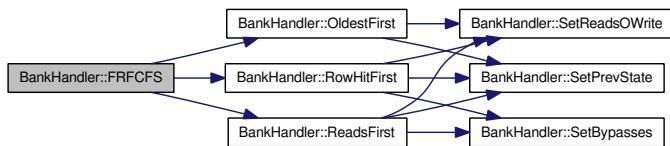
7.4.3.5 bool BankHandler::FRFCFS (Request * *req*, int * *index*)

Definition at line 326 of file bank_handler.cc.

References bufferId, myRank, OldestFirst(), RankHandler::rbuffer, ReadsFirst(), and RowHitFirst().

Referenced by MainScheduler().

Here is the call graph for this function:



Here is the caller graph for this function:



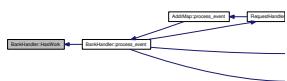
7.4.3.6 bool BankHandler::HasWork ()

Definition at line 245 of file bank_handler.cc.

References bankId, and myRank.

Referenced by process_event().

Here is the caller graph for this function:



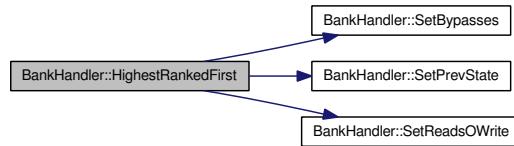
7.4.3.7 bool BankHandler::HighestRankedFirst (Request * *req*, UInt *highest*, int * *index*)

Definition at line 501 of file bank_handler.cc.

References bankId, Request::bankNo, bufferId, CLOSED, Request::cmdType, CONFLICT, HIGHEST_RANKED_FIRST, myRank, OPEN, RankHandler::rbuffer, rowBufferIndex, Request::rowNo, rowOpen, SetBypasses(), SetPrevState(), SetReadsOWrite(), Request::status, and Request::threadId.

Referenced by PARBS().

Here is the call graph for this function:



Here is the caller graph for this function:



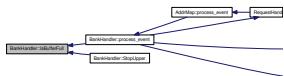
7.4.3.8 bool BankHandler::IsBufferFull ()

Definition at line 266 of file bank_handler.cc.

References bufferFull, bufferId, MAX_BUFFER_SIZE, and myRank.

Referenced by process_event(), and StopUpper().

Here is the caller graph for this function:



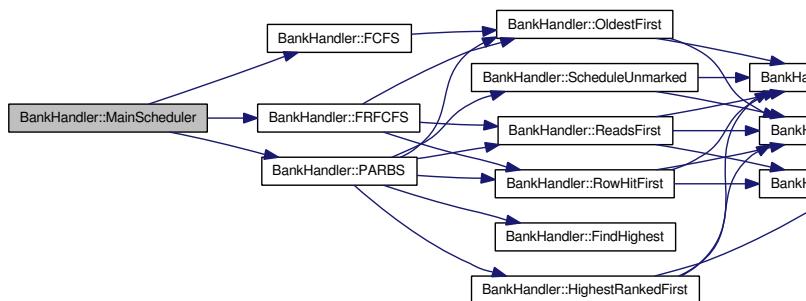
7.4.3.9 bool BankHandler::MainScheduler (Request * req, int * index)

Definition at line 282 of file bank_handler.cc.

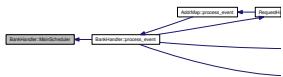
References FCFS(), FRFCFS(), and PARBS().

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



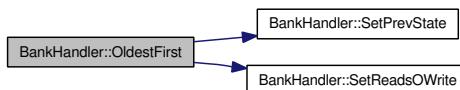
7.4.3.10 bool BankHandler::OldestFirst (Request * *req*, int * *index*)

Definition at line 361 of file bank_handler.cc.

References bankId, Request::bankNo, bufferId, CLOSED, Request::cmdType, CONFLICT, myRank, OLDEST_FIRST, OPEN, RankHandler::rbuffer, rowBufferIndex, Request::rowNo, rowOpen, SetPrevState(), SetReadsOWrite(), and Request::status.

Referenced by FCFS(), FRFCFS(), and PARBS().

Here is the call graph for this function:



Here is the caller graph for this function:



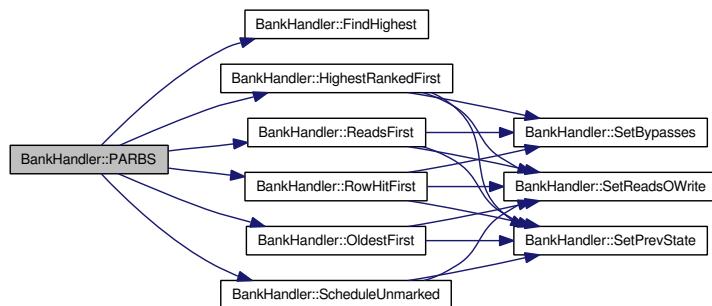
7.4.3.11 bool BankHandler::PARBS (Request * *req*, int * *index*)

Definition at line 301 of file bank_handler.cc.

References bufferId, FindHighest(), HighestRankedFirst(), myRank, OldestFirst(), RankHandler::rbuffer, ReadsFirst(), RowHitFirst(), and ScheduleUnmarked().

Referenced by MainScheduler().

Here is the call graph for this function:



Here is the caller graph for this function:



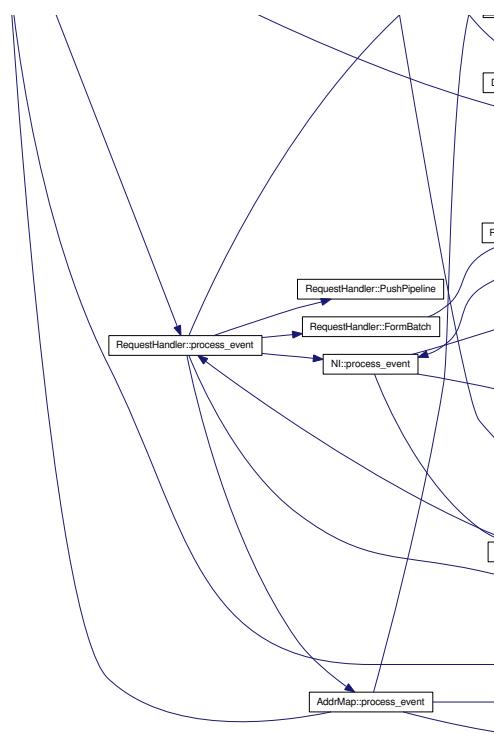
7.4.3.12 void BankHandler::process_event (IrisEvent * e)

Definition at line 69 of file bank_handler.cc.

References Request::address, bankId, Request::bankNo, bufferFull, bufferFullCounter, bufferOccupancy, bufferSize, CACHE_PREFETCH, CACHE_READ, CACHE_WRITE, Request::cbufferInsertionTime, Request::channelNo, Request::cmdType, CONTINUE, IrisEvent::dst, IrisEvent::event_data, generated, HasWork(), IsBufferFull(), MainScheduler(), myChannel, myRank, Simulator::Now(), parent, RequestHandler::process_event(), ResponseHandler::process_event(), CmdIssuer::process_event(), PUSH_BUFFER, RESPONSE_BUFFER_SIZE, Request::rowNo, Simulator::Schedule(), IrisEvent::src, START, STOP, stopSignal, Request::tag, and IrisEvent::type.

Referenced by AddrMap::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



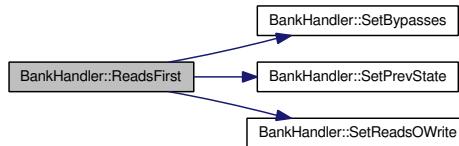
7.4.3.13 bool BankHandler::ReadsFirst (Request * *req*, int * *index*)

Definition at line 434 of file bank_handler.cc.

References bankId, Request::bankNo, bufferId, CACHE_PREFETCH, CACHE_READ, CACHE_WRITE, CLOSED, Request::cmdType, CONFLICT, MAX_READ_OV_WRITE, myRank, OPEN, RankHandler::rbuffer, READS_FIRST, RankHandler::readsOWrite, rowBufferIndex, Request::rowNo, rowOpen, SetBypasses(), SetPrevState(), SetReadsOWrite(), and Request::status.

Referenced by FRFCFS(), and PARBS().

Here is the call graph for this function:



Here is the caller graph for this function:



7.4.3.14 void BankHandler::RestorePrevState ()

Definition at line 644 of file bank_handler.cc.

References bypassReq, myRank, prevBufferIndex, prevBypassReq, RankHandler::prevReadsOWrite, prevRowOpen, RankHandler::readsOWrite, rowBufferIndex, and rowOpen.

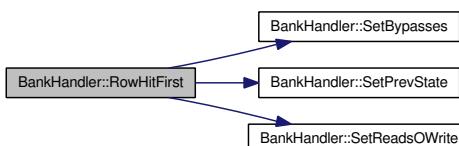
7.4.3.15 bool BankHandler::RowHitFirst (Request * *req*, int * *index*)

Definition at line 399 of file bank_handler.cc.

References bankId, Request::bankNo, bufferId, CLOSED, Request::cmdType, CONFLICT, HIT_FIRST, myRank, OPEN, RankHandler::rbuffer, rowBufferIndex, Request::rowNo, rowOpen, SetBypasses(), SetPrevState(), SetReadsOWrite(), and Request::status.

Referenced by FRFCFS(), and PARBS().

Here is the call graph for this function:



Here is the caller graph for this function:



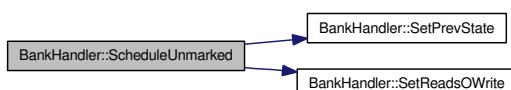
7.4.3.16 bool BankHandler::ScheduleUnmarked (Request * *req*, int * *index*)

Definition at line 470 of file bank_handler.cc.

References bankId, Request::bankNo, bufferId, CLOSED, Request::cmdType, CONFLICT, myRank, OPEN, RankHandler::rbuffer, rowBufferIndex, Request::rowNo, rowOpen, SetPrevState(), SetReadsOWrite(), Request::status, and UNMARKED_FIRST.

Referenced by PARBS().

Here is the call graph for this function:



Here is the caller graph for this function:



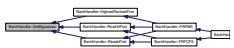
7.4.3.17 void BankHandler::SetBypasses (unsigned int *index*)

Definition at line 616 of file bank_handler.cc.

References bankId, bufferId, bypassReq, myRank, and RankHandler::rbuffer.

Referenced by HighestRankedFirst(), ReadsFirst(), and RowHitFirst().

Here is the caller graph for this function:



7.4.3.18 void BankHandler::SetPrevState ()

Definition at line 635 of file bank_handler.cc.

References bypassReq, myRank, prevBufferIndex, prevBypassReq, RankHandler::prevReadsOWrite, prevRowOpen, RankHandler::readsOWrite, rowBufferIndex, and rowOpen.

Referenced by HighestRankedFirst(), OldestFirst(), ReadsFirst(), RowHitFirst(), and ScheduleUnmarked().

Here is the caller graph for this function:



7.4.3.19 void BankHandler::SetReadsOWrite (CallerType *caller*, Command_t *cmdType*)

Definition at line 586 of file bank_handler.cc.

References CACHE_PREFETCH, CACHE_READ, CACHE_WRITE, HIGHEST_RANKED_FIRST, HIT_FIRST, myRank, OLDEST_FIRST, RankHandler::readsOWrite, and UNMARKED_FIRST.

Referenced by HighestRankedFirst(), OldestFirst(), ReadsFirst(), RowHitFirst(), and ScheduleUnmarked().

Here is the caller graph for this function:



7.4.3.20 bool BankHandler::StopUpper ()

Definition at line 253 of file bank_handler.cc.

References IsBufferFull().

Here is the call graph for this function:



7.4.3.21 std::string BankHandler::toString (void)

Definition at line 277 of file bank_handler.cc.

7.4.4 Member Data Documentation

7.4.4.1 uint BankHandler::address [private]

Definition at line 100 of file bank_handler.h.

Referenced by BankHandler().

7.4.4.2 **UIInt** BankHandler::bankId

Definition at line 53 of file bank_handler.h.

Referenced by CanUpperStart(), HasWork(), HighestRankedFirst(), OldestFirst(), process_event(), ReadsFirst(), RequestHandler::RequestHandler(), RowHitFirst(), ScheduleUnmarked(), and SetBypasses().

7.4.4.3 **unsigned int** BankHandler::bankTag

Definition at line 88 of file bank_handler.h.

7.4.4.4 **bool** BankHandler::bufferFull

Definition at line 59 of file bank_handler.h.

Referenced by BankHandler(), IsBufferFull(), and process_event().

7.4.4.5 **unsigned long long int** BankHandler::bufferFullCounter

Definition at line 92 of file bank_handler.h.

Referenced by BankHandler(), and process_event().

7.4.4.6 **UIInt** BankHandler::bufferId

Definition at line 54 of file bank_handler.h.

Referenced by FCFS(), FRFCFS(), HighestRankedFirst(), IsBufferFull(), OldestFirst(), PARBS(), ReadsFirst(), RequestHandler::RequestHandler(), RowHitFirst(), ScheduleUnmarked(), and SetBypasses().

7.4.4.7 **unsigned long long int** BankHandler::bufferOccupancy

Definition at line 93 of file bank_handler.h.

Referenced by BankHandler(), and process_event().

7.4.4.8 **unsigned long long int** BankHandler::bufferSize

Definition at line 94 of file bank_handler.h.

Referenced by BankHandler(), and process_event().

7.4.4.9 **unsigned int** BankHandler::bypassReq

Definition at line 85 of file bank_handler.h.

Referenced by BankHandler(), RestorePrevState(), SetBypasses(), and SetPrevState().

7.4.4.10 bool BankHandler::generated [private]

Definition at line 99 of file bank_handler.h.

Referenced by BankHandler(), and process_event().

7.4.4.11 UInt BankHandler::myChannel

Definition at line 81 of file bank_handler.h.

Referenced by process_event(), and RequestHandler::RequestHandler().

7.4.4.12 void* BankHandler::myRank

Definition at line 55 of file bank_handler.h.

Referenced by CanUpperStart(), FCFS(), FindHighest(), FRFCFS(), HasWork(), HighestRankedFirst(), IsBufferFull(), OldestFirst(), PARBS(), process_event(), ReadsFirst(), RequestHandler::RequestHandler(), RestorePrevState(), RowHitFirst(), ScheduleUnmarked(), SetBypasses(), SetPrevState(), and SetReadsOWrite().

7.4.4.13 Component* BankHandler::parent

Definition at line 56 of file bank_handler.h.

Referenced by process_event(), and RequestHandler::RequestHandler().

7.4.4.14 UInt BankHandler::prevBufferIndex

Definition at line 83 of file bank_handler.h.

Referenced by BankHandler(), RestorePrevState(), and SetPrevState().

7.4.4.15 unsigned int BankHandler::prevBypassReq

Definition at line 87 of file bank_handler.h.

Referenced by BankHandler(), RestorePrevState(), and SetPrevState().

7.4.4.16 bool BankHandler::prevRowOpen

Definition at line 86 of file bank_handler.h.

Referenced by BankHandler(), RestorePrevState(), and SetPrevState().

7.4.4.17 UInt BankHandler::rowBufferIndex

Definition at line 82 of file bank_handler.h.

Referenced by BankHandler(), HighestRankedFirst(), OldestFirst(), ReadsFirst(), RestorePrevState(), RowHitFirst(), ScheduleUnmarked(), and SetPrevState().

7.4.4.18 bool BankHandler::rowOpen

Definition at line 84 of file bank_handler.h.

Referenced by BankHandler(), HighestRankedFirst(), OldestFirst(), ReadsFirst(), RestorePrevState(), RowHitFirst(), ScheduleUnmarked(), and SetPrevState().

7.4.4.19 bool BankHandler::stoppedUpper

Definition at line 57 of file bank_handler.h.

Referenced by BankHandler().

7.4.4.20 bool BankHandler::stopSignal

Definition at line 58 of file bank_handler.h.

Referenced by BankHandler(), and process_event().

The documentation for this class was generated from the following files:

- [bank_handler.h](#)
- [bank_handler.cc](#)

7.5 BankState Struct Reference

```
#include <cmd_issuer.h>
```

Public Member Functions

- **BankState ()**

Public Attributes

- unsigned long long int **prevRow**
- DRAMCmd **prevCmd**
- Time **prevCmdTime**

7.5.1 Detailed Description

Definition at line 76 of file cmd_issuer.h.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 BankState::BankState () [inline]

Definition at line 81 of file cmd_issuer.h.

References PRECHARGE, prevCmd, prevCmdTime, and prevRow.

7.5.3 Member Data Documentation

7.5.3.1 DRAMCmd BankState::prevCmd

Definition at line 79 of file cmd_issuer.h.

Referenced by CmdIssuer::BankNotBusy(), BankState(), and CmdIssuer::SetPrevState().

7.5.3.2 Time BankState::prevCmdTime

Definition at line 80 of file cmd_issuer.h.

Referenced by CmdIssuer::BankNotBusy(), BankState(), and CmdIssuer::SetPrevState().

7.5.3.3 unsigned long long int BankState::prevRow

Definition at line 78 of file cmd_issuer.h.

Referenced by BankState(), and CmdIssuer::SetPrevState().

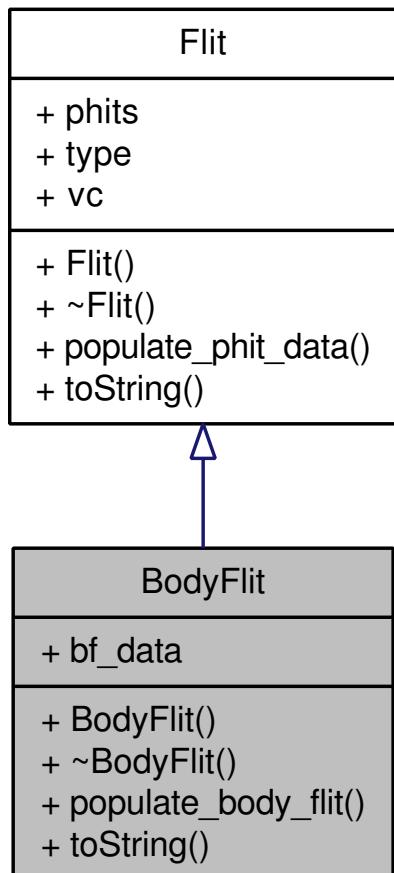
The documentation for this struct was generated from the following file:

- **cmd_issuer.h**

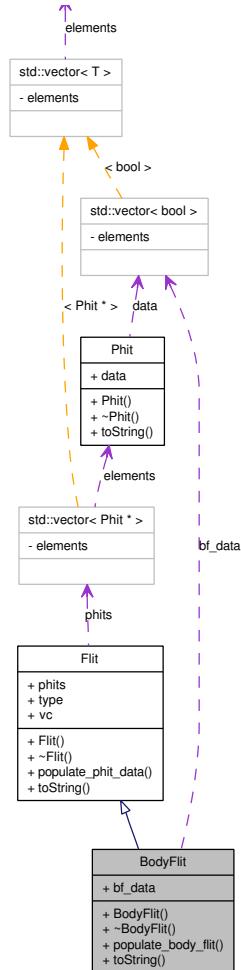
7.6 BodyFlit Class Reference

```
#include <flit.h>
```

Inheritance diagram for BodyFlit:



Collaboration diagram for BodyFlit:



Public Member Functions

- **BodyFlit ()**
- **~BodyFlit ()**
- **void populate_body_flit ()**
- **std::string toString () const**

Public Attributes

- **vector< bool > bf_data**

7.6.1 Detailed Description

Definition at line 126 of file flit.h.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 BodyFlit::BodyFlit ()

Definition at line 163 of file flit.cc.

References BODY, and Flit::type.

7.6.2.2 BodyFlit::~BodyFlit ()

Definition at line 168 of file flit.cc.

References Flit::phits.

7.6.3 Member Function Documentation

7.6.3.1 void BodyFlit::populate_body_flit ()

Definition at line 196 of file flit.cc.

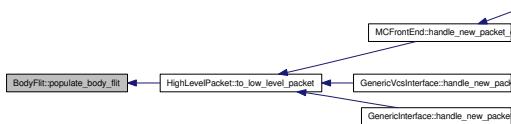
References bf_data, and Flit::populate_phit_data().

Referenced by HighLevelPacket::to_low_level_packet().

Here is the call graph for this function:



Here is the caller graph for this function:



7.6.3.2 string BodyFlit::toString () const

Reimplemented from **Flit** (p. 110).

Definition at line 182 of file flit.cc.

References Flit::phits, and Flit::type.

7.6.4 Member Data Documentation

7.6.4.1 `vector<bool> BodyFlit::bf_data`

Definition at line 132 of file flit.h.

Referenced by `LowLevelPacket::add()`, `populate_body_flit()`, and `HighLevelPacket::to_low_level_packet()`.

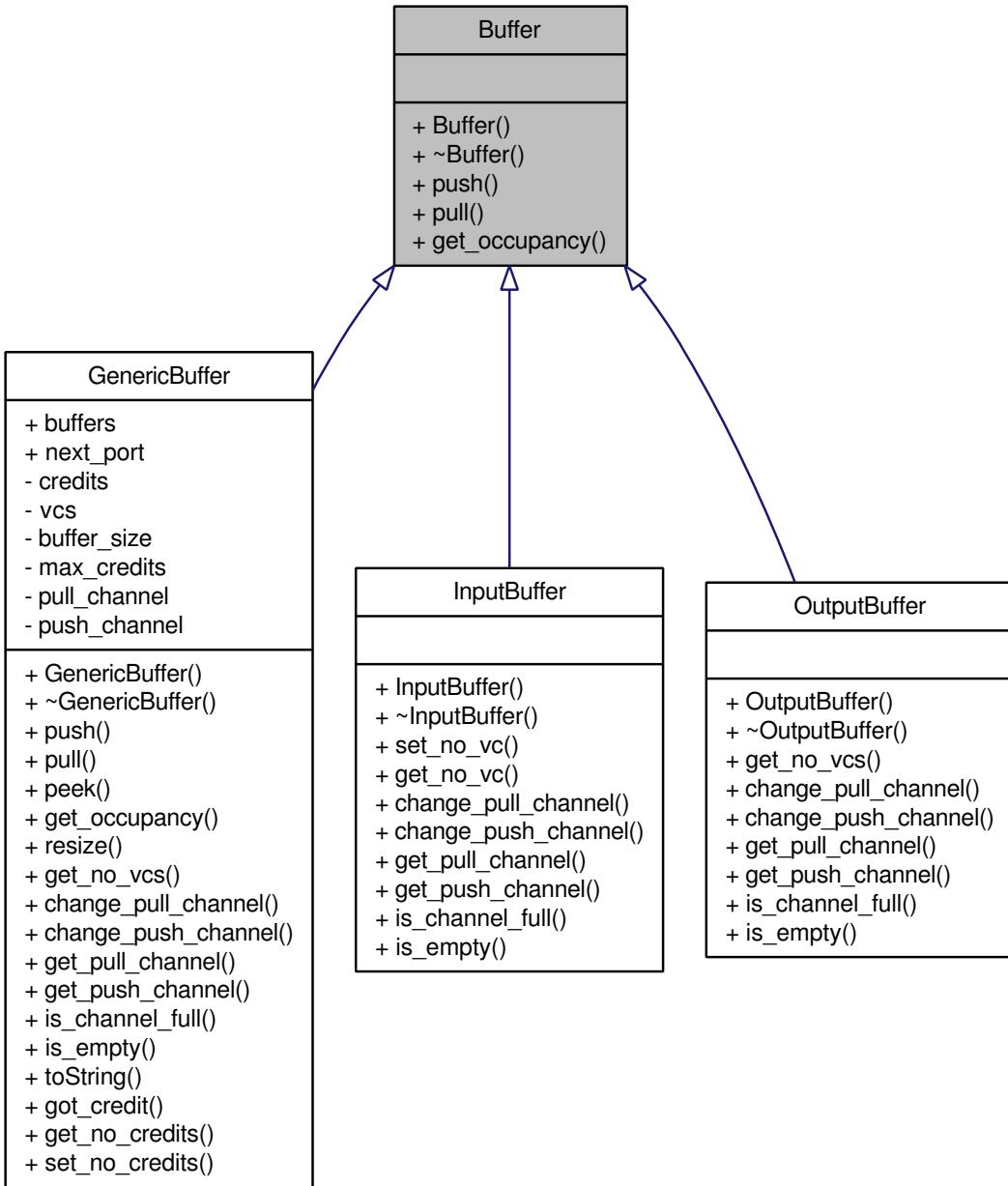
The documentation for this class was generated from the following files:

- `flit.h`
- `flit.cc`

7.7 Buffer Class Reference

```
#include <buffer.h>
```

Inheritance diagram for Buffer:



Public Member Functions

- **Buffer ()**
- **~Buffer ()**
- **virtual void push (Flit *f)=0**
- **virtual Flit * pull ()=0**

- virtual **uint get_occupancy (uint channel) const =0**

7.7.1 Detailed Description

Definition at line 34 of file buffer.h.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 **Buffer::Buffer () [inline]**

Definition at line 37 of file buffer.h.

7.7.2.2 **Buffer::~Buffer () [inline]**

Definition at line 38 of file buffer.h.

7.7.3 Member Function Documentation

7.7.3.1 **virtual uint Buffer::get_occupancy (uint channel) const [pure virtual]**

Implemented in **GenericBuffer** (p. 117).

7.7.3.2 **virtual Flit* Buffer::pull () [pure virtual]**

Implemented in **GenericBuffer** (p. 118).

7.7.3.3 **virtual void Buffer::push (Flit * f) [pure virtual]**

Implemented in **GenericBuffer** (p. 118).

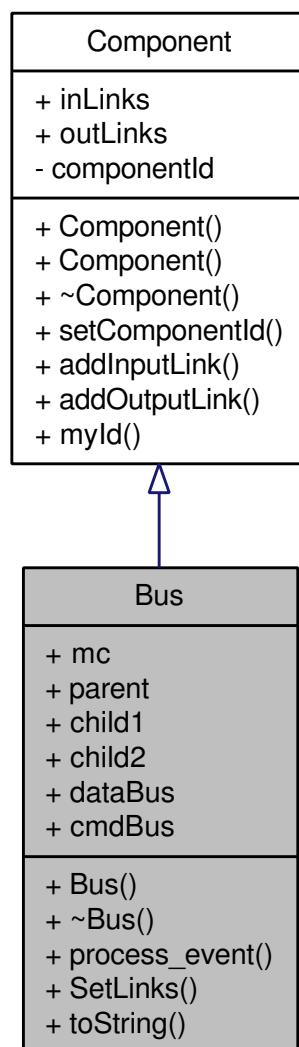
The documentation for this class was generated from the following file:

- **buffer.h**

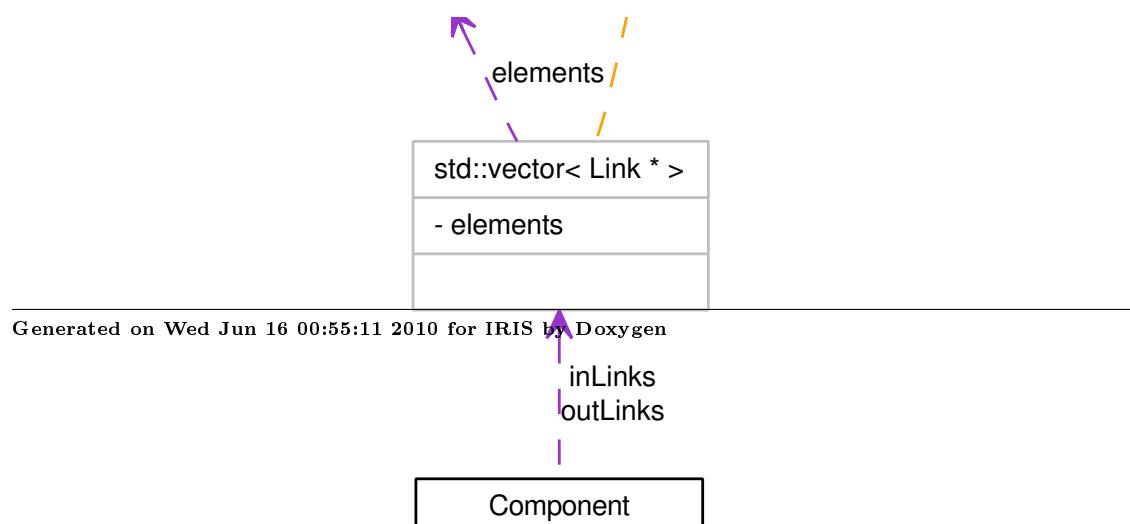
7.8 Bus Class Reference

```
#include <bus.h>
```

Inheritance diagram for Bus:



Collaboration diagram for Bus:



Public Member Functions

- **Bus ()**
- **~Bus ()**
- void **process_event (IrisEvent *e)**
- void **SetLinks ()**
- std::string **toString ()**

Public Attributes

- Component * **mc**
- Component * **parent**
- Component * **child1**
- Component * **child2**
- DataBusHandler **dataBus [NO_OF_CHANNELS]**
- CmdBusHandler **cmdBus [NO_OF_CHANNELS]**

7.8.1 Detailed Description

Definition at line 46 of file bus.h.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Bus::Bus ()

Definition at line 32 of file bus.cc.

7.8.2.2 Bus::~Bus ()

Definition at line 44 of file bus.cc.

7.8.3 Member Function Documentation

7.8.3.1 void Bus::process_event (IrisEvent * e)

Definition at line 56 of file bus.cc.

7.8.3.2 void Bus::SetLinks ()

Definition at line 61 of file bus.cc.

References CmdBusHandler::child, child1, DataBusHandler::child1, child2, DataBusHandler::child2, cmdBus, dataBus, NO_OF_CHANNELS, CmdBusHandler::parent, and DataBusHandler::parent.

Referenced by MC::Init().

Here is the caller graph for this function:



7.8.3.3 std::string Bus::toString (void)

Definition at line 72 of file bus.cc.

7.8.4 Member Data Documentation**7.8.4.1 Component* Bus::child1**

Definition at line 53 of file bus.h.

Referenced by MC::Init(), and SetLinks().

7.8.4.2 Component* Bus::child2

Definition at line 54 of file bus.h.

Referenced by MC::Init(), and SetLinks().

7.8.4.3 CmdBusHandler Bus::cmdBus[NO_OF_CHANNELS]

Definition at line 56 of file bus.h.

Referenced by SetLinks().

7.8.4.4 DataBusHandler Bus::dataBus[NO_OF_CHANNELS]

Definition at line 55 of file bus.h.

Referenced by SetLinks().

7.8.4.5 Component* Bus::mc

Definition at line 51 of file bus.h.

Referenced by MC::Init().

7.8.4.6 Component* Bus::parent

Definition at line 52 of file bus.h.

Referenced by MC::Init().

The documentation for this class was generated from the following files:

- **bus.h**
- **bus.cc**

7.9 BusHandler Class Reference

```
#include <bus_handler.h>
```

Collaboration diagram for BusHandler:

Public Member Functions

- **BusHandler ()**
- **~BusHandler ()**
- **bool BankRankFree (unsigned int bankNo, unsigned int rankNo, unsigned int channelNo)**
- **void SetIfFull ()**
- **bool IsFull (int i)**
- **void LowLevelCmdGen (Request *req)**
- **std::string toString ()**

Public Attributes

- **Component * parent**
- **CmdQueue cmdQueue [NO_OF_CHANNELS]**
- **CmdIssuer cmdIssuer [NO_OF_CHANNELS]**
- **bool stopSignal**
- **bool full [NO_OF_CHANNELS]**
- **bool linkBusy [NO_OF_CHANNELS]**
- **bool oneReq [NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]**

7.9.1 Detailed Description

Definition at line 46 of file bus_handler.h.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 BusHandler::BusHandler ()

Definition at line 34 of file bus_handler.cc.

References CmdIssuer::bufferId, cmdIssuer, full, CmdIssuer::Id, linkBusy, NO_OF_BANKS, NO_OF_CHANNELS, NO_OF_RANKS, oneReq, CmdIssuer::parent, and stopSignal.

7.9.2.2 BusHandler::~BusHandler ()

Definition at line 61 of file bus_handler.cc.

7.9.3 Member Function Documentation

7.9.3.1 bool BusHandler::BankRankFree (unsigned int *bankNo*, unsigned int *rankNo*, unsigned int *channelNo*)

Definition at line 85 of file bus_handler.cc.

References oneReq.

7.9.3.2 bool BusHandler::IsFull (int *i*)

Definition at line 81 of file bus_handler.cc.

References full.

7.9.3.3 void BusHandler::LowLevelCmdGen (Request * *req*)

Definition at line 90 of file bus_handler.cc.

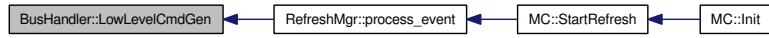
References ACTIVATE, ALL_BANK_REFRESH, CACHE_PREFETCH, CACHE_READ, CACHE_WRITE, CACHE_WRITEBACK, Request::channelNo, CLOSED, cmdQueue, Request::cmdType, CONFLICT, IDLE, NO_OF_CHANNELS, NORMAL, OPEN, PRECHARGE, PREFETCHL, READ, READL, REFRESH, DRAMCmdState::set(), Request::status, WRITE, WRITEBACKL, and WRITEL.

Referenced by RefreshMgr::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.9.3.4 void BusHandler::SetIfFull ()

Definition at line 71 of file bus_handler.cc.

References cmdQueue, full, MAX_CMD_BUFFER_SIZE, NO_OF_CHANNELS, and stopSignal.

7.9.3.5 std::string BusHandler::toString (void)

Definition at line 66 of file bus_handler.cc.

7.9.4 Member Data Documentation

7.9.4.1 CmdIssuer BusHandler::cmdIssuer[NO_OF_CHANNELS]

Definition at line 54 of file bus_handler.h.

Referenced by BusHandler(), RefreshMgr::process_event(), and RequestHandler::SetLinks().

7.9.4.2 CmdQueue BusHandler::cmdQueue[NO_OF_CHANNELS]

Definition at line 53 of file bus_handler.h.

Referenced by CmdIssuer::BanksFirstReq(), CmdIssuer::IssueCmd(), LowLevelCmdGen(), and SetIfFull().

7.9.4.3 bool BusHandler::full[NO_OF_CHANNELS]

Definition at line 56 of file bus_handler.h.

Referenced by BusHandler(), IsFull(), RequestHandler::process_event(), and SetIfFull().

7.9.4.4 bool BusHandler::linkBusy[NO_OF_CHANNELS]

Definition at line 57 of file bus_handler.h.

Referenced by BusHandler().

7.9.4.5 bool BusHandler::oneReq[NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]

Definition at line 58 of file bus_handler.h.

Referenced by BankRankFree(), and BusHandler().

7.9.4.6 Component* BusHandler::parent

Definition at line 51 of file bus_handler.h.

Referenced by RequestHandler::RequestHandler().

7.9.4.7 bool BusHandler::stopSignal

Definition at line 55 of file bus_handler.h.

Referenced by BusHandler(), RequestHandler::process_event(), and SetIfFull().

The documentation for this class was generated from the following files:

- **bus_handler.h**
- **bus_handler.cc**

7.10 ChannelHandler Class Reference

```
#include <channel_handler.h>
```

Collaboration diagram for ChannelHandler:

Public Member Functions

- `ChannelHandler ()`
- `~ChannelHandler ()`

Public Attributes

- `short chanId`
- `RankHandler rank [NO_OF_RANKS]`

7.10.1 Detailed Description

Definition at line 44 of file channel_handler.h.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 `ChannelHandler::ChannelHandler ()`

Definition at line 32 of file channel_handler.cc.

7.10.2.2 `ChannelHandler::~ChannelHandler ()`

Definition at line 44 of file channel_handler.cc.

7.10.3 Member Data Documentation

7.10.3.1 `short ChannelHandler::chanId`

Definition at line 49 of file channel_handler.h.

Referenced by RequestHandler::RequestHandler().

7.10.3.2 `RankHandler ChannelHandler::rank[NO_OF_RANKS]`

Definition at line 50 of file channel_handler.h.

Referenced by RequestHandler::MarkAll(), RequestHandler::MarkBatchOnly(), and RequestHandler::RequestHandler().

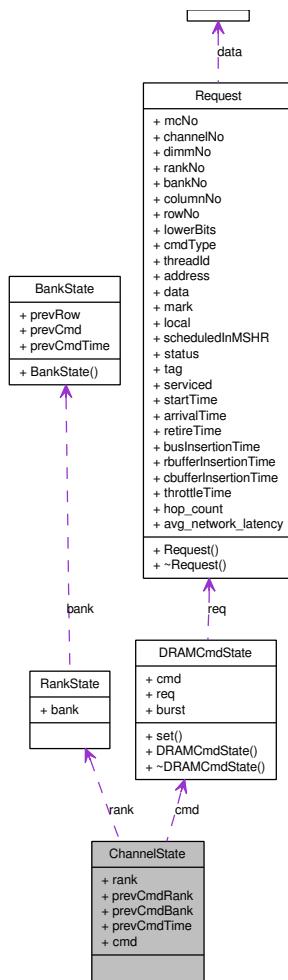
The documentation for this class was generated from the following files:

- `channel_handler.h`
- `channel_handler.cc`

7.11 ChannelState Struct Reference

```
#include <cmd_issuer.h>
```

Collaboration diagram for ChannelState:



Public Attributes

- **RankState rank [NO_OF_RANKS]**
- **unsigned int prevCmdRank**
- **unsigned int prevCmdBank**
- **Time prevCmdTime**
- **DRAMCmdState cmd**

7.11.1 Detailed Description

Definition at line 96 of file cmd_issuer.h.

7.11.2 Member Data Documentation

7.11.2.1 DRAMCmdState ChannelState::cmd

Definition at line 102 of file cmd_issuer.h.

Referenced by CmdIssuer::BankNotBusy(), CmdIssuer::CmdIssuer(), CmdIssuer::IssueCmd(), and CmdIssuer::SetPrevState().

7.11.2.2 unsigned int ChannelState::prevCmdBank

Definition at line 100 of file cmd_issuer.h.

Referenced by CmdIssuer::CmdIssuer(), and CmdIssuer::SetPrevState().

7.11.2.3 unsigned int ChannelState::prevCmdRank

Definition at line 99 of file cmd_issuer.h.

Referenced by CmdIssuer::CmdIssuer(), and CmdIssuer::SetPrevState().

7.11.2.4 Time ChannelState::prevCmdTime

Definition at line 101 of file cmd_issuer.h.

Referenced by CmdIssuer::CanSchedule(), CmdIssuer::CmdIssuer(), and CmdIssuer::SetPrevState().

7.11.2.5 RankState ChannelState::rank[NO_OF_RANKS]

Definition at line 98 of file cmd_issuer.h.

Referenced by CmdIssuer::BankNotBusy(), and CmdIssuer::SetPrevState().

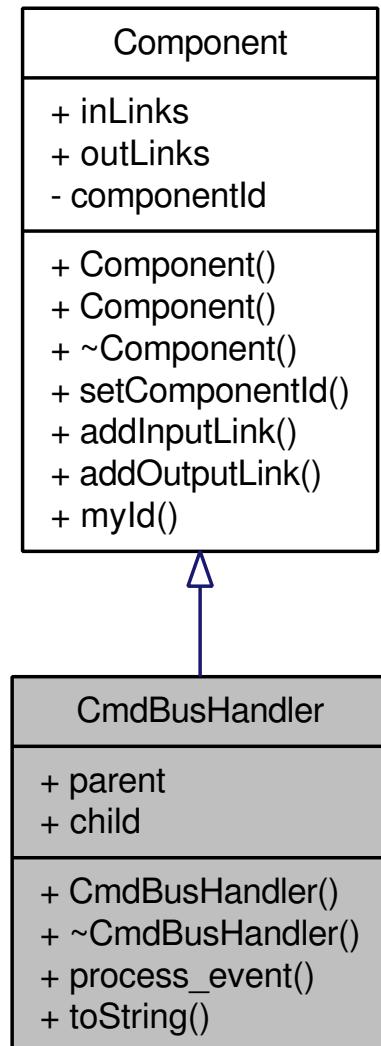
The documentation for this struct was generated from the following file:

- cmd_issuer.h

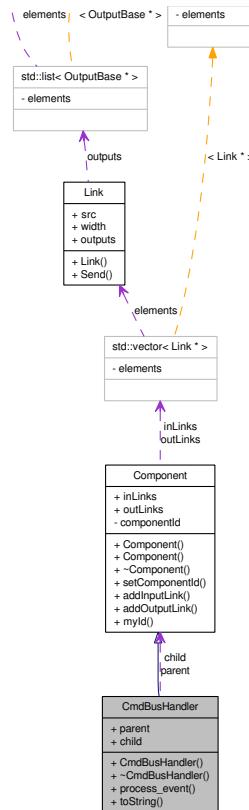
7.12 CmdBusHandler Class Reference

```
#include <cmd_bus_handler.h>
```

Inheritance diagram for CmdBusHandler:



Collaboration diagram for CmdBusHandler:



Public Member Functions

- **CmdBusHandler ()**
- **~CmdBusHandler ()**
- **void process_event (IrisEvent *e)**
- **std::string toString ()**

Public Attributes

- **Component * parent**
- **Component * child**

7.12.1 Detailed Description

Definition at line 44 of file cmd_bus_handler.h.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 CmdBusHandler::CmdBusHandler ()

Definition at line 34 of file cmd_bus_handler.cc.

7.12.2.2 CmdBusHandler::~CmdBusHandler ()

Definition at line 46 of file cmd_bus_handler.cc.

7.12.3 Member Function Documentation

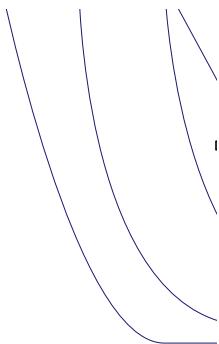
7.12.3.1 void CmdBusHandler::process_event (IrisEvent * e)

Definition at line 58 of file cmd_bus_handler.cc.

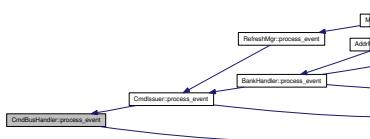
References child, IrisEvent::event_data, Simulator::Now(), DRAMChannel::process_event(), Simulator::Schedule(), START_READ, and t_CMD.

Referenced by CmdIssuer::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.12.3.2 std::string CmdBusHandler::toString ()**7.12.4 Member Data Documentation****7.12.4.1 Component* CmdBusHandler::child**

Definition at line 50 of file cmd_bus_handler.h.

Referenced by process_event(), and Bus::SetLinks().

7.12.4.2 Component* CmdBusHandler::parent

Definition at line 49 of file cmd_bus_handler.h.

Referenced by Bus::SetLinks().

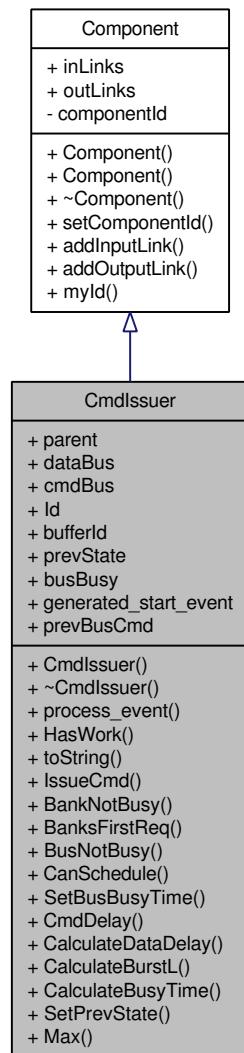
The documentation for this class was generated from the following files:

- [cmd_bus_handler.h](#)
- [cmd_bus_handler.cc](#)

7.13 CmdIssuer Class Reference

```
#include <cmd_issuer.h>
```

Inheritance diagram for CmdIssuer:



Collaboration diagram for CmdIssuer:

Public Member Functions

- **CmdIssuer ()**
- **~CmdIssuer ()**
- void **process_event** (IrisEvent *e)
- bool **HasWork ()**
- std::string **toString ()**
- bool **IssueCmd** (bool *isWrite, DRAMCmdState *scheduledCmd)
- bool **BankNotBusy** (DRAMCmdState currentCmd)
- bool **BanksFirstReq** (DRAMCmdState currentCmd, unsigned int index)
- bool **BusNotBusy** (DRAMCmdState currentCmd)
- bool **CanSchedule** (DRAMCmdState currentCmd, DRAMCmdState prevCmd)
- void **SetBusBusyTime** (DRAMCmdState currentCmd)
- Time **CmdDelay** (DRAMCmd cmd)
- Time **CalculateDataDelay** (DRAMCmdState cmdS)
- Time **CalculateBurstL** (DRAMCmdState prevCmd)
- Time **CalculateBusyTime** (DRAMCmd curCmd, DRAMCmd prevCmd, RankBankComb rankC, RankBankComb bankC, Time prevBurstL)
- void **SetPrevState** (DRAMCmdState currentCmd)
- Time **Max** (Time t1, Time t2)

Public Attributes

- Component * parent
- DataBusHandler * dataBus
- CmdBusHandler * cmdBus
- short Id
- short bufferId
- ChannelState prevState
- Time busBusy
- bool generated_start_event
- DRAMCmd prevBusCmd

7.13.1 Detailed Description

Definition at line 106 of file cmd_issuer.h.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 CmdIssuer::CmdIssuer ()

Definition at line 28 of file cmd_issuer.cc.

References busBusy, ChannelState::cmd, generated_start_event, NO_OF_BANKS, NO_OF_RANKS, NORMAL, PRECHARGE, ChannelState::prevCmdBank, ChannelState::prevCmdRank, ChannelState::prevCmdTime, prevState, and DRAMCmdState::set().

Here is the call graph for this function:



7.13.2.2 CmdIssuer::~CmdIssuer ()

Definition at line 47 of file cmd_issuer.cc.

7.13.3 Member Function Documentation

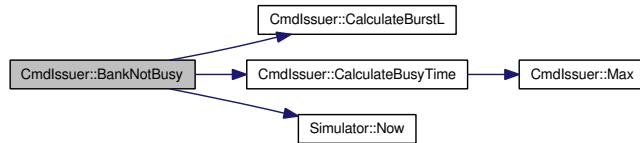
7.13.3.1 bool CmdIssuer::BankNotBusy (DRAMCmdState *currentCmd*)

Definition at line 236 of file cmd_issuer.cc.

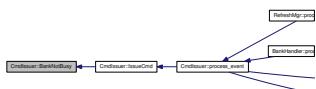
References ALL_BANK_REFRESH, RankState::bank, Request::bankNo, CalculateBurstL(), CalculateBusyTime(), ChannelState::cmd, DRAMCmdState::cmd, NO_OF_BANKS, Simulator::Now(), BankState::prevCmd, BankState::prevCmdTime, prevState, ChannelState::rank, Request::rankNo, DRAMCmdState::req, and SAME.

Referenced by IssueCmd().

Here is the call graph for this function:



Here is the caller graph for this function:



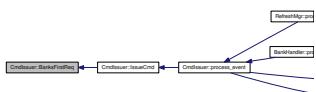
7.13.3.2 bool CmdIssuer::BanksFirstReq (DRAMCmdState *currentCmd*, unsigned int *index*)

Definition at line 225 of file cmd_issuer.cc.

References Request::bankNo, bufferId, BusHandler::cmdQueue, parent, and DRAMCmdState::req.

Referenced by IssueCmd().

Here is the caller graph for this function:



7.13.3.3 bool CmdIssuer::BusNotBusy (DRAMCmdState *currentCmd*)

Definition at line 189 of file cmd_issuer.cc.

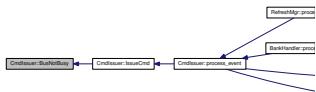
References busBusy, DRAMCmdState::cmd, Simulator::Now(), prevBusCmd, READ, t_CMD, and WRITE.

Referenced by IssueCmd().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.4 Time CmdIssuer::CalculateBurstL (DRAMCmdState *prevCmd*)

Definition at line 326 of file cmd_issuer.cc.

References DRAMCmdState::burst, BUS_CYCLE, DDR_BUS_WIDTH, PREFETCH_SIZE, PREFETCHL, READ_SIZE, READL, WRITE_SIZE, WRITEBACK_SIZE, WRITEBACKL, and WRITEL.

Referenced by BankNotBusy(), CanSchedule(), and SetBusBusyTime().

Here is the caller graph for this function:



7.13.3.5 Time CmdIssuer::CalculateBusyTime (DRAMCmd *curCmd*, DRAMCmd *prevCmd*, RankBankComb *rankC*, RankBankComb *bankC*, Time *prevBurstL*)

Definition at line 349 of file cmd_issuer.cc.

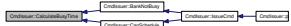
References ACTIVATE, ALL_BANK_REFRESH, DIFF, Max(), PRECHARGE, READ, SAME, t_AL, t_CAS, t_CCD, t_CMD, t_CWD, t_OST, t_RAS, t_RCD, t_RFC, t_RP, t_RRD, t_RTP, t_RTRS, t_WR, t_WTR, and WRITE.

Referenced by BankNotBusy(), and CanSchedule().

Here is the call graph for this function:



Here is the caller graph for this function:



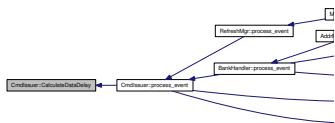
7.13.3.6 Time CmdIssuer::CalculateDataDelay (DRAMCmdState cmdS)

Definition at line 289 of file cmd_issuer.cc.

References DRAMCmdState::cmd, t_CMD, t_CWD, and WRITE.

Referenced by process_event().

Here is the caller graph for this function:



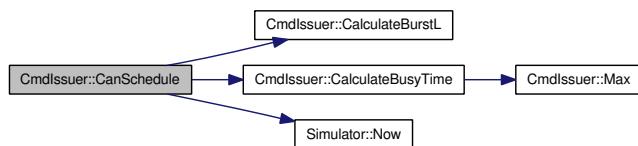
7.13.3.7 bool CmdIssuer::CanSchedule (DRAMCmdState currentCmd, DRAMCmdState prevCmd)

Definition at line 301 of file cmd_issuer.cc.

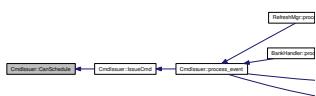
References Request::bankNo, CalculateBurstL(), CalculateBusyTime(), DRAMCmdState::cmd, DIFF, Simulator::Now(), ChannelState::prevCmdTime, prevState, Request::rankNo, DRAMCmdState::req, and SAME.

Referenced by IssueCmd().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.8 Time CmdIssuer::CmdDelay (DRAMCmd cmd)

Definition at line 273 of file cmd_issuer.cc.

References ACTIVATE, ALL_BANK_REFRESH, PRECHARGE, READ, t_CAS, t_RCD, t_RFC, t_RP, and WRITE.

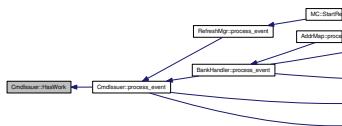
7.13.3.9 bool CmdIssuer::HasWork ()

Definition at line 141 of file cmd_issuer.cc.

References bufferId, and parent.

Referenced by process_event().

Here is the caller graph for this function:



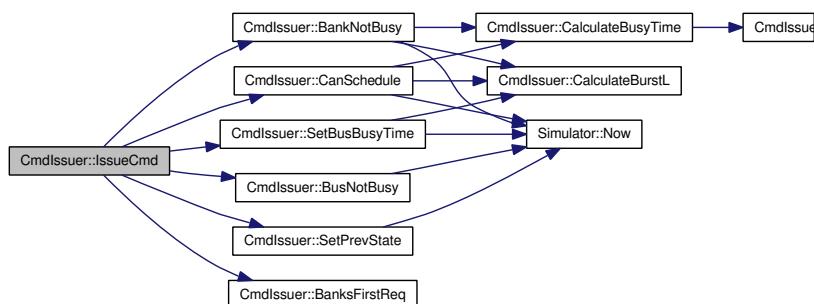
7.13.3.10 bool CmdIssuer::IssueCmd (bool * isWrite, DRAMCmdState * scheduledCmd)

Definition at line 151 of file cmd_issuer.cc.

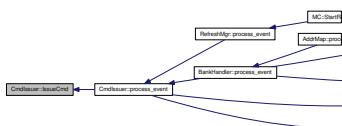
References BankNotBusy(), BanksFirstReq(), bufferId, BusNotBusy(), CanSchedule(), ChannelState::cmd, BusHandler::cmdQueue, parent, prevState, SetBusBusyTime(), SetPrevState(), and WRITE.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.11 Time CmdIssuer::Max (Time *t1*, Time *t2*)

Definition at line 720 of file cmd_issuer.cc.

Referenced by CalculateBusyTime().

Here is the caller graph for this function:



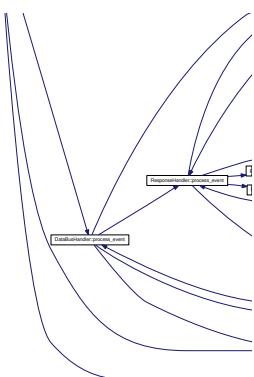
7.13.3.12 void CmdIssuer::process_event (IrisEvent * *e*)

Definition at line 59 of file cmd_issuer.cc.

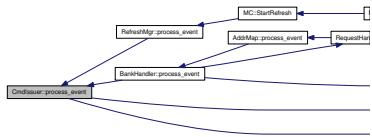
References Request::address, Request::bankNo, Request::busInsertionTime, CalculateDataDelay(), Request::channelNo, DRAMCmdState::cmd, cmdBus, dataBus, IrisEvent::dst, IrisEvent::event_data, generated_start_event, HasWork(), Id, IssueCmd(), Simulator::Now(), parent, PRECHARGE, DataBusHandler::process_event(), CmdBusHandler::process_event(), Request::rankNo, READ, DRAMCmdState::req, Simulator::Schedule(), IrisEvent::src, START, START_WRITE, IrisEvent::type, and WRITE.

Referenced by RefreshMgr::process_event(), and BankHandler::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



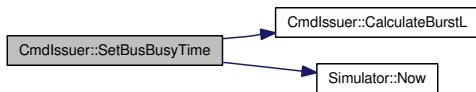
7.13.3.13 void CmdIssuer::SetBusBusyTime (DRAMCmdState *currentCmd*)

Definition at line 212 of file cmd_issuer.cc.

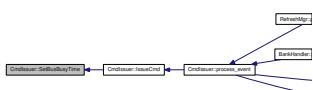
References busBusy, CalculateBurstL(), DRAMCmdState::cmd, Simulator::Now(), prevBusCmd, READ, and WRITE.

Referenced by IssueCmd().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.14 void CmdIssuer::SetPrevState (DRAMCmdState *currentCmd*)

Definition at line 708 of file cmd_issuer.cc.

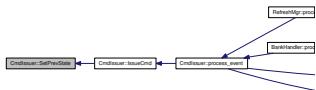
References RankState::bank, Request::bankNo, DRAMCmdState::cmd, ChannelState::cmd, Simulator::Now(), BankState::prevCmd, ChannelState::prevCmdBank, ChannelState::prevCmdRank, BankState::prevCmdTime, ChannelState::prevCmdTime, BankState::prevRow, prevState, ChannelState::rank, Request::rankNo, DRAMCmdState::req, and Request::rowNo.

Referenced by IssueCmd().

Here is the call graph for this function:



Here is the caller graph for this function:



7.13.3.15 std::string CmdIssuer::toString (void)

Definition at line 146 of file cmd_issuer.cc.

7.13.4 Member Data Documentation

7.13.4.1 short CmdIssuer::bufferId

Definition at line 115 of file cmd_issuer.h.

Referenced by BanksFirstReq(), BusHandler::BusHandler(), HasWork(), and IssueCmd().

7.13.4.2 Time CmdIssuer::busBusy

Definition at line 117 of file cmd_issuer.h.

Referenced by BusNotBusy(), CmdIssuer(), and SetBusBusyTime().

7.13.4.3 CmdBusHandler* CmdIssuer::cmdBus

Definition at line 113 of file cmd_issuer.h.

Referenced by process_event(), and RequestHandler::SetLinks().

7.13.4.4 DataBusHandler* CmdIssuer::dataBus

Definition at line 112 of file cmd_issuer.h.

Referenced by process_event(), and RequestHandler::SetLinks().

7.13.4.5 bool CmdIssuer::generated_start_event

Definition at line 134 of file cmd_issuer.h.

Referenced by CmdIssuer(), and process_event().

7.13.4.6 short CmdIssuer::Id

Definition at line 114 of file cmd_issuer.h.

Referenced by BusHandler::BusHandler(), and process_event().

7.13.4.7 Component* CmdIssuer::parent

Definition at line 111 of file cmd_issuer.h.

Referenced by BanksFirstReq(), BusHandler::BusHandler(), HasWork(), IssueCmd(), and process_event().

7.13.4.8 DRAMCmd CmdIssuer::prevBusCmd

Definition at line 135 of file cmd_issuer.h.

Referenced by BusNotBusy(), and SetBusBusyTime().

7.13.4.9 ChannelState CmdIssuer::prevState

Definition at line 116 of file cmd_issuer.h.

Referenced by BankNotBusy(), CanSchedule(), CmdIssuer(), IssueCmd(), and SetPrevState().

The documentation for this class was generated from the following files:

- [cmd_issuer.h](#)
- [cmd_issuer.cc](#)

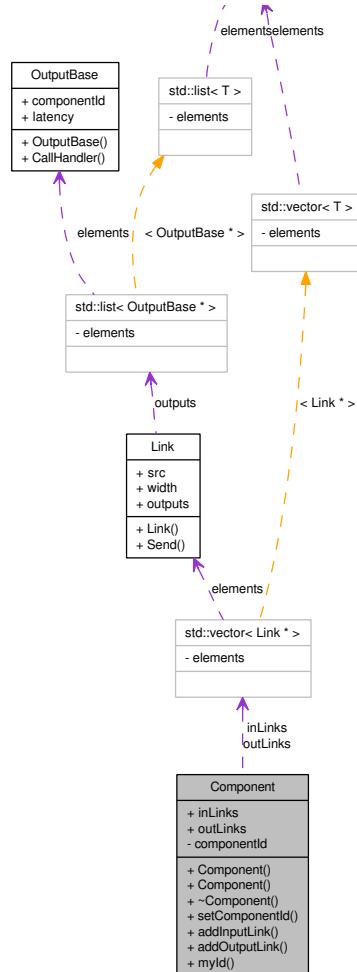
7.14 Component Class Reference

```
#include <component.h>
```

Inheritance diagram for Component:



Collaboration diagram for Component:



Public Member Functions

- **Component ()**
- **Component (int lpId)**
- **~Component ()**
- **void setComponentId (int id)**
- **void addInputLink (Link *)**
- **void addOutputLink (Link *)**
- **int myId ()**

Public Attributes

- **std::vector< Link * > inLinks**
- **std::vector< Link * > outLinks**

Private Attributes

- int **componentId**

7.14.1 Detailed Description

Definition at line 11 of file component.h.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Component::Component ()

Definition at line 10 of file component.cc.

References Simulator::registerComponent().

Here is the call graph for this function:



7.14.2.2 Component::Component (int *lpId*)

Definition at line 15 of file component.cc.

References Simulator::registerComponent().

Here is the call graph for this function:



7.14.2.3 Component::~Component ()

Definition at line 5 of file component.cc.

7.14.3 Member Function Documentation

7.14.3.1 void Component::addInputLink (Link * *l*)

Definition at line 25 of file component.cc.

References inLinks.

7.14.3.2 void Component::addOutputLink (Link * *l*)

Definition at line 30 of file component.cc.

References outLinks.

Referenced by Link::Link().

Here is the caller graph for this function:



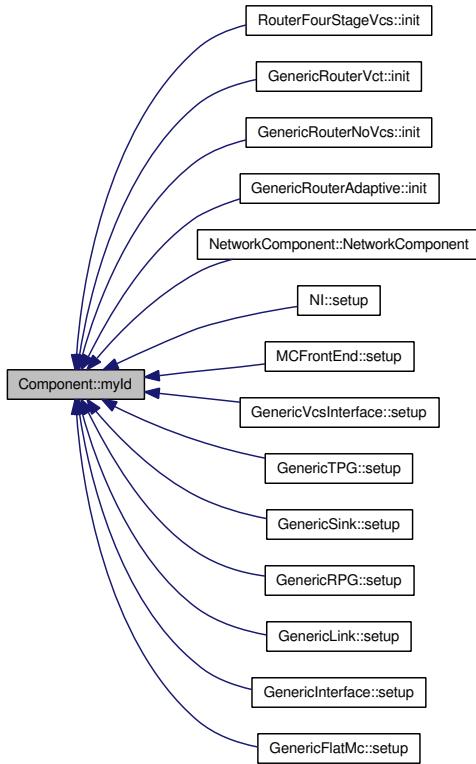
7.14.3.3 int Component::myId () [inline]

Definition at line 24 of file component.h.

References componentId.

Referenced by RouterFourStageVcs::init(), GenericRouterVct::init(), GenericRouterNoVcs::init(), GenericRouterAdaptive::init(), NetworkComponent::NetworkComponent(), NI::setup(), MCFrontEnd::setup(), GenericVcsInterface::setup(), GenericTPG::setup(), GenericSink::setup(), GenericRPG::setup(), GenericLink::setup(), GenericInterface::setup(), and GenericFlatMc::setup().

Here is the caller graph for this function:



7.14.3.4 void Component::setComponentId (int *id*)

Definition at line 20 of file component.cc.

References componentId.

Referenced by Simulator::registerComponent().

Here is the caller graph for this function:



7.14.4 Member Data Documentation

7.14.4.1 int Component::componentId [private]

Definition at line 14 of file component.h.

Referenced by myId(), and setComponentId().

7.14.4.2 std::vector<Link*> Component::inLinks

Definition at line 16 of file component.h.

Referenced by addInputLink().

7.14.4.3 std::vector<Link*> Component::outLinks

Definition at line 17 of file component.h.

Referenced by addOutputLink().

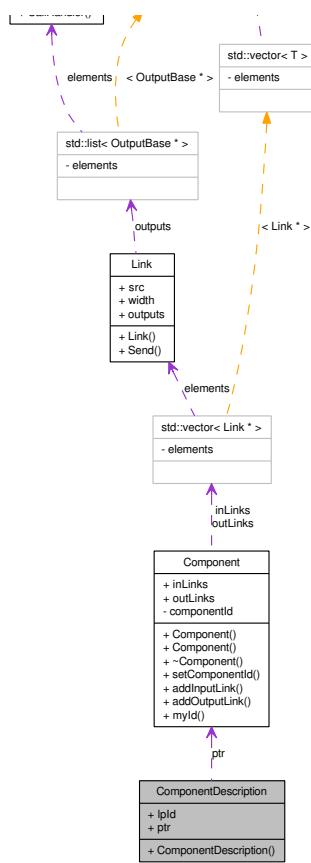
The documentation for this class was generated from the following files:

- **component.h**
- **component.cc**

7.15 ComponentDescription Class Reference

```
#include <simulator.h>
```

Collaboration diagram for ComponentDescription:



Public Member Functions

- **ComponentDescription** (int lp, Component *obj)

Public Attributes

- int lpId
- Component * ptr

7.15.1 Detailed Description

Definition at line 284 of file simulator.h.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 ComponentDescription::ComponentDescription (int *lp*, Component * *obj*) [inline]

Definition at line 287 of file simulator.h.

7.15.3 Member Data Documentation

7.15.3.1 int ComponentDescription::lpId

Definition at line 288 of file simulator.h.

Referenced by Link::Link(), and Link::Send().

7.15.3.2 Component* ComponentDescription::ptr

Definition at line 289 of file simulator.h.

Referenced by Link::Link().

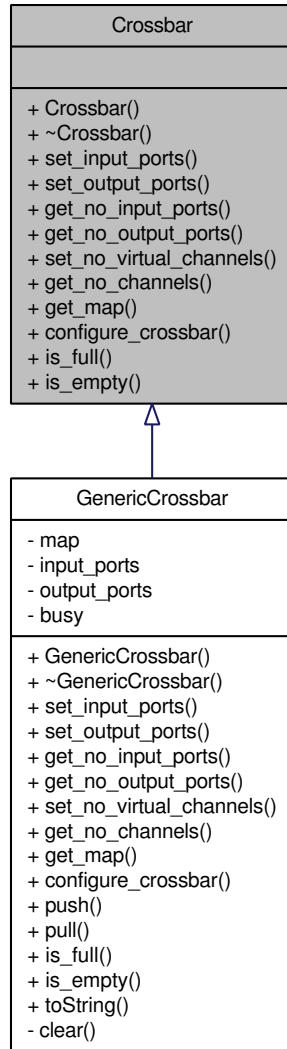
The documentation for this class was generated from the following file:

- **simulator.h**

7.16 Crossbar Class Reference

```
#include <crossbar.h>
```

Inheritance diagram for Crossbar:



Public Member Functions

- **Crossbar ()**
- virtual **~Crossbar ()**
- virtual void **set_input_ports** (unsigned int ports)=0
- virtual void **set_output_ports** (unsigned int ports)=0
- virtual unsigned int **get_no_input_ports** ()=0
- virtual unsigned int **get_no_output_ports** ()=0
- virtual void **set_no_virtual_channels** (unsigned int number)=0
- virtual unsigned int **get_no_channels** ()=0
- virtual unsigned int **get_map** (unsigned int input_port, unsigned int channel)=0

- virtual void **configure_crossbar** (unsigned int input_port, unsigned int output_port, unsigned int channel)=0
- virtual bool **is_full** (unsigned int input_port, unsigned int channel)=0
- virtual bool **is_empty** (unsigned int output_port, unsigned int channel)=0

7.16.1 Detailed Description

Definition at line 32 of file crossbar.h.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 Crossbar::Crossbar () [inline]

Definition at line 35 of file crossbar.h.

7.16.2.2 virtual Crossbar::~Crossbar () [inline, virtual]

Definition at line 36 of file crossbar.h.

7.16.3 Member Function Documentation

7.16.3.1 virtual void Crossbar::configure_crossbar (unsigned int *input_port*, unsigned int *output_port*, unsigned int *channel*) [pure virtual]

Implemented in **GenericCrossbar** (p. 121).

7.16.3.2 virtual unsigned int Crossbar::get_map (unsigned int *input_port*, unsigned int *channel*) [pure virtual]

Implemented in **GenericCrossbar** (p. 121).

7.16.3.3 virtual unsigned int Crossbar::get_no_channels () [pure virtual]

Implemented in **GenericCrossbar** (p. 121).

7.16.3.4 virtual unsigned int Crossbar::get_no_input_ports () [pure virtual]

Implemented in **GenericCrossbar** (p. 121).

7.16.3.5 virtual unsigned int Crossbar::get_no_output_ports () [pure virtual]

Implemented in **GenericCrossbar** (p. 122).

7.16.3.6 virtual bool Crossbar::is_empty (unsigned int *output_port*, unsigned int *channel*) [pure virtual]

Implemented in **GenericCrossbar** (p. 122).

7.16.3.7 `virtual bool Crossbar::is_full (unsigned int input_port, unsigned int channel) [pure virtual]`

Implemented in **GenericCrossbar** (p. 122).

7.16.3.8 `virtual void Crossbar::set_input_ports (unsigned int ports) [pure virtual]`

Implemented in **GenericCrossbar** (p. 123).

7.16.3.9 `virtual void Crossbar::set_no_virtual_channels (unsigned int number) [pure virtual]`

Implemented in **GenericCrossbar** (p. 123).

7.16.3.10 `virtual void Crossbar::set_output_ports (unsigned int ports) [pure virtual]`

Implemented in **GenericCrossbar** (p. 123).

The documentation for this class was generated from the following file:

- `crossbar.h`

7.17 Data Struct Reference

```
#include <request.h>
```

Public Attributes

- `unsigned long long int value`
- `short size`

7.17.1 Detailed Description

Definition at line 45 of file request.h.

7.17.2 Member Data Documentation

7.17.2.1 `short Data::size`

Definition at line 48 of file request.h.

Referenced by `MCFrontEnd::convertFromBitStream()`, `GenericTPG::convertFromBitStream()`, `DataBusHandler::process_event()`, and `DRAMCmdState::set()`.

7.17.2.2 `unsigned long long int Data::value`

Definition at line 47 of file request.h.

Referenced by `MCFrontEnd::convertFromBitStream()`, and `GenericTPG::convertFromBitStream()`.

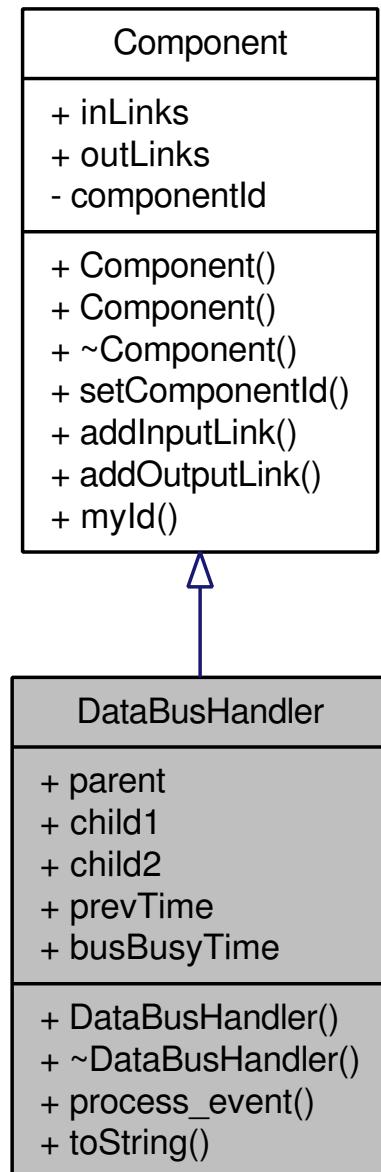
The documentation for this struct was generated from the following file:

- `request.h`

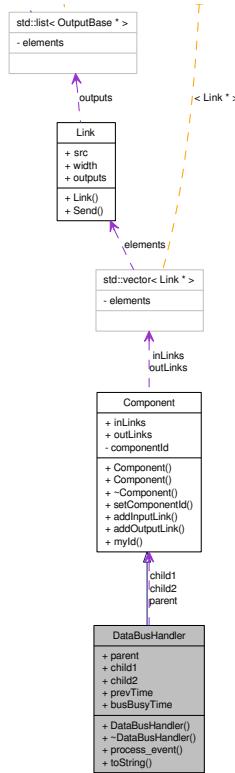
7.18 DataBusHandler Class Reference

```
#include <data_bus_handler.h>
```

Inheritance diagram for DataBusHandler:



Collaboration diagram for DataBusHandler:



Public Member Functions

- **DataBusHandler ()**
- **~DataBusHandler ()**
- **void process_event (IrisEvent *e)**
- **std::string toString ()**

Public Attributes

- **Component * parent**
- **Component * child1**
- **Component * child2**
- **Time prevTime**
- **Time busBusyTime**

7.18.1 Detailed Description

Definition at line 45 of file data_bus_handler.h.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 DataBusHandler::DataBusHandler ()

Definition at line 34 of file data_bus_handler.cc.

References busBusyTime, and prevTime.

7.18.2.2 DataBusHandler::~DataBusHandler ()

Definition at line 47 of file data_bus_handler.cc.

7.18.3 Member Function Documentation

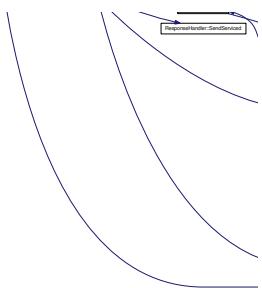
7.18.3.1 void DataBusHandler::process_event (IrisEvent * e)

Definition at line 59 of file data_bus_handler.cc.

References Request::bankNo, BUS_CYCLE, busBusyTime, child1, child2, Request::data, DDR_BUS_WIDTH, IrisEvent::event_data, Simulator::Now(), prevTime, DRAMChannel::process_event(), ResponseHandler::process_event(), Request::rankNo, REPLY, DRAMCmdState::req, Simulator::Schedule(), Data::size, START_READ, START_WRITE, t_CMD, and IrisEvent::type.

Referenced by DRAMChannel::process_event(), and CmdIssuer::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.18.3.2 std::string DataBusHandler::toString ()

7.18.4 Member Data Documentation

7.18.4.1 Time DataBusHandler::busBusyTime

Definition at line 56 of file data_bus_handler.h.

Referenced by DataBusHandler(), and process_event().

7.18.4.2 Component* DataBusHandler::child1

Definition at line 51 of file data_bus_handler.h.

Referenced by process_event(), and Bus::SetLinks().

7.18.4.3 Component* DataBusHandler::child2

Definition at line 52 of file data_bus_handler.h.

Referenced by process_event(), and Bus::SetLinks().

7.18.4.4 Component* DataBusHandler::parent

Definition at line 50 of file data_bus_handler.h.

Referenced by Bus::SetLinks().

7.18.4.5 Time DataBusHandler::prevTime

Definition at line 55 of file data_bus_handler.h.

Referenced by DataBusHandler(), and process_event().

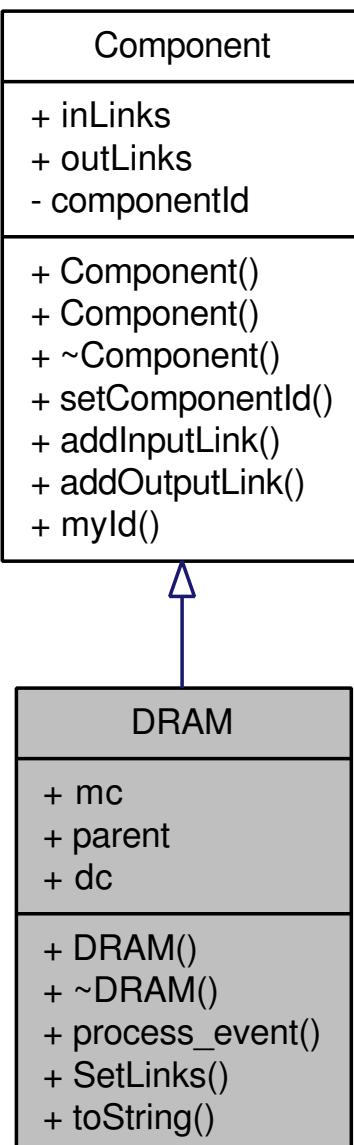
The documentation for this class was generated from the following files:

- [data_bus_handler.h](#)
- [data_bus_handler.cc](#)

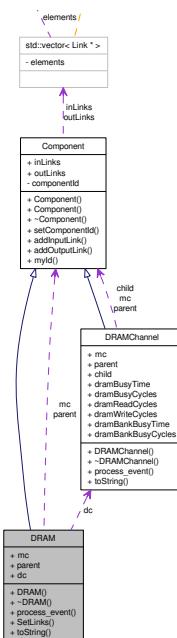
7.19 DRAM Class Reference

```
#include <dram.h>
```

Inheritance diagram for DRAM:



Collaboration diagram for DRAM:



Public Member Functions

- **DRAM ()**
- **~DRAM ()**
- **void process_event (IrisEvent *e)**
- **void SetLinks ()**
- **std::string toString ()**

Public Attributes

- **Component * mc**
- **Component * parent**
- **DRAMChannel dc [NO_OF_CHANNELS]**

7.19.1 Detailed Description

Definition at line 70 of file dram.h.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 DRAM::DRAM ()

Definition at line 36 of file dram.cc.

References dc, DRAMChannel::dramBankBusyCycles, DRAMChannel::dramBankBusyTime, DRAMChannel::dramBusyCycles, DRAMChannel::dramBusyTime, NO_OF_BANKS, NO_OF_CHANNELS, and NO_OF_RANKS.

7.19.2.2 DRAM::~DRAM ()

Definition at line 60 of file dram.cc.

7.19.3 Member Function Documentation

7.19.3.1 void DRAM::process_event (IrisEvent * e)

Definition at line 82 of file dram.cc.

7.19.3.2 void DRAM::SetLinks ()

Definition at line 65 of file dram.cc.

References DRAMChannel::child, dc, mc, DRAMChannel::mc, NO_OF_CHANNELS, parent, and DRAMChannel::parent.

Referenced by MC::Init().

Here is the caller graph for this function:



7.19.3.3 std::string DRAM::toString ()

7.19.4 Member Data Documentation

7.19.4.1 DRAMChannel DRAM::dc[NO_OF_CHANNELS]

Definition at line 77 of file dram.h.

Referenced by DRAM(), and SetLinks().

7.19.4.2 Component* DRAM::mc

Definition at line 75 of file dram.h.

Referenced by MC::Init(), and SetLinks().

7.19.4.3 Component* DRAM::parent

Definition at line 76 of file dram.h.

Referenced by MC::Init(), and SetLinks().

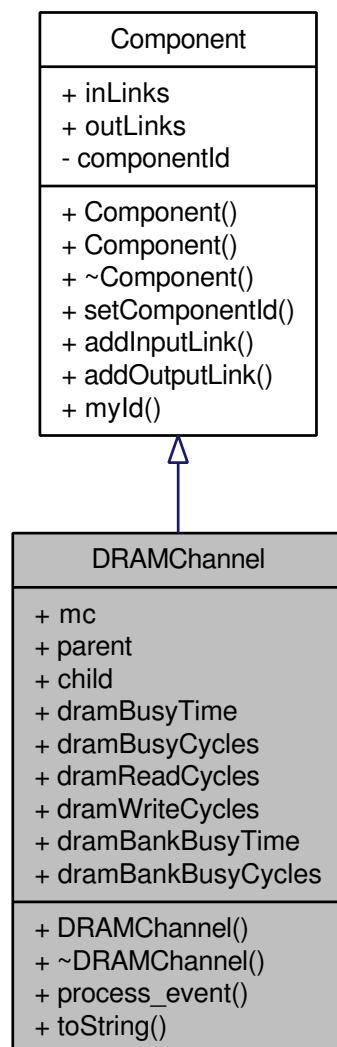
The documentation for this class was generated from the following files:

- **dram.h**
- **dram.cc**

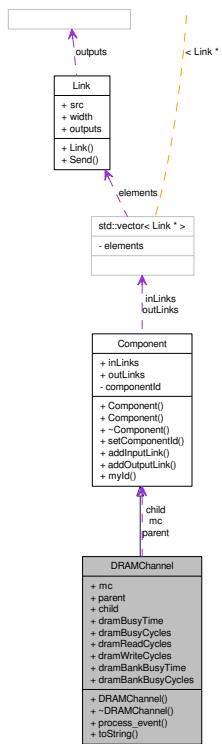
7.20 DRAMChannel Class Reference

```
#include <dram.h>
```

Inheritance diagram for DRAMChannel:



Collaboration diagram for DRAMChannel:



Public Member Functions

- **DRAMChannel ()**
- **~DRAMChannel ()**
- **void process_event (IrisEvent *e)**
- **std::string toString ()**

Public Attributes

- **Component * mc**
- **Component * parent**
- **Component * child**
- **unsigned long long int dramBusyTime**
- **unsigned long long int dramBusyCycles**

- `unsigned long long int dramReadCycles`
- `unsigned long long int dramWriteCycles`
- `unsigned long long int dramBankBusyTime [NO_OF_RANKS][NO_OF_BANKS]`
- `unsigned long long int dramBankBusyCycles [NO_OF_RANKS][NO_OF_BANKS]`

7.20.1 Detailed Description

Definition at line 35 of file `dram.h`.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 DRAMChannel::DRAMChannel ()

Definition at line 105 of file `dram.cc`.

7.20.2.2 DRAMChannel::~DRAMChannel ()

Definition at line 117 of file `dram.cc`.

7.20.3 Member Function Documentation

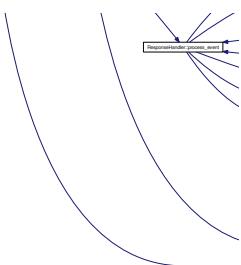
7.20.3.1 void DRAMChannel::process_event (IrisEvent * e)

Definition at line 129 of file `dram.cc`.

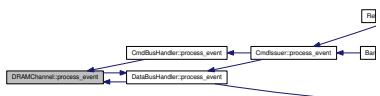
References ACTIVATE, Request::address, ALL_BANK_REFRESH, Request::arrivalTime, Request::bankNo, child, DRAMCmdState::cmd, dramBankBusyCycles, dramBankBusyTime, dramBusyCycles, dramBusyTime, IrisEvent::event_data, mc, NO_OF_BANKS, Simulator::Now(), PRECHARGE, DataBusHandler::process_event(), Request::rankNo, READ, DRAMCmdState::req, Request::retireTime, Simulator::Schedule(), START_READ, START_WRITE, t_CAS, t_RCD, t_RFC, t_WP, IrisEvent::type, and WRITE.

Referenced by DataBusHandler::process_event(), and CmdBusHandler::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.20.3.2 std::string DRAMChannel::toString ()

7.20.4 Member Data Documentation

7.20.4.1 Component* DRAMChannel::child

Definition at line 42 of file dram.h.

Referenced by process_event(), and DRAM::SetLinks().

7.20.4.2 unsigned long long int DRAMChannel::dramBankBusyCycles[NO_OF_RANKS][NO_OF_BANKS]

Definition at line 51 of file dram.h.

Referenced by DRAM::DRAM(), and process_event().

7.20.4.3 unsigned long long int DRAMChannel::dramBankBusyTime[NO_OF_RANKS][NO_OF_BANKS]

Definition at line 50 of file dram.h.

Referenced by DRAM::DRAM(), and process_event().

7.20.4.4 unsigned long long int DRAMChannel::dramBusyCycles

Definition at line 47 of file dram.h.

Referenced by DRAM::DRAM(), and process_event().

7.20.4.5 unsigned long long int DRAMChannel::dramBusyTime

Definition at line 46 of file dram.h.

Referenced by DRAM::DRAM(), and process_event().

7.20.4.6 unsigned long long int DRAMChannel::dramReadCycles

Definition at line 48 of file dram.h.

7.20.4.7 unsigned long long int DRAMChannel::dramWriteCycles

Definition at line 49 of file dram.h.

7.20.4.8 Component* DRAMChannel::mc

Definition at line 40 of file dram.h.

Referenced by process_event(), and DRAM::SetLinks().

7.20.4.9 Component* DRAMChannel::parent

Definition at line 41 of file dram.h.

Referenced by DRAM::SetLinks().

The documentation for this class was generated from the following files:

- **dram.h**
- **dram.cc**

7.21 DRAMCmdState Struct Reference

```
#include <cmd_issuer.h>
```

Collaboration diagram for DRAMCmdState:



Public Member Functions

- void **set** (**DRAMCmd** cmdS, **BurstLength** burstL, **Request** req)
- **DRAMCmdState** ()
- **~DRAMCmdState** ()

Public Attributes

- **DRAMCmd** cmd
- **Request** req
- **BurstLength** burst

7.21.1 Detailed Description

Definition at line 44 of file cmd_issuer.h.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 DRAMCmdState::DRAMCmdState () [inline]

Definition at line 64 of file cmd_issuer.h.

References burst, cmd, NORMAL, and PRECHARGE.

7.21.2.2 DRAMCmdState::~DRAMCmdState () [inline]

Definition at line 69 of file cmd_issuer.h.

7.21.3 Member Function Documentation

7.21.3.1 void DRAMCmdState::set (DRAMCmd *cmds*, BurstLength *burstL*, Request *reqI*) [inline]

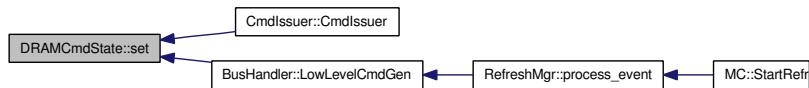
TODO needs to set this

Definition at line 49 of file cmd_issuer.h.

References burst, cmd, Request::data, PREFETCH_SIZE, PREFETCHL, READ_SIZE, READL, req, Data::size, WRITE_SIZE, WRITEBACK_SIZE, WRITEBACKL, and WRITEL.

Referenced by CmdIssuer::CmdIssuer(), and BusHandler::LowLevelCmdGen().

Here is the caller graph for this function:



7.21.4 Member Data Documentation

7.21.4.1 BurstLength DRAMCmdState::burst

Definition at line 48 of file cmd_issuer.h.

Referenced by CmdIssuer::CalculateBurstL(), DRAMCmdState(), and set().

7.21.4.2 DRAMCmd DRAMCmdState::cmd

Definition at line 46 of file cmd_issuer.h.

Referenced by CmdIssuer::BankNotBusy(), CmdIssuer::BusNotBusy(), CmdIssuer::CalculateDataDelay(), CmdIssuer::CanSchedule(), DRAMCmdState(), DRAMChannel::process_event(), CmdIssuer::process_event(), set(), CmdIssuer::SetBusBusyTime(), and CmdIssuer::SetPrevState().

7.21.4.3 Request DRAMCmdState::req

Definition at line 47 of file cmd_issuer.h.

Referenced by CmdIssuer::BankNotBusy(), CmdIssuer::BanksFirstReq(), CmdIssuer::CanSchedule(), ResponseHandler::process_event(), DRAMChannel::process_event(), DataBusHandler::process_event(), CmdIssuer::process_event(), ResponseHandler::SearchBuffer(), set(), and CmdIssuer::SetPrevState().

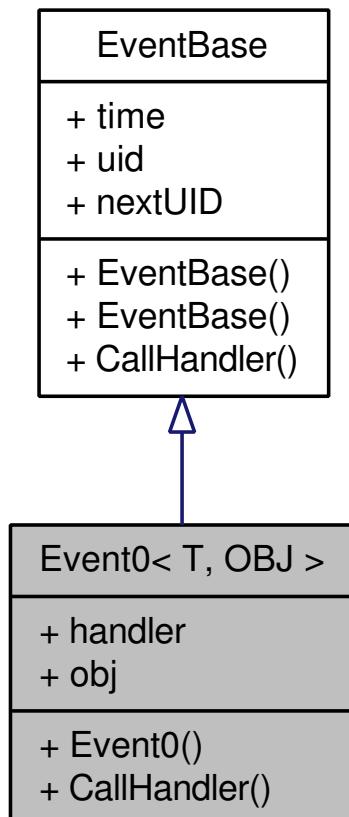
The documentation for this struct was generated from the following file:

- [cmd_issuer.h](#)

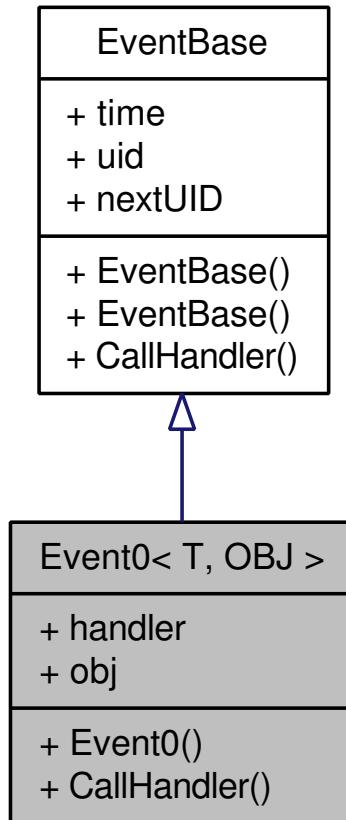
7.22 Event0< T, OBJ > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event0< T, OBJ >:



Collaboration diagram for Event0< T, OBJ >:



Public Member Functions

- **Event0** (double t, void(T::*f)(void), OBJ *obj0)
- void **CallHandler** ()

Public Attributes

- void(T::* **handler**) (void)
- OBJ * **obj**

7.22.1 Detailed Description

`template<typename T, typename OBJ> class Event0< T, OBJ >`

Definition at line 48 of file simulator.h.

7.22.2 Constructor & Destructor Documentation

**7.22.2.1 template<typename T, typename OBJ> Event0< T, OBJ >::Event0
(double t, void(T::*)(void) f, OBJ * obj0) [inline]**

Definition at line 51 of file simulator.h.

7.22.3 Member Function Documentation

7.22.3.1 template<typename T , typename OBJ > void Event0< T, OBJ >::CallHandler () [inline, virtual]

Implements **EventBase** (p. 106).

Definition at line 60 of file simulator.h.

References Event0< T, OBJ >::handler, and Event0< T, OBJ >::obj.

7.22.4 Member Data Documentation

7.22.4.1 template<typename T, typename OBJ> void(T::* Event0< T, OBJ >::handler)(void)

Referenced by Event0< T, OBJ >::CallHandler().

7.22.4.2 template<typename T, typename OBJ> OBJ* Event0< T, OBJ >::obj

Definition at line 54 of file simulator.h.

Referenced by Event0< T, OBJ >::CallHandler().

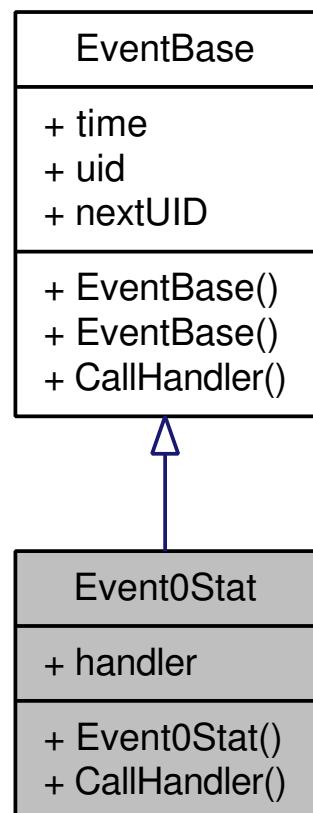
The documentation for this class was generated from the following file:

- **simulator.h**

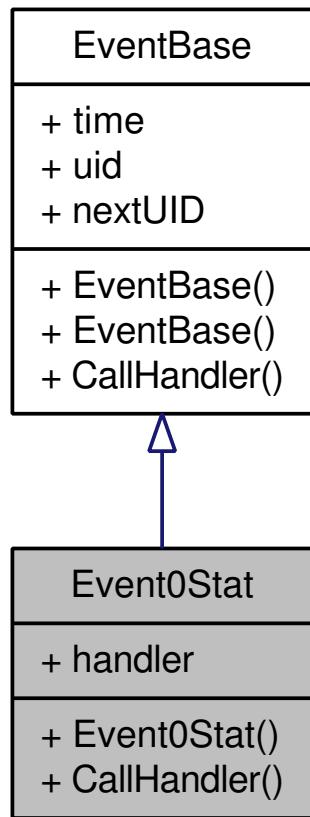
7.23 Event0Stat Class Reference

```
#include <simulator.h>
```

Inheritance diagram for Event0Stat:



Collaboration diagram for Event0Stat:



Public Member Functions

- `Event0Stat (double t, void(*f)(void))`
- `void CallHandler ()`

Public Attributes

- `void(* handler)(void)`

7.23.1 Detailed Description

Definition at line 168 of file simulator.h.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 `Event0Stat::Event0Stat (double t, void(*)(void) f) [inline]`

Definition at line 171 of file simulator.h.

7.23.3 Member Function Documentation

7.23.3.1 void Event0Stat::CallHandler () [virtual]

Implements **EventBase** (p. 106).

Definition at line 15 of file simulator.cc.

References handler.

7.23.4 Member Data Documentation

7.23.4.1 void(* Event0Stat::handler)(void)

Referenced by CallHandler().

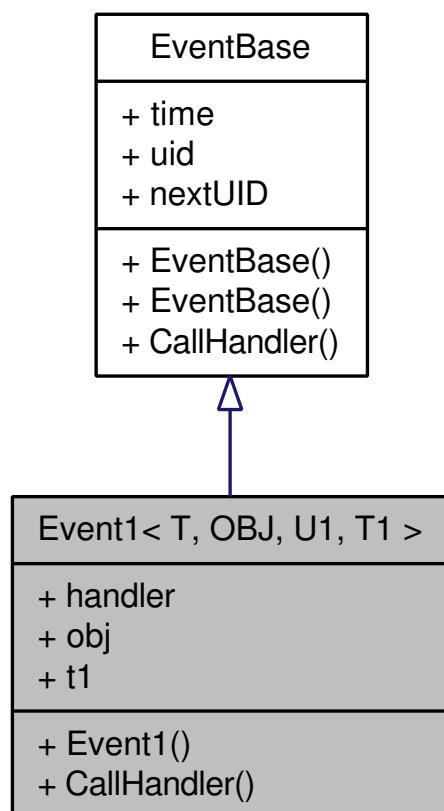
The documentation for this class was generated from the following files:

- **simulator.h**
- **simulator.cc**

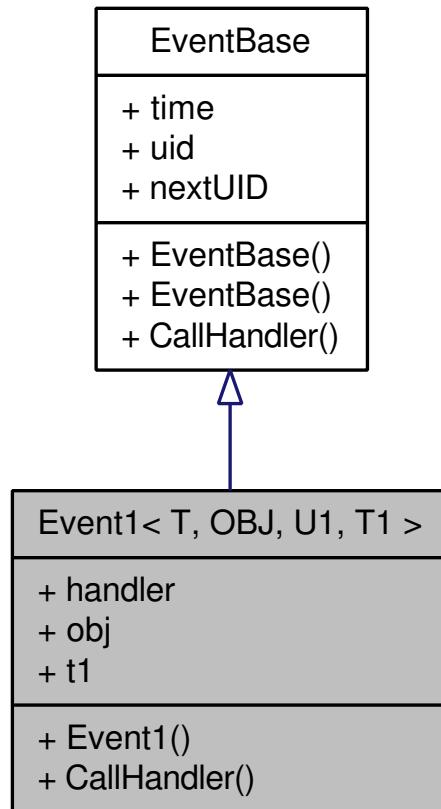
7.24 Event1< T, OBJ, U1, T1 > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event1< T, OBJ, U1, T1 >:



Collaboration diagram for Event1< T, OBJ, U1, T1 >:



Public Member Functions

- **Event1** (double t, void(T::*f)(U1), OBJ *obj0, T1 t1_0)
- void **CallHandler** ()

Public Attributes

- void(T::* **handler**)(U1)
- OBJ * **obj**
- T1 **t1**

7.24.1 Detailed Description

```
template<typename T, typename OBJ, typename U1, typename T1> class Event1< T, OBJ, U1, T1 >
```

Definition at line 66 of file simulator.h.

7.24.2 Constructor & Destructor Documentation

**7.24.2.1 template<typename T, typename OBJ, typename U1, typename T1>
Event1< T, OBJ, U1, T1 >::Event1 (double t, void(T::*)(U1) f, OBJ *
obj0, T1 t1_0) [inline]**

Definition at line 69 of file simulator.h.

7.24.3 Member Function Documentation

**7.24.3.1 template<typename T, typename OBJ, typename U1, typename T1>
void Event1< T, OBJ, U1, T1 >::CallHandler () [inline, virtual]**

Implements **EventBase** (p. 106).

Definition at line 79 of file simulator.h.

References `Event1< T, OBJ, U1, T1 >::handler`, `Event1< T, OBJ, U1, T1 >::obj`, and `Event1< T, OBJ, U1, T1 >::t1`.

7.24.4 Member Data Documentation

**7.24.4.1 template<typename T, typename OBJ, typename U1, typename T1>
void(T::* Event1< T, OBJ, U1, T1 >::handler)(U1)**

Referenced by `Event1< T, OBJ, U1, T1 >::CallHandler()`.

**7.24.4.2 template<typename T, typename OBJ, typename U1, typename T1>
OBJ* Event1< T, OBJ, U1, T1 >::obj**

Definition at line 72 of file simulator.h.

Referenced by `Event1< T, OBJ, U1, T1 >::CallHandler()`.

**7.24.4.3 template<typename T, typename OBJ, typename U1, typename T1> T1
Event1< T, OBJ, U1, T1 >::t1**

Definition at line 73 of file simulator.h.

Referenced by `Event1< T, OBJ, U1, T1 >::CallHandler()`.

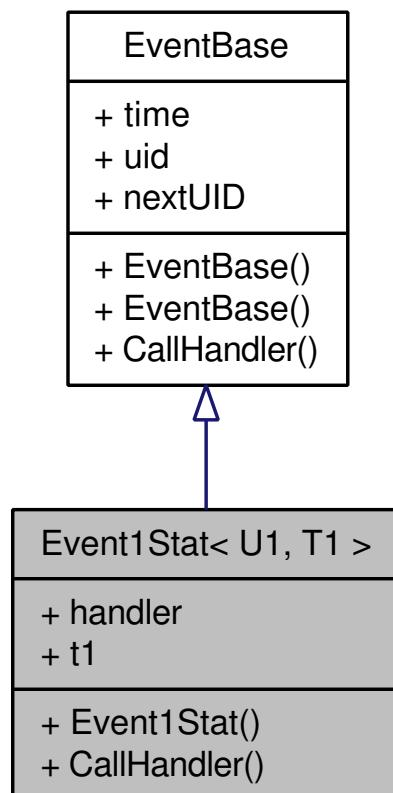
The documentation for this class was generated from the following file:

- **simulator.h**

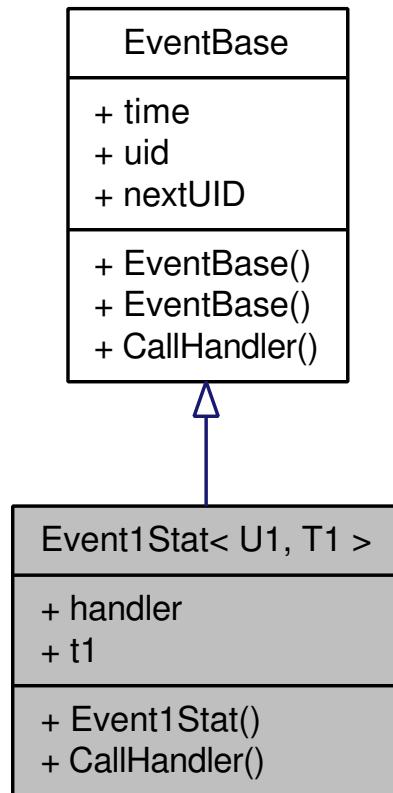
7.25 Event1Stat< U1, T1 > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event1Stat< U1, T1 >:



Collaboration diagram for Event1Stat< U1, T1 >:



Public Member Functions

- **Event1Stat** (double t, void(*f)(U1), T1 t1_0)
- void **CallHandler** ()

Public Attributes

- void(* **handler**)(U1)
- T1 **t1**

7.25.1 Detailed Description

`template<typename U1, typename T1> class Event1Stat< U1, T1 >`

Definition at line 179 of file simulator.h.

7.25.2 Constructor & Destructor Documentation

7.25.2.1 `template<typename U1, typename T1> Event1Stat< U1, T1 >::Event1Stat (double t, void(*)(U1) f, T1 t1_0) [inline]`

Definition at line 182 of file simulator.h.

7.25.3 Member Function Documentation

7.25.3.1 `template<typename U1 , typename T1 > void Event1Stat< U1, T1 >::CallHandler () [inline, virtual]`

Implements **EventBase** (p. 106).

Definition at line 191 of file simulator.h.

References `Event1Stat< U1, T1 >::handler`, and `Event1Stat< U1, T1 >::t1`.

7.25.4 Member Data Documentation

7.25.4.1 `template<typename U1, typename T1> void(* Event1Stat< U1, T1 >::handler)(U1)`

Referenced by `Event1Stat< U1, T1 >::CallHandler()`.

7.25.4.2 `template<typename U1, typename T1> T1 Event1Stat< U1, T1 >::t1`

Definition at line 185 of file simulator.h.

Referenced by `Event1Stat< U1, T1 >::CallHandler()`.

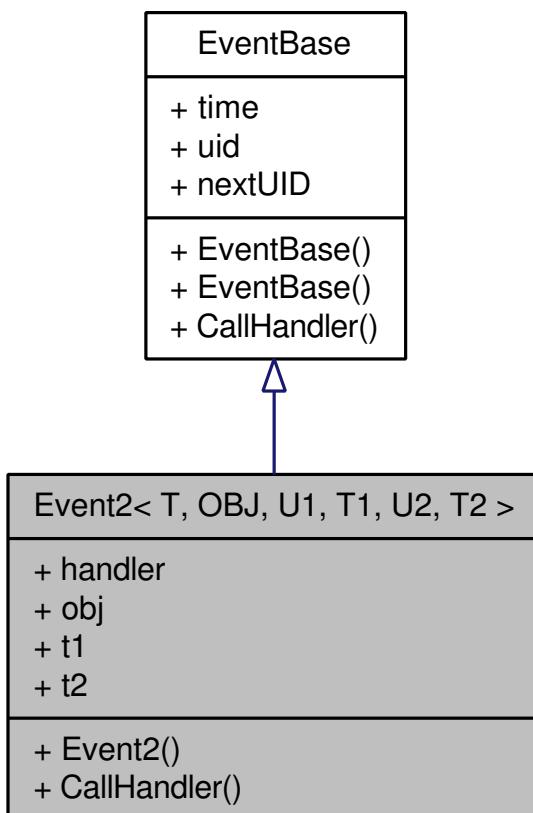
The documentation for this class was generated from the following file:

- `simulator.h`

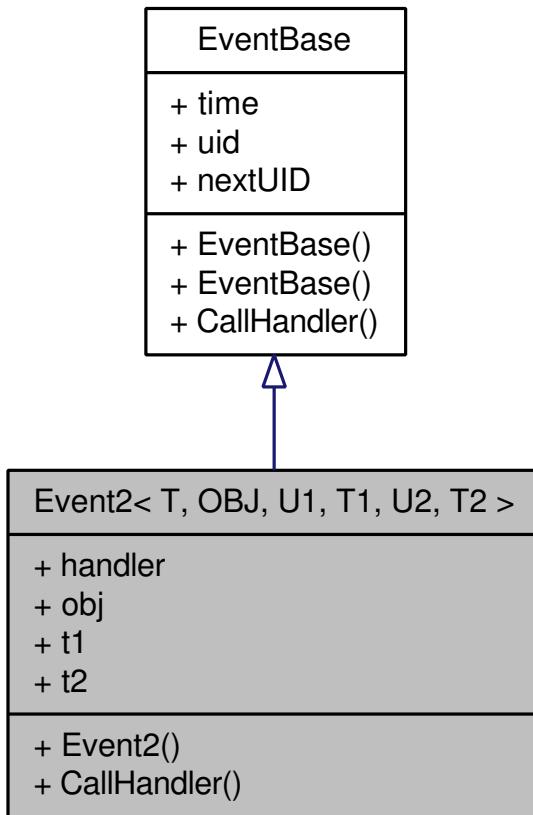
7.26 Event2< T, OBJ, U1, T1, U2, T2 > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event2< T, OBJ, U1, T1, U2, T2 >:



Collaboration diagram for Event2< T, OBJ, U1, T1, U2, T2 >:



Public Member Functions

- **Event2** (double t, void(T::*f)(U1, U2), OBJ *obj0, T1 t1_0, T2 t2_0)
- void **CallHandler** ()

Public Attributes

- void(T::* **handler**)(U1, U2)
- OBJ * **obj**
- T1 **t1**
- T2 **t2**

7.26.1 Detailed Description

`template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2> class Event2< T, OBJ, U1, T1, U2, T2 >`

Definition at line 87 of file simulator.h.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2> Event2< T, OBJ, U1, T1, U2, T2 >::Event2
 (double t, void(T::*)(U1, U2) f, OBJ * obj0, T1 t1_0, T2 t2_0)
 [inline]

Definition at line 90 of file simulator.h.

7.26.3 Member Function Documentation

7.26.3.1 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2> void Event2< T, OBJ, U1, T1, U2, T2 >::CallHandler () [inline, virtual]

Implements **EventBase** (p. 106).

Definition at line 103 of file simulator.h.

References Event2< T, OBJ, U1, T1, U2, T2 >::handler, Event2< T, OBJ, U1, T1, U2, T2 >::obj, Event2< T, OBJ, U1, T1, U2, T2 >::t1, and Event2< T, OBJ, U1, T1, U2, T2 >::t2.

7.26.4 Member Data Documentation

7.26.4.1 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2> void(T::* Event2< T, OBJ, U1, T1, U2, T2 >::handler)(U1, U2)

Referenced by Event2< T, OBJ, U1, T1, U2, T2 >::CallHandler().

7.26.4.2 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2> OBJ* Event2< T, OBJ, U1, T1, U2, T2 >::obj

Definition at line 93 of file simulator.h.

Referenced by Event2< T, OBJ, U1, T1, U2, T2 >::CallHandler().

7.26.4.3 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2> T1 Event2< T, OBJ, U1, T1, U2, T2 >::t1

Definition at line 94 of file simulator.h.

Referenced by Event2< T, OBJ, U1, T1, U2, T2 >::CallHandler().

7.26.4.4 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2> T2 Event2< T, OBJ, U1, T1, U2, T2 >::t2

Definition at line 95 of file simulator.h.

Referenced by Event2< T, OBJ, U1, T1, U2, T2 >::CallHandler().

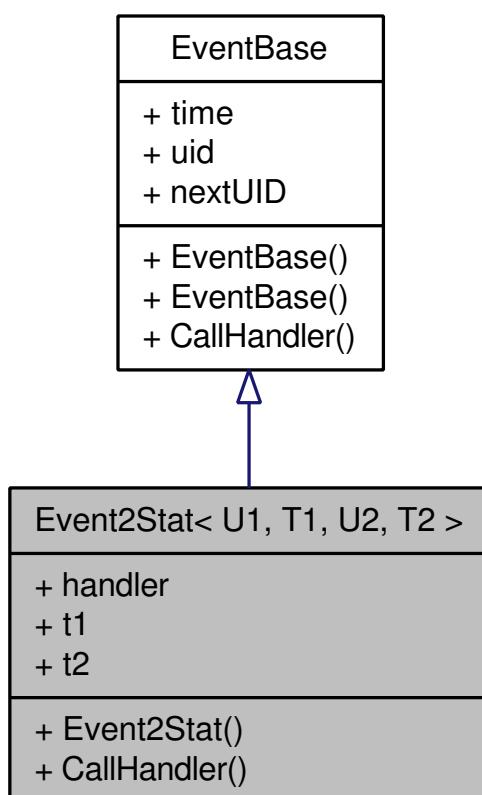
The documentation for this class was generated from the following file:

- `simulator.h`

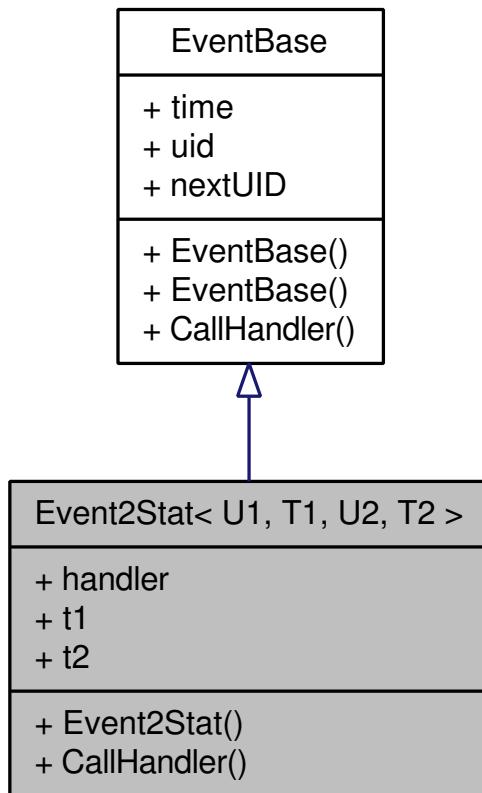
7.27 Event2Stat< U1, T1, U2, T2 > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event2Stat< U1, T1, U2, T2 >:



Collaboration diagram for Event2Stat< U1, T1, U2, T2 >:



Public Member Functions

- **Event2Stat** (double t, void(*f)(U1, U2), T1 t1_0, T2 t2_0)
- void **CallHandler** ()

Public Attributes

- void(* **handler**)(U1, U2)
- T1 **t1**
- T2 **t2**

7.27.1 Detailed Description

```
template<typename U1, typename T1, typename U2, typename T2> class
Event2Stat< U1, T1, U2, T2 >
```

Definition at line 199 of file simulator.h.

7.27.2 Constructor & Destructor Documentation

**7.27.2.1 template<typename U1, typename T1, typename U2, typename T2>
Event2Stat< U1, T1, U2, T2 >::Event2Stat (double t, void(*)(U1, U2) f,
T1 t1_0, T2 t2_0) [inline]**

Definition at line 202 of file simulator.h.

7.27.3 Member Function Documentation

**7.27.3.1 template<typename U1, typename T1, typename U2, typename T2>
void Event2Stat< U1, T1, U2, T2 >::CallHandler () [inline, virtual]**

Implements **EventBase** (p. 106).

Definition at line 213 of file simulator.h.

References `Event2Stat< U1, T1, U2, T2 >::handler`, `Event2Stat< U1, T1, U2, T2 >::t1`, and `Event2Stat< U1, T1, U2, T2 >::t2`.

7.27.4 Member Data Documentation

**7.27.4.1 template<typename U1, typename T1, typename U2, typename T2>
void(* Event2Stat< U1, T1, U2, T2 >::handler)(U1, U2)**

Referenced by `Event2Stat< U1, T1, U2, T2 >::CallHandler()`.

**7.27.4.2 template<typename U1, typename T1, typename U2, typename T2> T1
Event2Stat< U1, T1, U2, T2 >::t1**

Definition at line 205 of file simulator.h.

Referenced by `Event2Stat< U1, T1, U2, T2 >::CallHandler()`.

**7.27.4.3 template<typename U1, typename T1, typename U2, typename T2> T2
Event2Stat< U1, T1, U2, T2 >::t2**

Definition at line 206 of file simulator.h.

Referenced by `Event2Stat< U1, T1, U2, T2 >::CallHandler()`.

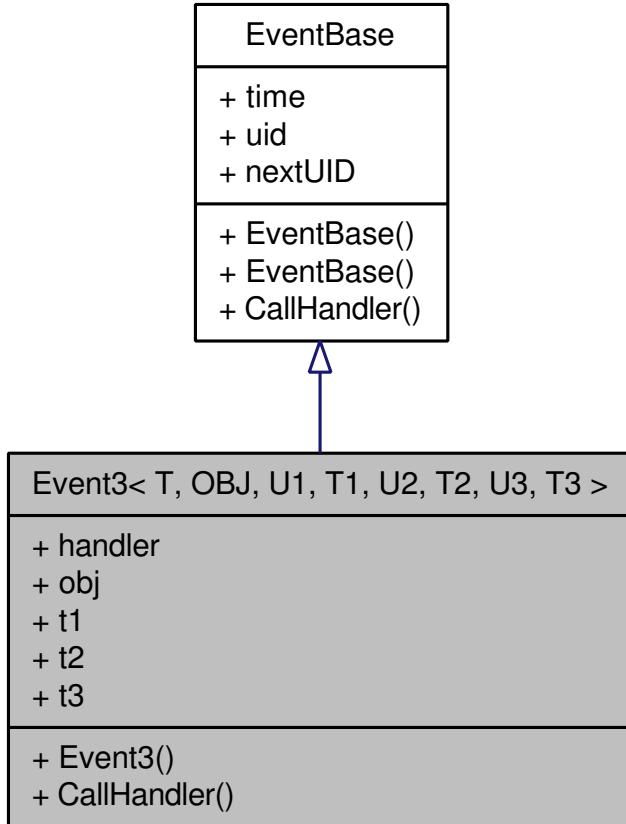
The documentation for this class was generated from the following file:

- **simulator.h**

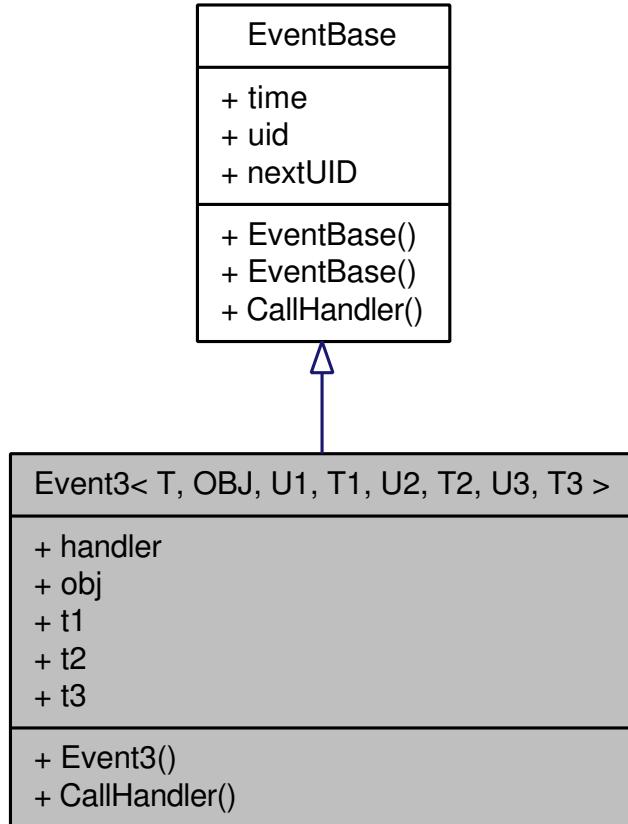
7.28 Event3< T, OBJ, U1, T1, U2, T2, U3, T3 > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >:



Collaboration diagram for Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >:



Public Member Functions

- `Event3` (double t, void(T::*f)(U1, U2, U3), OBJ *obj0, T1 t1_0, T2 t2_0, T3 t3_0)
- void `CallHandler` ()

Public Attributes

- void(T::* **handler**) (U1, U2, U3)
- OBJ * **obj**
- T1 **t1**
- T2 **t2**
- T3 **t3**

7.28.1 Detailed Description

`template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2, typename U3, typename T3> class Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >`

Definition at line 112 of file simulator.h.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 `template<typename T, typename OBJ, typename U1, typename T1,
typename U2, typename T2, typename U3, typename T3> Event3< T,
OBJ, U1, T1, U2, T2, U3, T3 >::Event3 (double t, void(T::*)(U1, U2,
U3) f, OBJ * obj0, T1 t1_0, T2 t2_0, T3 t3_0) [inline]`

Definition at line 114 of file simulator.h.

7.28.3 Member Function Documentation

7.28.3.1 `template<typename T , typename OBJ , typename U1 , typename T1 ,
typename U2 , typename T2 , typename U3 , typename T3 > void
Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::CallHandler () [inline,
virtual]`

Implements **EventBase** (p. 106).

Definition at line 130 of file simulator.h.

References `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::handler`, `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::obj`, `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::t1`, `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::t2`, and `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::t3`.

7.28.4 Member Data Documentation

7.28.4.1 `template<typename T, typename OBJ, typename U1, typename T1,
typename U2, typename T2, typename U3, typename T3> void(T::*
Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::handler)(U1, U2, U3)`

Referenced by `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::CallHandler()`.

7.28.4.2 `template<typename T, typename OBJ, typename U1, typename T1,
typename U2, typename T2, typename U3, typename T3> OBJ* Event3<
T, OBJ, U1, T1, U2, T2, U3, T3 >::obj`

Definition at line 117 of file simulator.h.

Referenced by `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::CallHandler()`.

7.28.4.3 `template<typename T, typename OBJ, typename U1, typename T1,
typename U2, typename T2, typename U3, typename T3> T1 Event3< T,
OBJ, U1, T1, U2, T2, U3, T3 >::t1`

Definition at line 118 of file simulator.h.

Referenced by `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::CallHandler()`.

7.28.4.4 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2, typename U3, typename T3> T2 Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::t2

Definition at line 119 of file simulator.h.

Referenced by Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::CallHandler().

7.28.4.5 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2, typename U3, typename T3> T3 Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::t3

Definition at line 120 of file simulator.h.

Referenced by Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >::CallHandler().

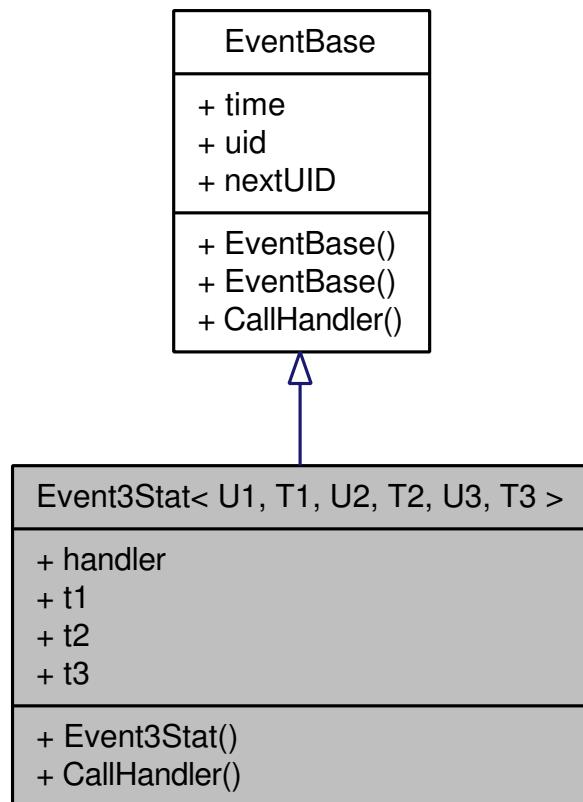
The documentation for this class was generated from the following file:

- **simulator.h**

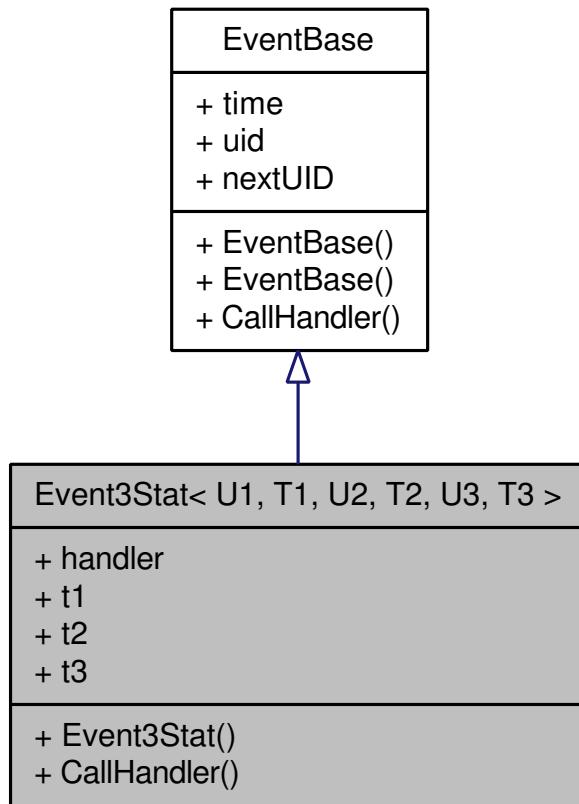
7.29 Event3Stat< U1, T1, U2, T2, U3, T3 > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event3Stat< U1, T1, U2, T2, U3, T3 >:



Collaboration diagram for Event3Stat< U1, T1, U2, T2, U3, T3 >:



Public Member Functions

- **Event3Stat** (double t, void(*f)(U1, U2, U3), T1 t1_0, T2 t2_0, T3 t3_0)
- void **CallHandler** ()

Public Attributes

- void(* **handler**)(U1, U2, U3)
- T1 **t1**
- T2 **t2**
- T3 **t3**

7.29.1 Detailed Description

```
template<typename U1, typename T1, typename U2, typename T2, typename U3,
typename T3> class Event3Stat< U1, T1, U2, T2, U3, T3 >
```

Definition at line 221 of file simulator.h.

7.29.2 Constructor & Destructor Documentation

7.29.2.1 `template<typename U1, typename T1, typename U2, typename T2,
typename U3, typename T3> Event3Stat< U1, T1, U2, T2, U3, T3
>::Event3Stat (double t, void(*)(U1, U2, U3) f, T1 t1_0, T2 t2_0, T3
t3_0) [inline]`

Definition at line 223 of file simulator.h.

7.29.3 Member Function Documentation

7.29.3.1 `template<typename U1 , typename T1 , typename U2 , typename T2 ,
typename U3 , typename T3 > void Event3Stat< U1, T1, U2, T2, U3, T3
>::CallHandler () [inline, virtual]`

Implements **EventBase** (p. 106).

Definition at line 237 of file simulator.h.

References `Event3Stat< U1, T1, U2, T2, U3, T3 >::handler`, `Event3Stat< U1, T1, U2, T2, U3, T3 >::t1`, `Event3Stat< U1, T1, U2, T2, U3, T3 >::t2`, and `Event3Stat< U1, T1, U2, T2, U3, T3 >::t3`.

7.29.4 Member Data Documentation

7.29.4.1 `template<typename U1, typename T1, typename U2, typename T2,
typename U3, typename T3> void(* Event3Stat< U1, T1, U2, T2, U3, T3
>::handler)(U1, U2, U3)`

Referenced by `Event3Stat< U1, T1, U2, T2, U3, T3 >::CallHandler()`.

7.29.4.2 `template<typename U1, typename T1, typename U2, typename T2,
typename U3, typename T3> T1 Event3Stat< U1, T1, U2, T2, U3, T3
>::t1`

Definition at line 226 of file simulator.h.

Referenced by `Event3Stat< U1, T1, U2, T2, U3, T3 >::CallHandler()`.

7.29.4.3 `template<typename U1, typename T1, typename U2, typename T2,
typename U3, typename T3> T2 Event3Stat< U1, T1, U2, T2, U3, T3
>::t2`

Definition at line 227 of file simulator.h.

Referenced by `Event3Stat< U1, T1, U2, T2, U3, T3 >::CallHandler()`.

7.29.4.4 `template<typename U1, typename T1, typename U2, typename T2,
typename U3, typename T3> T3 Event3Stat< U1, T1, U2, T2, U3, T3
>::t3`

Definition at line 228 of file simulator.h.

Referenced by Event3Stat< U1, T1, U2, T2, U3, T3 >::CallHandler().

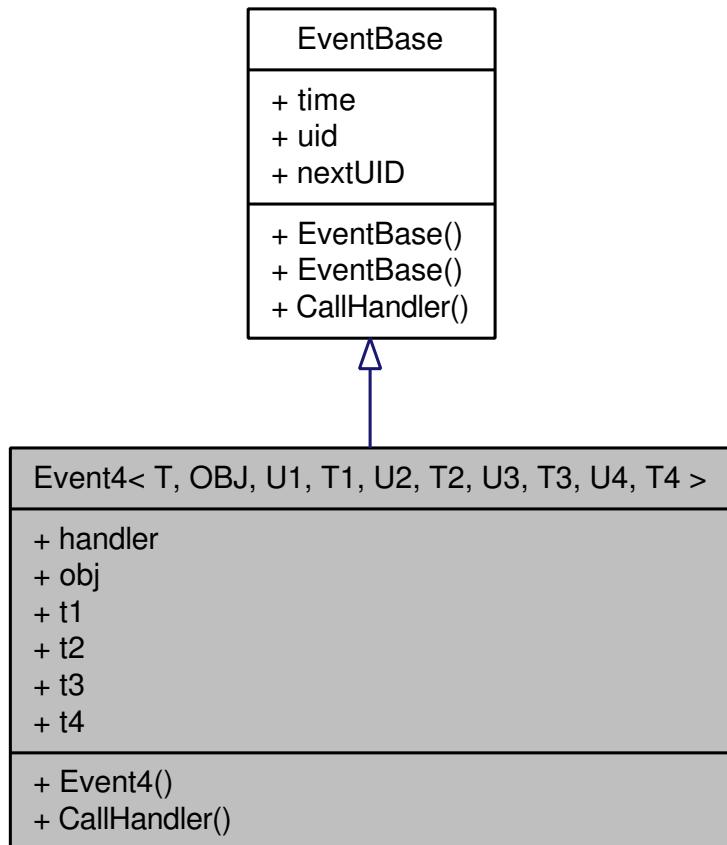
The documentation for this class was generated from the following file:

- **simulator.h**

7.30 Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 > Class Template Reference

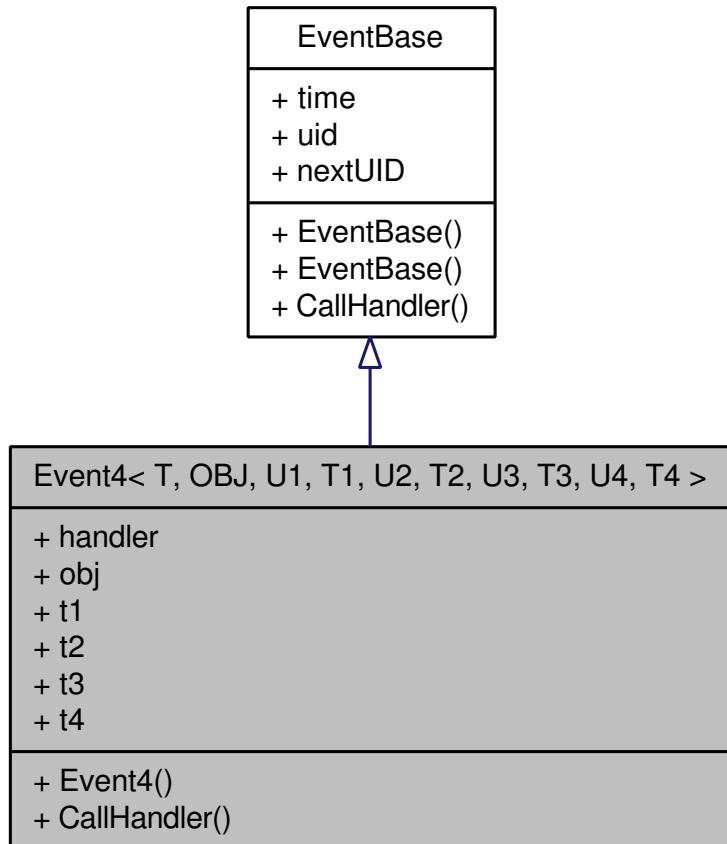
```
#include <simulator.h>
```

Inheritance diagram for Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >:



7.30 Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 > Class Template Reference

Collaboration diagram for Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >:



Public Member Functions

- `Event4` (double t, void(T::*f)(U1, U2, U3, U4), OBJ *obj0, T1 t1_0, T2 t2_0, T3 t3_0, T4 t4_0)
- void `CallHandler` ()

Public Attributes

- void(T::* **handler**) (U1, U2, U3, U4)
- OBJ * **obj**
- T1 **t1**
- T2 **t2**
- T3 **t3**
- T4 **t4**

7.30.1 Detailed Description

```
template<typename T, typename OBJ, typename U1, typename T1, typename U2,
typename T2, typename U3, typename T3, typename U4, typename T4> class
Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >
```

Definition at line 139 of file simulator.h.

7.30.2 Constructor & Destructor Documentation

```
7.30.2.1 template<typename T, typename OBJ, typename U1, typename T1,
typename U2, typename T2, typename U3, typename T3, typename
U4, typename T4> Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4
>::Event4 (double t, void(T::*)(U1, U2, U3, U4) f, OBJ * obj0, T1
t1_0, T2 t2_0, T3 t3_0, T4 t4_0) [inline]
```

Definition at line 141 of file simulator.h.

7.30.3 Member Function Documentation

```
7.30.3.1 template<typename T , typename OBJ , typename U1 , typename T1 ,
typename U2 , typename T2 , typename U3 , typename T3 , typename
U4 , typename T4 > void Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4,
T4 >::CallHandler () [inline, virtual]
```

Implements **EventBase** (p. 106).

Definition at line 159 of file simulator.h.

References Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::handler, Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::obj, Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t1, Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t2, Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t3, and Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t4.

7.30.4 Member Data Documentation

```
7.30.4.1 template<typename T, typename OBJ, typename U1, typename T1,
typename U2, typename T2, typename U3, typename T3, typename U4,
typename T4> void(T::* Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4,
T4 >::handler)(U1, U2, U3, U4)
```

Referenced by Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

```
7.30.4.2 template<typename T, typename OBJ, typename U1, typename T1,
typename U2, typename T2, typename U3, typename T3, typename U4,
typename T4> OBJ* Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4
>::obj
```

Definition at line 144 of file simulator.h.

Referenced by Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

7.30 Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 > Class Template Reference

7.30.4.3 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T1 Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t1

Definition at line 145 of file simulator.h.

Referenced by Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

7.30.4.4 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T2 Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t2

Definition at line 146 of file simulator.h.

Referenced by Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

7.30.4.5 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T3 Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t3

Definition at line 147 of file simulator.h.

Referenced by Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

7.30.4.6 template<typename T, typename OBJ, typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T4 Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::t4

Definition at line 148 of file simulator.h.

Referenced by Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

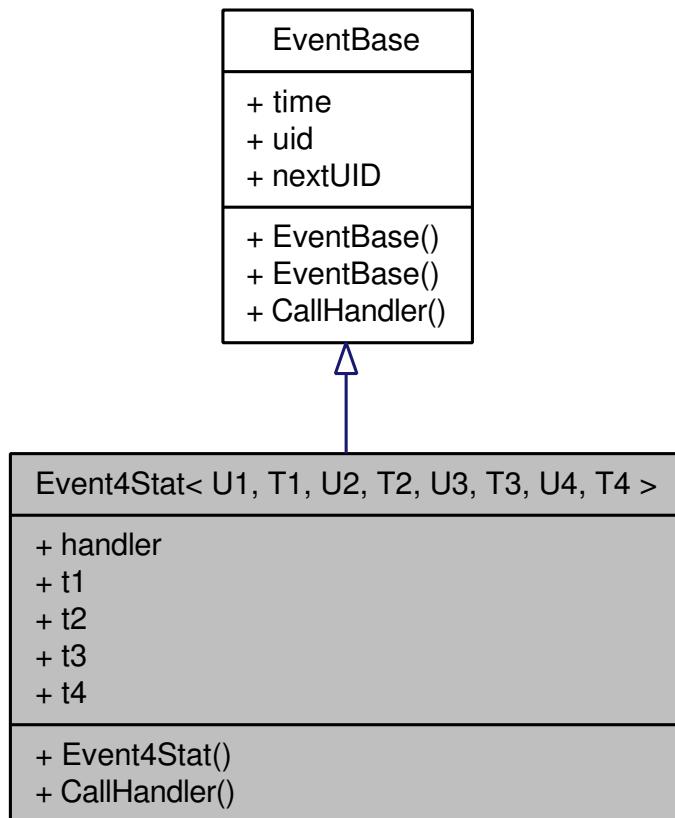
The documentation for this class was generated from the following file:

- **simulator.h**

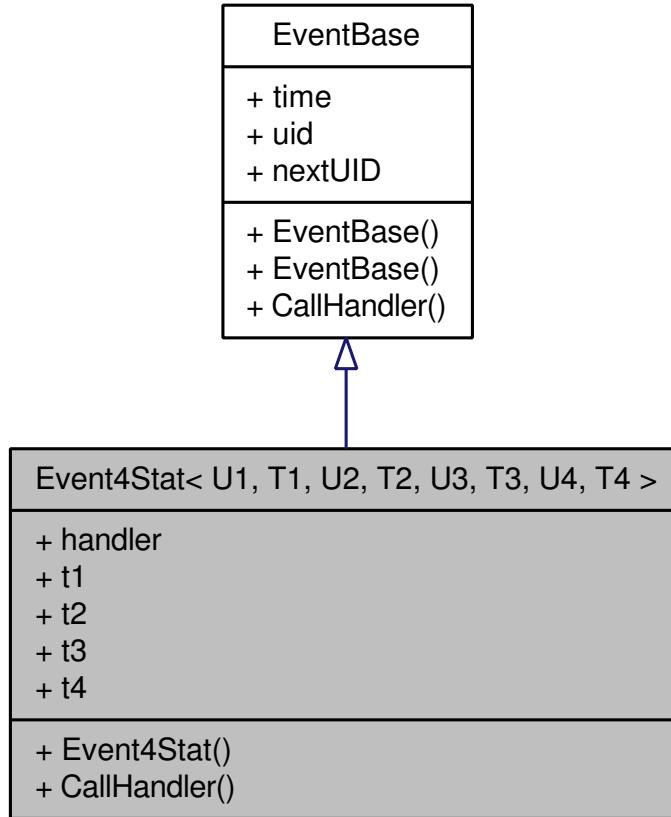
7.31 Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 > Class Template Reference

```
#include <simulator.h>
```

Inheritance diagram for Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >:



Collaboration diagram for Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >:



Public Member Functions

- `Event4Stat (double t, void(*f)(U1, U2, U3, U4), T1 t1_0, T2 t2_0, T3 t3_0, T4 t4_0)`
- `void CallHandler ()`

Public Attributes

- `void(* handler)(U1, U2, U3, U4)`
- `T1 t1`
- `T2 t2`
- `T3 t3`
- `T4 t4`

7.31.1 Detailed Description

`template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> class Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >`

Definition at line 246 of file simulator.h.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 `template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::Event4Stat(double t, void(*)(U1, U2, U3, U4) f, T1 t1_0, T2 t2_0, T3 t3_0, T4 t4_0) [inline]`

Definition at line 248 of file simulator.h.

7.31.3 Member Function Documentation

7.31.3.1 `template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> void Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::CallHandler() [inline, virtual]`

Implements **EventBase** (p. 106).

Definition at line 264 of file simulator.h.

References `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::handler`, `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::t1`, `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::t2`, `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::t3`, and `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::t4`.

7.31.4 Member Data Documentation

7.31.4.1 `template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> void(* Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::handler)(U1, U2, U3, U4)`

Referenced by `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::CallHandler()`.

7.31.4.2 `template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T1 Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::t1`

Definition at line 251 of file simulator.h.

Referenced by `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::CallHandler()`.

7.31.4.3 `template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T2 Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::t2`

Definition at line 252 of file simulator.h.

Referenced by `Event4Stat<U1, T1, U2, T2, U3, T3, U4, T4>::CallHandler()`.

**7.31.4.4 template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T3
Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >::t3**

Definition at line 253 of file simulator.h.

Referenced by Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

**7.31.4.5 template<typename U1, typename T1, typename U2, typename T2, typename U3, typename T3, typename U4, typename T4> T4
Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >::t4**

Definition at line 254 of file simulator.h.

Referenced by Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >::CallHandler().

The documentation for this class was generated from the following file:

- **simulator.h**

7.32 event_less Class Reference

```
#include <simulator.h>
```

Public Member Functions

- **event_less ()**
- **bool operator() (EventBase *const &l, const EventBase *const &r) const**

7.32.1 Detailed Description

Definition at line 269 of file simulator.h.

7.32.2 Constructor & Destructor Documentation

7.32.2.1 event_less::event_less () [inline]

Definition at line 272 of file simulator.h.

7.32.3 Member Function Documentation

7.32.3.1 bool event_less::operator() (EventBase *const &l, const EventBase *const &r) const [inline]

Definition at line 273 of file simulator.h.

References EventBase::time, and EventBase::uid.

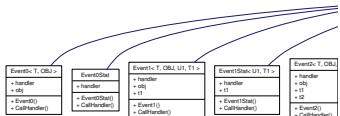
The documentation for this class was generated from the following file:

- **simulator.h**

7.33 EventBase Class Reference

```
#include <simulator.h>
```

Inheritance diagram for EventBase:



Public Member Functions

- **EventBase** (double t)
- **EventBase** (double t, int u)
- virtual void **CallHandler** ()=0

Public Attributes

- double **time**
- int **uid**

Static Public Attributes

- static int **nextUID** = 0

7.33.1 Detailed Description

Definition at line 24 of file simulator.h.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 EventBase::EventBase (double t) [inline]

Definition at line 27 of file simulator.h.

7.33.2.2 EventBase::EventBase (double t, int u) [inline]

Definition at line 28 of file simulator.h.

7.33.3 Member Function Documentation

7.33.3.1 virtual void EventBase::CallHandler () [pure virtual]

Implemented in **EventId** (p. 108), **Event0< T, OBJ >** (p. 83), **Event1< T, OBJ, U1, T1 >** (p. 86), **Event2< T, OBJ, U1, T1, U2, T2 >** (p. 90), **Event3< T, OBJ, U1, T1, U2, T2 >** (p. 90).

T2, U3, T3 > (p. 95), **Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >** (p. 100), **Event0Stat** (p. 85), **Event1Stat< U1, T1 >** (p. 88), **Event2Stat< U1, T1, U2, T2 >** (p. 92), **Event3Stat< U1, T1, U2, T2, U3, T3 >** (p. 97), and **Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >** (p. 103).

Referenced by Simulator::Run().

Here is the caller graph for this function:



7.33.4 Member Data Documentation

7.33.4.1 int EventBase::nextUID = 0 [static]

Definition at line 33 of file simulator.h.

7.33.4.2 double EventBase::time

Definition at line 31 of file simulator.h.

Referenced by event_less::operator()(), and Simulator::Run().

7.33.4.3 int EventBase::uid

Definition at line 32 of file simulator.h.

Referenced by event_less::operator()(), and Simulator::Schedule().

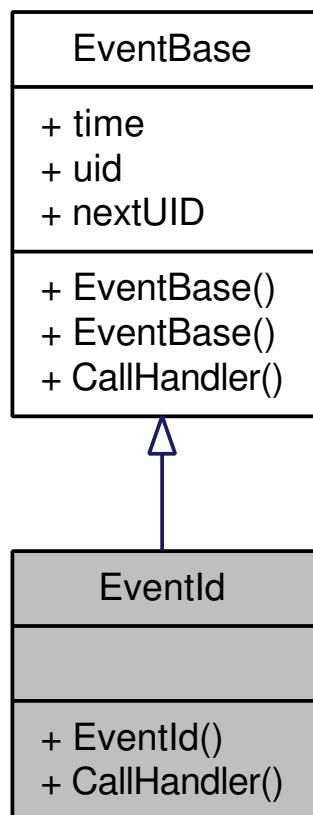
The documentation for this class was generated from the following files:

- **simulator.h**
- **simulator.cc**

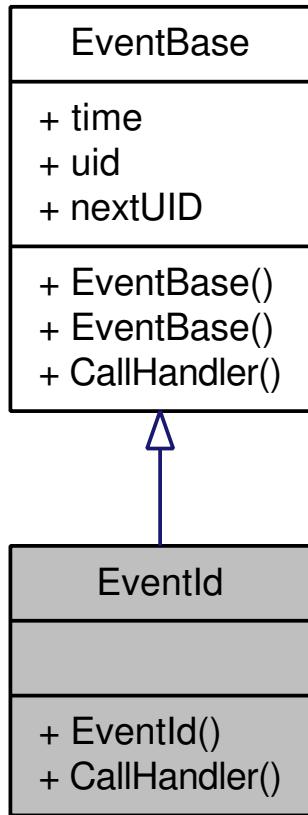
7.34 EventId Class Reference

```
#include <simulator.h>
```

Inheritance diagram for EventId:



Collaboration diagram for EventId:



Public Member Functions

- **EventId** (double t, int u)
- void **CallHandler** ()

7.34.1 Detailed Description

Definition at line 39 of file simulator.h.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 EventId::EventId (double t, int u) [inline]

Definition at line 42 of file simulator.h.

7.34.3 Member Function Documentation

7.34.3.1 void EventId::CallHandler () [inline, virtual]

Implements **EventBase** (p. 106).

Definition at line 43 of file simulator.h.

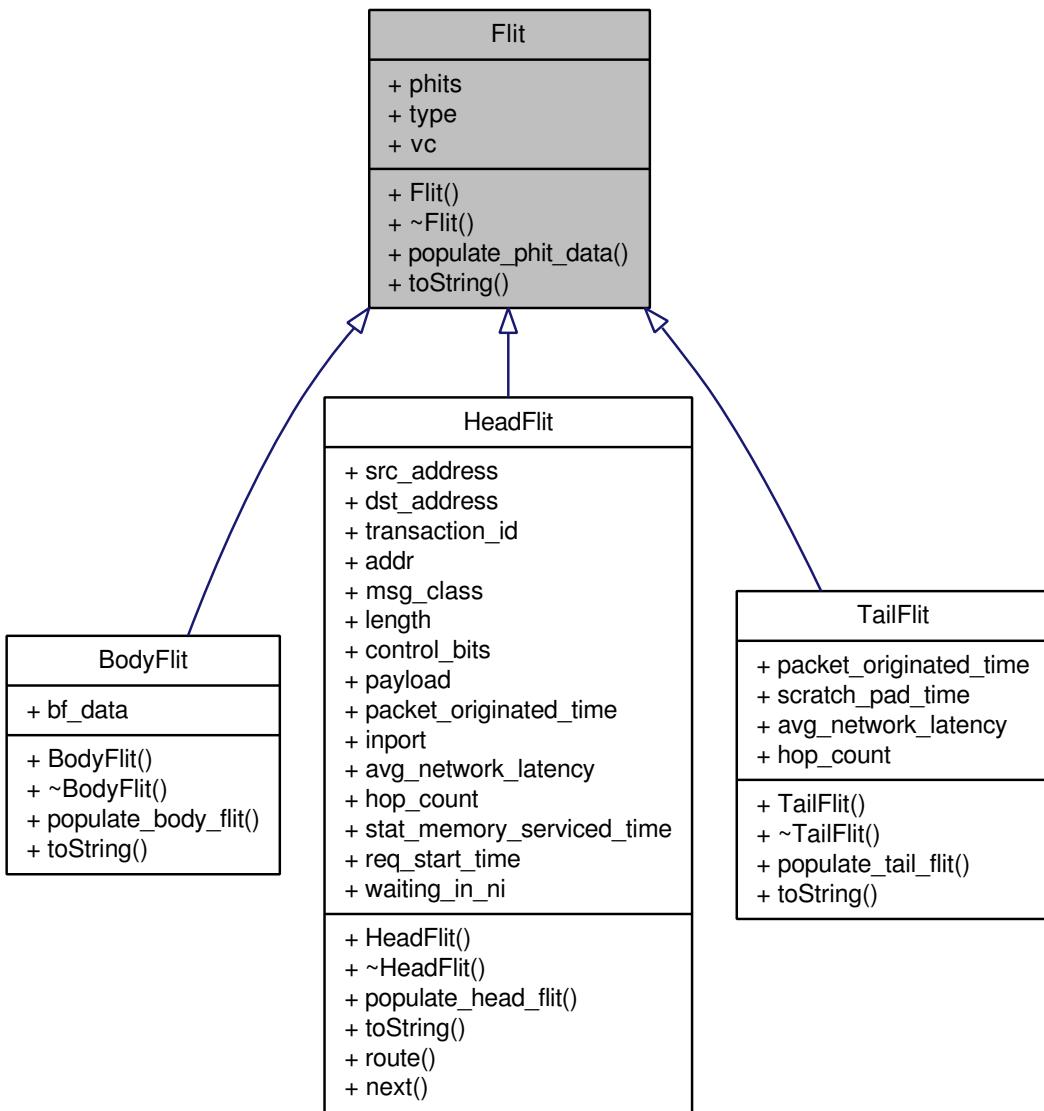
The documentation for this class was generated from the following file:

- **simulator.h**

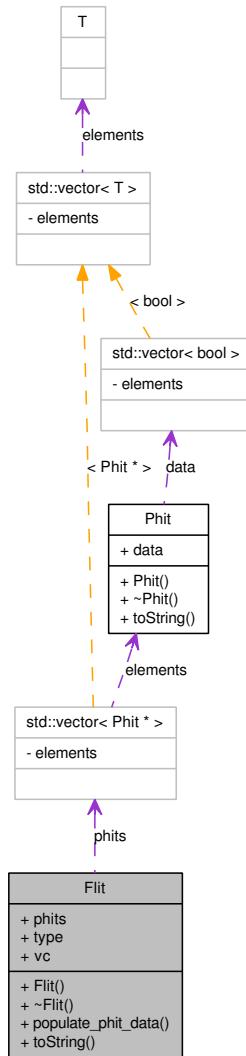
7.35 Flit Class Reference

```
#include <flit.h>
```

Inheritance diagram for Flit:



Collaboration diagram for Flit:



Public Member Functions

- `Flit ()`
- `~Flit ()`
- `void populate_phit_data (vector< bool > *c)`
- `string toString () const`

Public Attributes

- `vector< Phit * > phits`
- `flit_type type`
- `uint vc`

7.35.1 Detailed Description

Definition at line 61 of file flit.h.

7.35.2 Constructor & Destructor Documentation

7.35.2.1 Flit::Flit ()

Definition at line 52 of file flit.cc.

7.35.2.2 Flit::~Flit ()

Definition at line 71 of file flit.cc.

7.35.3 Member Function Documentation

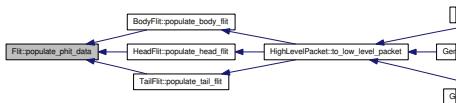
7.35.3.1 void Flit::populate_phit_data (vector< bool > * c)

Definition at line 57 of file flit.cc.

References max_phy_link_bits, and phits.

Referenced by BodyFlit::populate_body_flit(), HeadFlit::populate_head_flit(), and TailFlit::populate_tail_flit().

Here is the caller graph for this function:



7.35.3.2 string Flit::toString () const

Reimplemented in **HeadFlit** (p. 216), **BodyFlit** (p. 46), and **TailFlit** (p. 357).

Definition at line 79 of file flit.cc.

References phits, and type.

7.35.4 Member Data Documentation

7.35.4.1 vector<Phit*> Flit::phits

Definition at line 67 of file flit.h.

Referenced by populate_phit_data(), TailFlit::toString(), BodyFlit::toString(), HeadFlit::toString(), toString(), BodyFlit::~BodyFlit(), HeadFlit::~HeadFlit(), and TailFlit::~TailFlit().

7.35.4.2 flit_type Flit::type

Definition at line 70 of file flit.h.

Referenced by LowLevelPacket::add(), BodyFlit::BodyFlit(), GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), MCFrontEnd::handle_out_arbitrate_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericInterface::handle_tick_event(), HeadFlit::HeadFlit(), GenericPortArbiter::pull_winner(), GenericRC::push(), TailFlit::TailFlit(), TailFlit::toString(), BodyFlit::toString(), and toString().

7.35.4.3 uint Flit::vc

Definition at line 71 of file flit.h.

Referenced by LowLevelPacket::add(), GenericRC::push(), HighLevelPacket::to_low_level_packet(), and HeadFlit::toString().

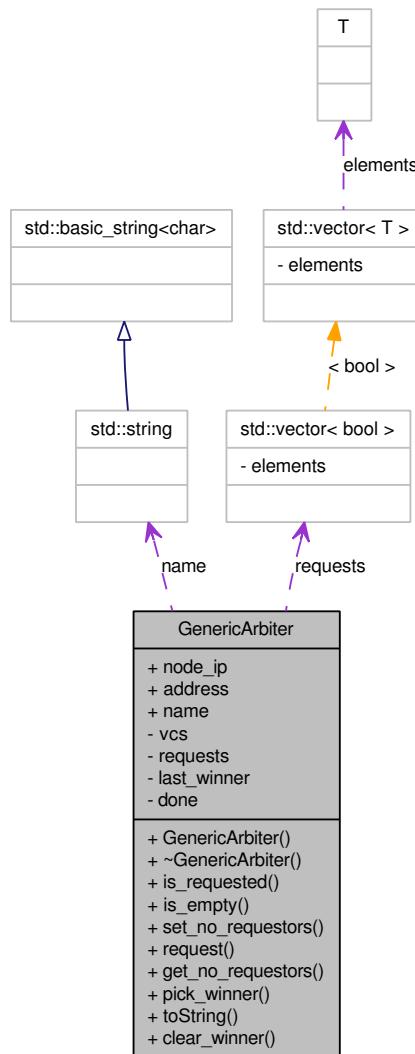
The documentation for this class was generated from the following files:

- **flit.h**
- **flit.cc**

7.36 GenericArbiter Class Reference

```
#include <genericArbiter.h>
```

Collaboration diagram for GenericArbiter:



Public Member Functions

- `GenericArbiter ()`
- `~GenericArbiter ()`
- `bool is_requested (uint ch)`
- `bool is_empty ()`
- `void set_no_requestors (uint ch)`
- `void request (uint ch)`
- `uint get_no_requestors ()`
- `uint pick_winner ()`
- `string toString () const`
- `void clear_winner ()`

Public Attributes

- `uint node_ip`
- `uint address`
- `string name`

Private Attributes

- `uint vcs`
- `vector< bool > requests`
- `uint last_winner`
- `bool done`

7.36.1 Detailed Description

Definition at line 31 of file genericArbiter.h.

7.36.2 Constructor & Destructor Documentation

7.36.2.1 GenericArbiter::GenericArbiter ()

Definition at line 23 of file genericArbiter.cc.

References `address`, `done`, `last_winner`, `name`, and `node_ip`.

7.36.2.2 GenericArbiter::~GenericArbiter ()

Definition at line 32 of file genericArbiter.cc.

7.36.3 Member Function Documentation

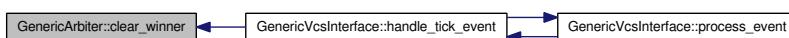
7.36.3.1 void GenericArbiter::clear_winner ()

Definition at line 84 of file genericArbiter.cc.

References `done`, `last_winner`, and `requests`.

Referenced by `GenericVcsInterface::handle_tick_event()`.

Here is the caller graph for this function:



7.36.3.2 uint GenericArbiter::get_no_requestors ()

Definition at line 73 of file genericArbiter.cc.

References `requests`, and `vcs`.

7.36.3.3 bool GenericArbiter::is_empty ()

Definition at line 62 of file genericArbiter.cc.

References requests, and vcs.

Referenced by GenericVcsInterface::handle_tick_event().

Here is the caller graph for this function:



7.36.3.4 bool GenericArbiter::is_requested (uint ch)

Definition at line 55 of file genericArbiter.cc.

References requests.

Referenced by GenericVcsInterface::handle_tick_event().

Here is the caller graph for this function:



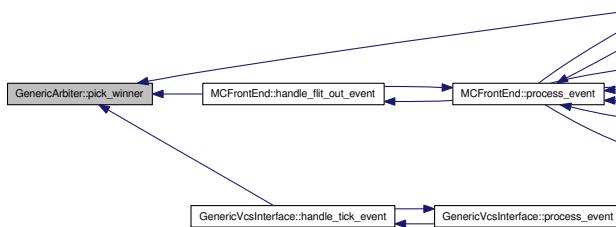
7.36.3.5 uint GenericArbiter::pick_winner ()

Definition at line 92 of file genericArbiter.cc.

References done, last_winner, and requests.

Referenced by MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), and GenericVcsInterface::handle_tick_event().

Here is the caller graph for this function:



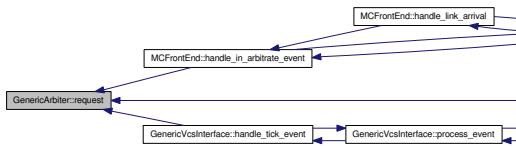
7.36.3.6 void GenericArbiter::request (uint ch)

Definition at line 47 of file genericArbiter.cc.

References done, and requests.

Referenced by MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_out_arbitrate_event(), and GenericVcsInterface::handle_tick_event().

Here is the caller graph for this function:



7.36.3.7 void GenericArbiter::set_no_requestors (uint ch)

Definition at line 38 of file genericArbiter.cc.

References requests, and vcs.

Referenced by `GenericVcsInterface::setup()`.

Here is the caller graph for this function:



7.36.3.8 string GenericArbiter::toString () const

Definition at line 120 of file genericArbiter.cc.

References last_winner, and requests.

Referenced by `MCFrontEnd::toString()`.

Here is the caller graph for this function:



7.36.4 Member Data Documentation

7.36.4.1 uint GenericArbiter::address

Definition at line 37 of file genericArbiter.h.

Referenced by `GenericArbiter()`.

7.36.4.2 bool GenericArbiter::done [private]

Definition at line 54 of file genericArbiter.h.

Referenced by `clear_winner()`, `GenericArbiter()`, `pick_winner()`, and `request()`.

7.36.4.3 uint GenericArbiter::last_winner [private]

Definition at line 53 of file genericArbiter.h.

Referenced by clear_winner(), GenericArbiter(), pick_winner(), and toString().

7.36.4.4 **string GenericArbiter::name**

Definition at line 38 of file genericArbiter.h.

Referenced by GenericArbiter().

7.36.4.5 **uint GenericArbiter::node_ip**

Definition at line 36 of file genericArbiter.h.

Referenced by GenericArbiter().

7.36.4.6 **vector< bool > GenericArbiter::requests [private]**

Definition at line 52 of file genericArbiter.h.

Referenced by clear_winner(), get_no_requestors(), is_empty(), is_requested(), pick_winner(), request(), set_no_requestors(), and toString().

7.36.4.7 **uint GenericArbiter::vcs [private]**

Definition at line 51 of file genericArbiter.h.

Referenced by get_no_requestors(), is_empty(), and set_no_requestors().

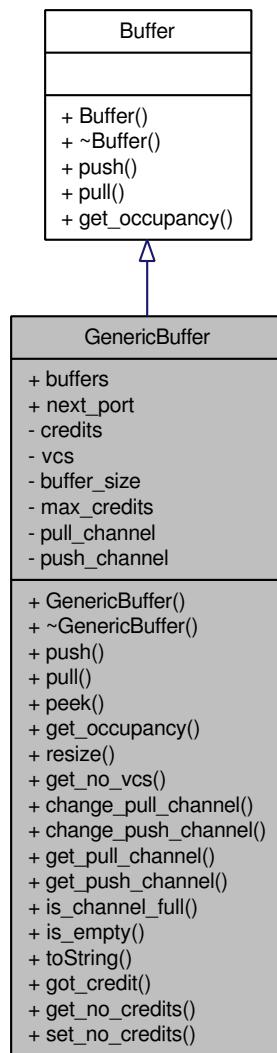
The documentation for this class was generated from the following files:

- **genericArbiter.h**
- **genericArbiter.cc**

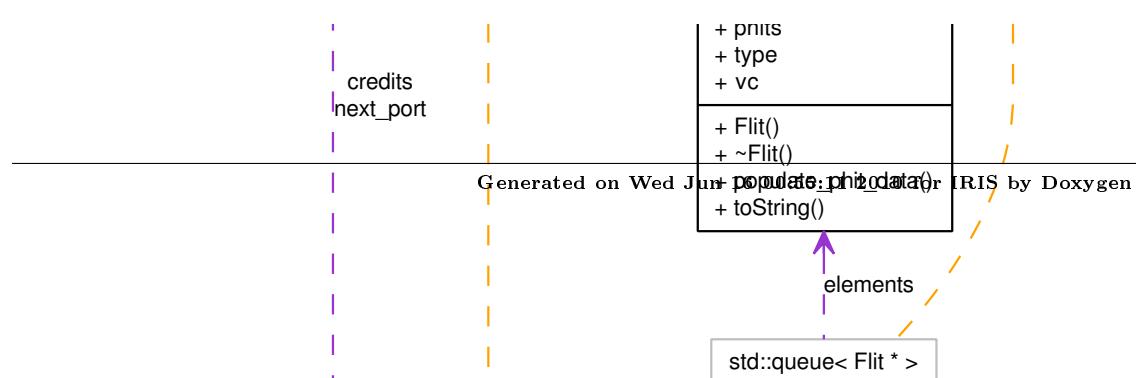
7.37 GenericBuffer Class Reference

```
#include <genericBuffer.h>
```

Inheritance diagram for GenericBuffer:



Collaboration diagram for GenericBuffer:



Public Member Functions

- **GenericBuffer ()**
- **~GenericBuffer ()**
- **void push (Flit *f)**
- **Flit * pull ()**
- **Flit * peek ()**
- **uint get_occupancy (uint ch) const**
- **void resize (uint vcs, uint buffer_size)**
- **uint get_no_vcs () const**
- **void change_pull_channel (uint ch)**
- **void change_push_channel (uint ch)**
- **uint get_pull_channel () const**
- **uint get_push_channel () const**
- **bool is_channel_full (uint ch) const**
- **bool is_empty (uint ch) const**
- **string toString () const**
- **void got_credit (uint ch)**
- **uint get_no_credits (uint ch) const**
- **void set_no_credits (uint no)**

Public Attributes

- **vector< queue< Flit * > > buffers**
- **vector< int > next_port**

Private Attributes

- **vector< int > credits**
- **uint vcs**
- **uint buffer_size**
- **uint max_credits**
- **uint pull_channel**
- **uint push_channel**

7.37.1 Detailed Description

Description: =====

Definition at line 32 of file genericBuffer.h.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 GenericBuffer::GenericBuffer ()

Definition at line 23 of file genericBuffer.cc.

References pull_channel, and push_channel.

7.37.2.2 GenericBuffer::~GenericBuffer ()

Definition at line 29 of file genericBuffer.cc.

7.37.3 Member Function Documentation

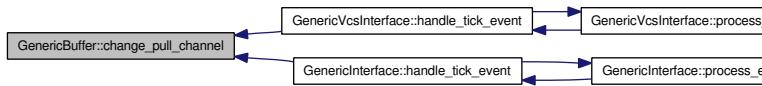
7.37.3.1 void GenericBuffer::change_pull_channel (uint ch)

Definition at line 97 of file genericBuffer.cc.

References pull_channel, and vcs.

Referenced by GenericVcsInterface::handle_tick_event(), and GenericInterface::handle_tick_event().

Here is the caller graph for this function:



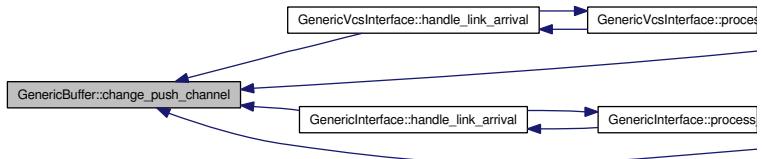
7.37.3.2 void GenericBuffer::change_push_channel (uint ch)

Definition at line 105 of file genericBuffer.cc.

References push_channel.

Referenced by GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), GenericVcsInterface::handle_tick_event(), and GenericInterface::handle_tick_event().

Here is the caller graph for this function:



7.37.3.3 uint GenericBuffer::get_no_credits (uint ch) const

7.37.3.4 uint GenericBuffer::get_no_vcs () const

Definition at line 89 of file genericBuffer.cc.

References buffers.

7.37.3.5 uint GenericBuffer::get_occupancy (uint ch) const [virtual]

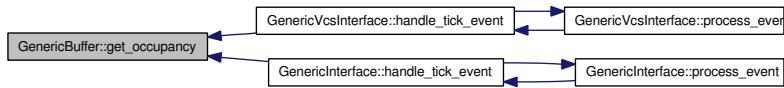
Implements **Buffer** (p. 47).

Definition at line 66 of file genericBuffer.cc.

References buffers.

Referenced by `GenericVcsInterface::handle_tick_event()`, and `GenericInterface::handle_tick_event()`.

Here is the caller graph for this function:



7.37.3.6 `uint GenericBuffer::get_pull_channel () const`

Definition at line 113 of file genericBuffer.cc.

References pull_channel.

7.37.3.7 `uint GenericBuffer::get_push_channel () const`

Definition at line 119 of file genericBuffer.cc.

References push_channel.

7.37.3.8 `void GenericBuffer::got_credit (uint ch)`

7.37.3.9 `bool GenericBuffer::is_channel_full (uint ch) const`

Definition at line 125 of file genericBuffer.cc.

References BUFFER_SIZE, and buffers.

7.37.3.10 `bool GenericBuffer::is_empty (uint ch) const`

Definition at line 132 of file genericBuffer.cc.

References buffers.

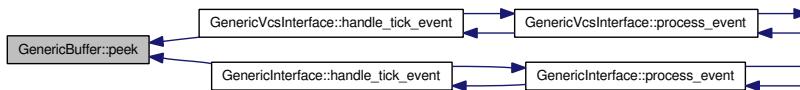
7.37.3.11 `Flit * GenericBuffer::peek ()`

Definition at line 54 of file genericBuffer.cc.

References buffers, and pull_channel.

Referenced by `GenericVcsInterface::handle_tick_event()`, and `GenericInterface::handle_tick_event()`.

Here is the caller graph for this function:



7.37.3.12 Flit * GenericBuffer::pull () [virtual]

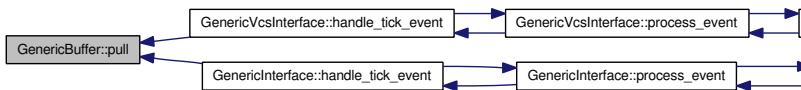
Implements **Buffer** (p. 47).

Definition at line 41 of file genericBuffer.cc.

References buffers, and pull_channel.

Referenced by GenericVcsInterface::handle_tick_event(), and GenericInterface::handle_tick_event().

Here is the caller graph for this function:



7.37.3.13 void GenericBuffer::push (Flit * f) [virtual]

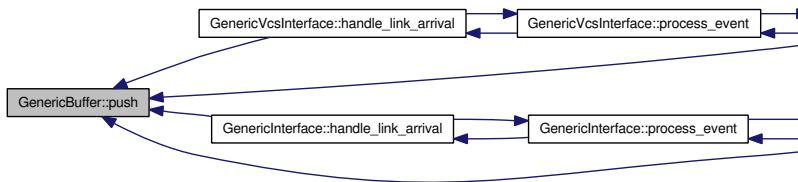
Implements **Buffer** (p. 47).

Definition at line 34 of file genericBuffer.cc.

References buffers, and push_channel.

Referenced by GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), GenericVcsInterface::handle_tick_event(), and GenericInterface::handle_tick_event().

Here is the caller graph for this function:



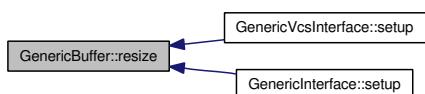
7.37.3.14 void GenericBuffer::resize (uint vcs, uint buffer_size)

Definition at line 72 of file genericBuffer.cc.

References buffer_size, buffers, and vcs.

Referenced by GenericVcsInterface::setup(), and GenericInterface::setup().

Here is the caller graph for this function:



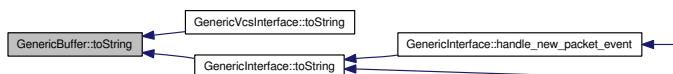
7.37.3.15 void GenericBuffer::set_no_credits (uint no)**7.37.3.16 string GenericBuffer::toString () const**

Definition at line 138 of file genericBuffer.cc.

References buffer_size, and buffers.

Referenced by GenericVcsInterface::toString(), and GenericInterface::toString().

Here is the caller graph for this function:

**7.37.4 Member Data Documentation****7.37.4.1 uint GenericBuffer::buffer_size [private]**

Definition at line 63 of file genericBuffer.h.

Referenced by `resize()`, and `toString()`.

7.37.4.2 vector< queue<Flit*> > GenericBuffer::buffers

Definition at line 55 of file genericBuffer.h.

Referenced by `get_no_vcs()`, `get_occupancy()`, `is_channel_full()`, `is_empty()`, `peek()`, `pull()`, `push()`, `resize()`, and `toString()`.

7.37.4.3 vector< int > GenericBuffer::credits [private]

Definition at line 61 of file genericBuffer.h.

7.37.4.4 uint GenericBuffer::max_credits [private]

Definition at line 64 of file genericBuffer.h.

7.37.4.5 vector< int > GenericBuffer::next_port

Definition at line 56 of file genericBuffer.h.

7.37.4.6 uint GenericBuffer::pull_channel [private]

Definition at line 65 of file genericBuffer.h.

Referenced by `change_pull_channel()`, `GenericBuffer()`, `get_pull_channel()`, `peek()`, and `pull()`.

7.37.4.7 uint GenericBuffer::push_channel [private]

Definition at line 66 of file genericBuffer.h.

Referenced by change_push_channel(), GenericBuffer(), get_push_channel(), and push().

7.37.4.8 uint GenericBuffer::vcs [private]

Definition at line 62 of file genericBuffer.h.

Referenced by change_pull_channel(), and resize().

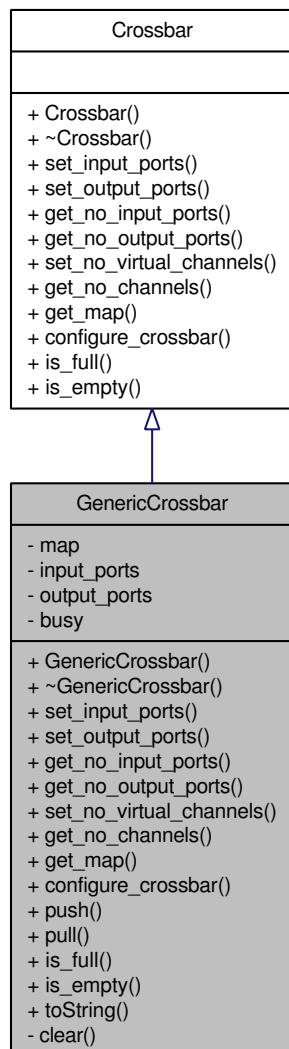
The documentation for this class was generated from the following files:

- **genericBuffer.h**
- **genericBuffer.cc**

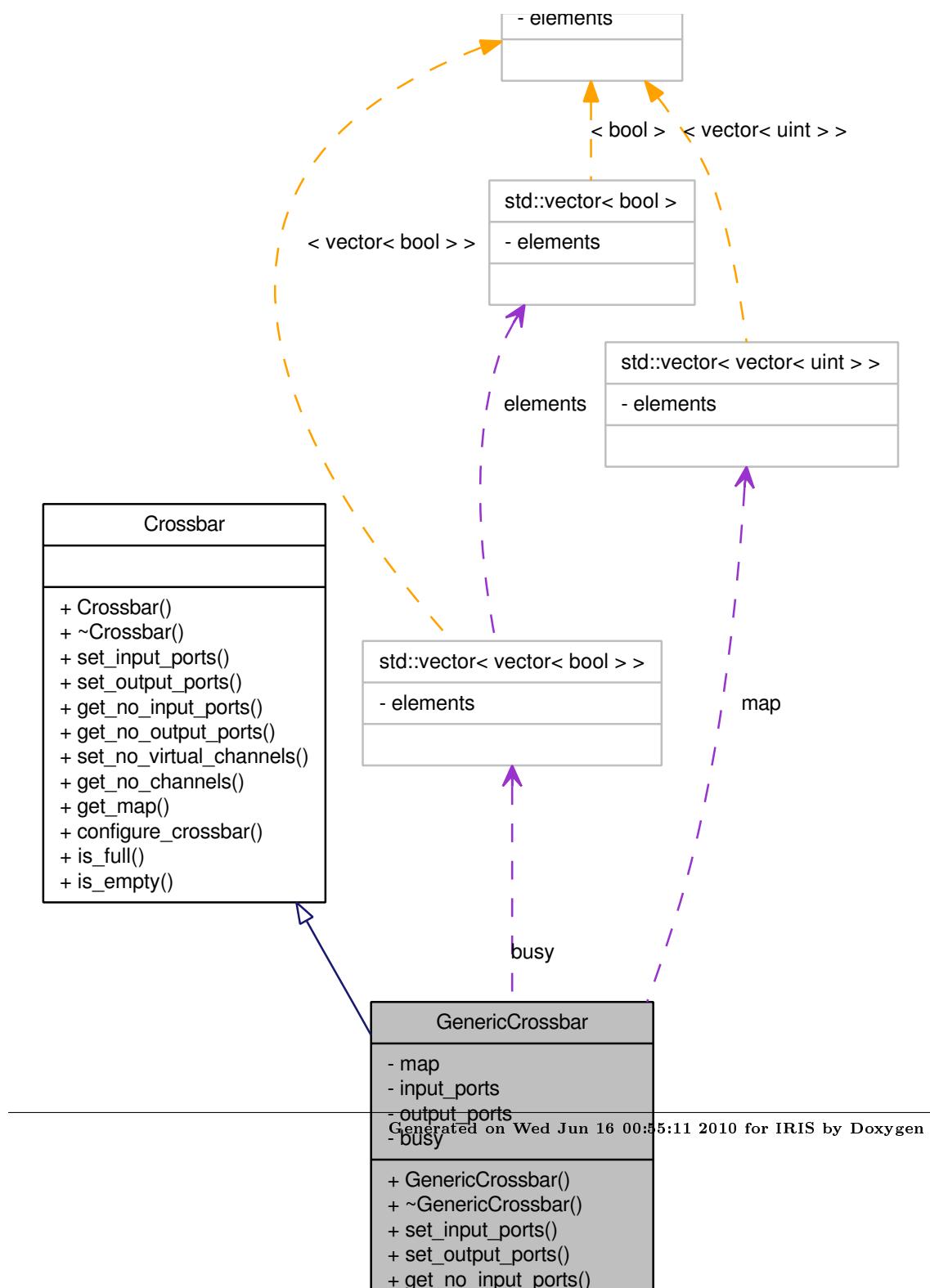
7.38 GenericCrossbar Class Reference

```
#include <genericCrossbar.h>
```

Inheritance diagram for GenericCrossbar:



Collaboration diagram for GenericCrossbar:



Classes

- class **CrossbarUnit**

Public Member Functions

- **GenericCrossbar ()**
- **~GenericCrossbar ()**
- **void set_input_ports (uint ports)**
- **void set_output_ports (uint ports)**
- **uint get_no_input_ports ()**
- **uint get_no_output_ports ()**
- **void set_no_virtual_channels (uint no)**
- **uint get_no_channels ()**
- **uint get_map (uint in_port, uint ch)**
- **void configure_crossbar (uint in_port, uint out_port, uint ch)**
- **void push (uint in_port, uint ch)**
- **void pull (uint out_port, uint ch)**
- **bool is_full (uint in_port, uint ch)**
- **bool is_empty (uint out_port, uint ch)**
- **string toString () const**

Private Member Functions

- **void clear ()**

Private Attributes

- **vector< vector< uint > > map**
- **uint input_ports**
- **uint output_ports**
- **vector< vector< bool > > busy**

7.38.1 Detailed Description

Definition at line 33 of file genericCrossbar.h.

7.38.2 Constructor & Destructor Documentation

7.38.2.1 GenericCrossbar::GenericCrossbar ()

Definition at line 25 of file genericCrossbar.cc.

References `input_ports`, and `output_ports`.

7.38.2.2 GenericCrossbar::~GenericCrossbar ()

Definition at line 31 of file genericCrossbar.cc.

7.38.3 Member Function Documentation

7.38.3.1 void GenericCrossbar::clear () [private]

Definition at line 37 of file genericCrossbar.cc.

7.38.3.2 void GenericCrossbar::configure_crossbar (uint *in_port*, uint *out_port*, uint *ch*) [virtual]

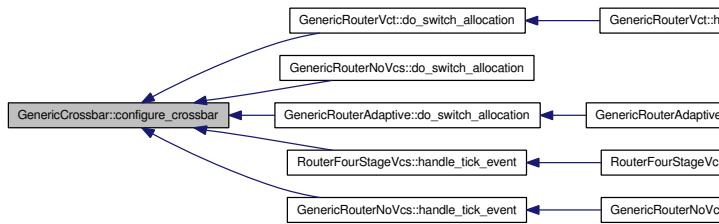
Implements **Crossbar** (p. 70).

Definition at line 112 of file genericCrossbar.cc.

References busy, and map.

Referenced by GenericRouterVct::do_switch_allocation(), GenericRouterNoVcs::do_switch_allocation(), GenericRouterAdaptive::do_switch_allocation(), RouterFourStageVcs::handle_tick_event(), and GenericRouterNoVcs::handle_tick_event().

Here is the caller graph for this function:



7.38.3.3 uint GenericCrossbar::get_map (uint *in_port*, uint *ch*) [virtual]

Implements **Crossbar** (p. 70).

Definition at line 100 of file genericCrossbar.cc.

References input_ports, and map.

7.38.3.4 uint GenericCrossbar::get_no_channels () [virtual]

Implements **Crossbar** (p. 71).

Definition at line 94 of file genericCrossbar.cc.

References map.

7.38.3.5 uint GenericCrossbar::get_no_input_ports () [virtual]

Implements **Crossbar** (p. 71).

Definition at line 56 of file genericCrossbar.cc.

References input_ports.

7.38.3.6 uint GenericCrossbar::get_no_output_ports () [virtual]

Implements **Crossbar** (p. 71).

Definition at line 62 of file genericCrossbar.cc.

References output_ports.

7.38.3.7 bool GenericCrossbar::is_empty (uint out_port, uint ch) [virtual]

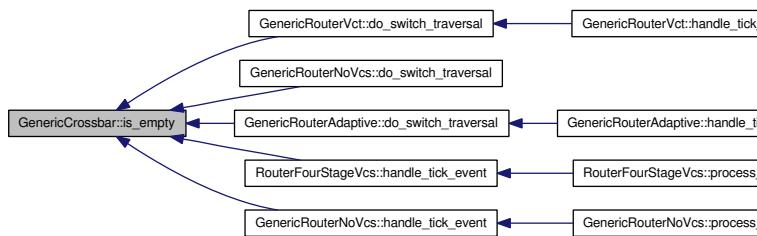
Implements **Crossbar** (p. 71).

Definition at line 144 of file genericCrossbar.cc.

References busy.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), RouterFourStageVcs::handle_tick_event(), and GenericRouterNoVcs::handle_tick_event().

Here is the caller graph for this function:

**7.38.3.8 bool GenericCrossbar::is_full (uint in_port, uint ch) [virtual]**

Implements **Crossbar** (p. 71).

Definition at line 134 of file genericCrossbar.cc.

References busy, and map.

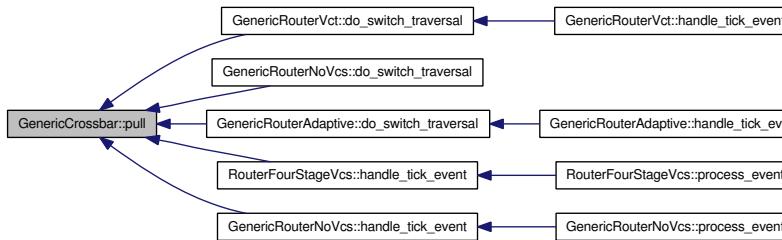
7.38.3.9 void GenericCrossbar::pull (uint out_port, uint ch)

Definition at line 127 of file genericCrossbar.cc.

References busy.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), RouterFourStageVcs::handle_tick_event(), and GenericRouterNoVcs::handle_tick_event().

Here is the caller graph for this function:



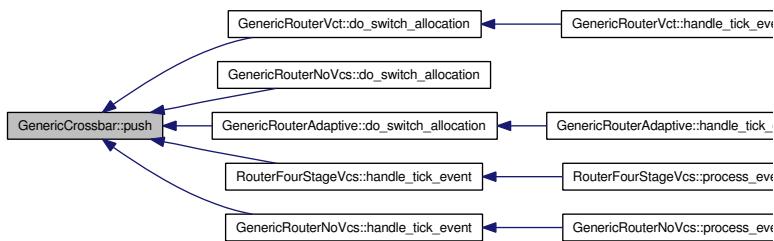
7.38.3.10 void GenericCrossbar::push (uint *in_port*, uint *ch*)

Definition at line 120 of file genericCrossbar.cc.

References busy.

Referenced by GenericRouterVct::do_switch_allocation(), GenericRouterNoVcs::do_switch_allocation(), GenericRouterAdaptive::do_switch_allocation(), RouterFourStageVcs::handle_tick_event(), and GenericRouterNoVcs::handle_tick_event().

Here is the caller graph for this function:



7.38.3.11 void GenericCrossbar::set_input_ports (uint *ports*) [virtual]

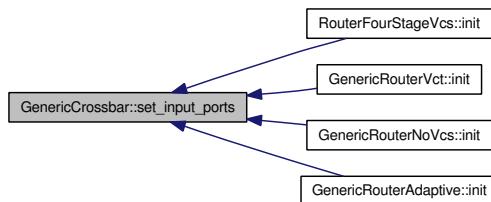
Implements **Crossbar** (p. 71).

Definition at line 43 of file genericCrossbar.cc.

References input_ports.

Referenced by RouterFourStageVcs::init(), GenericRouterVct::init(), GenericRouterNoVcs::init(), and GenericRouterAdaptive::init().

Here is the caller graph for this function:



7.38.3.12 void GenericCrossbar::set_no_virtual_channels (uint no) [virtual]

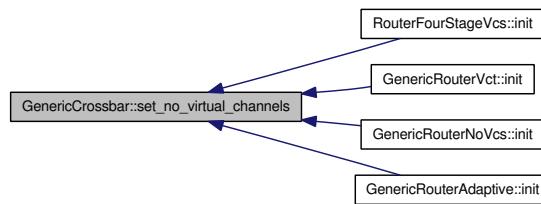
Implements **Crossbar** (p. 71).

Definition at line 68 of file genericCrossbar.cc.

References busy, input_ports, map, and output_ports.

Referenced by RouterFourStageVcs::init(), GenericRouterVct::init(), GenericRouterNoVcs::init(), and GenericRouterAdaptive::init().

Here is the caller graph for this function:



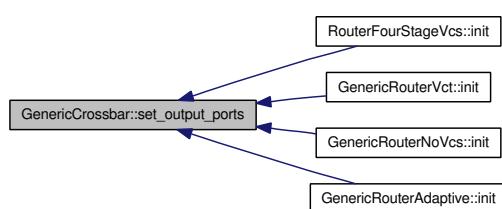
7.38.3.13 void GenericCrossbar::set_output_ports (uint ports) [virtual]

Implements **Crossbar** (p. 71).

Definition at line 49 of file genericCrossbar.cc.

Referenced by RouterFourStageVcs::init(), GenericRouterVct::init(), GenericRouterNoVcs::init(), and GenericRouterAdaptive::init().

Here is the caller graph for this function:



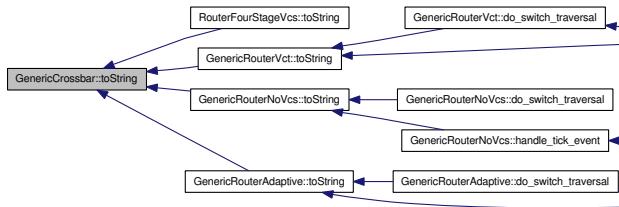
7.38.3.14 string GenericCrossbar::toString () const

Definition at line 156 of file genericCrossbar.cc.

References map, and output_ports.

Referenced by RouterFourStageVcs::toString(), GenericRouterVct::toString(), GenericRouterNoVcs::toString(), and GenericRouterAdaptive::toString().

Here is the caller graph for this function:



7.38.4 Member Data Documentation

7.38.4.1 `vector< vector<bool> > GenericCrossbar::busy [private]`

Definition at line 76 of file genericCrossbar.h.

Referenced by `configure_crossbar()`, `is_empty()`, `is_full()`, `pull()`, `push()`, and `set_no_virtual_channels()`.

7.38.4.2 `uint GenericCrossbar::input_ports [private]`

Definition at line 56 of file genericCrossbar.h.

Referenced by `GenericCrossbar()`, `get_map()`, `get_no_input_ports()`, `set_input_ports()`, and `set_no_virtual_channels()`.

7.38.4.3 `vector< vector<uint> > GenericCrossbar::map [private]`

Definition at line 55 of file genericCrossbar.h.

Referenced by `configure_crossbar()`, `get_map()`, `get_no_channels()`, `is_full()`, `set_no_virtual_channels()`, and `toString()`.

7.38.4.4 `uint GenericCrossbar::output_ports [private]`

Definition at line 57 of file genericCrossbar.h.

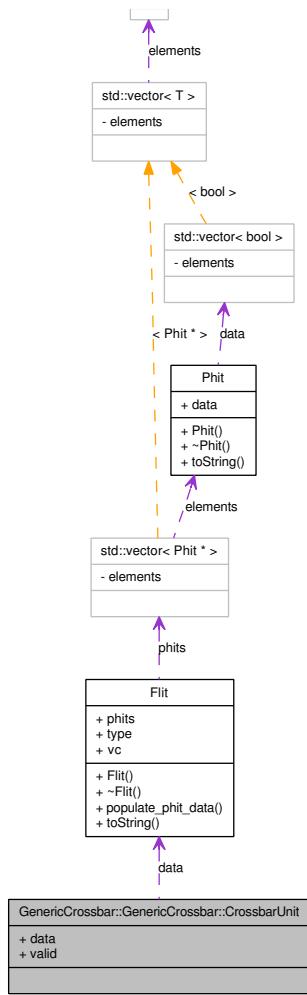
Referenced by `GenericCrossbar()`, `get_no_output_ports()`, `set_no_virtual_channels()`, and `toString()`.

The documentation for this class was generated from the following files:

- `genericCrossbar.h`
- `genericCrossbar.cc`

7.39 GenericCrossbar::GenericCrossbar::CrossbarUnit Class Reference

Collaboration diagram for GenericCrossbar::GenericCrossbar::CrossbarUnit:



Public Attributes

- **Flit * data**
- **bool valid**

7.39.1 Detailed Description

Definition at line 65 of file genericCrossbar.h.

7.39.2 Member Data Documentation

7.39.2.1 Flit* GenericCrossbar::GenericCrossbar::CrossbarUnit::data

Definition at line 68 of file genericCrossbar.h.

7.39.2.2 bool GenericCrossbar::GenericCrossbar::CrossbarUnit::valid

Definition at line 69 of file genericCrossbar.h.

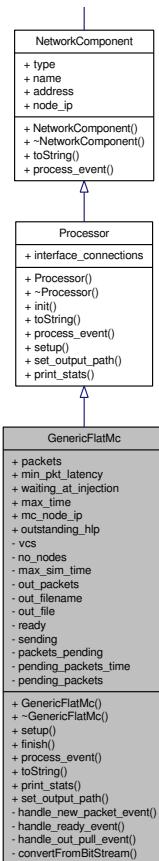
The documentation for this class was generated from the following file:

- `genericCrossbar.h`

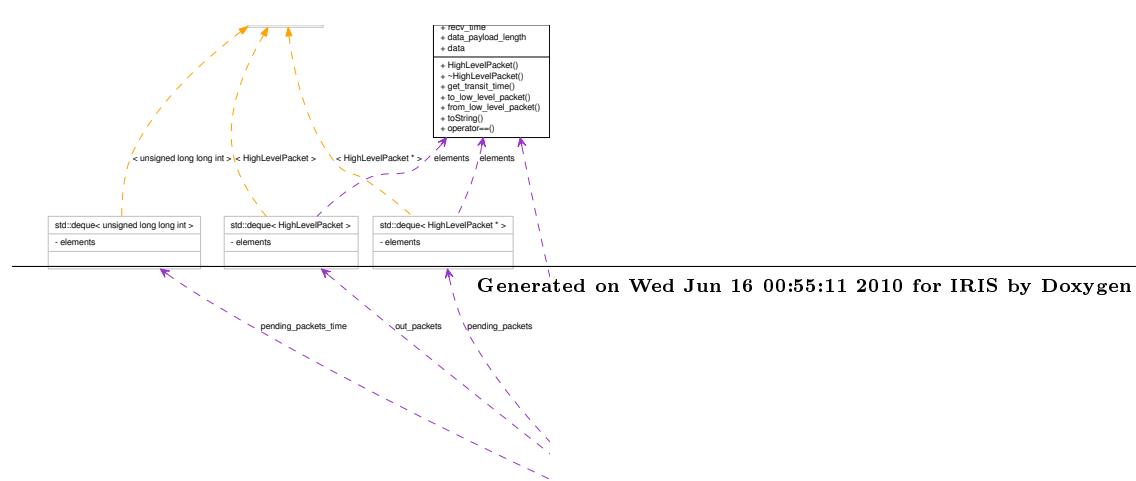
7.40 GenericFlatMc Class Reference

```
#include <genericFlatMc.h>
```

Inheritance diagram for GenericFlatMc:



Collaboration diagram for GenericFlatMc:



Public Member Functions

- `GenericFlatMc ()`
- `~GenericFlatMc ()`
- `void setup (uint no_nodes, uint vcs, uint max_sim_time)`
- `void finish ()`
- `void process_event (IrisEvent *e)`
- `string toString () const`
- `string print_stats () const`
- `void set_output_path (string outpath)`

Public Attributes

- `unsigned int packets`
- `double min_pkt_latency`
- `unsigned long long int waiting_at_injection`
- `unsigned long long int max_time`
- `vector< uint > mc_node_ip`
- `HighLevelPacket * outstanding_hlp`

Private Member Functions

- `void handle_new_packet_event (IrisEvent *e)`
- `void handle_ready_event (IrisEvent *e)`
- `void handle_out_pull_event (IrisEvent *e)`
- `void convertFromBitStream (Request *req, HighLevelPacket *hlp)`

Private Attributes

- `uint vcs`
- `uint no_nodes`
- `unsigned long long int max_sim_time`
- `deque< HighLevelPacket > out_packets`
- `string out_filename`
- `ofstream out_file`
- `vector< bool > ready`
- `bool sending`
- `long int packets_pending`
- `deque< unsigned long long int > pending_packets_time`
- `deque< HighLevelPacket * > pending_packets`

7.40.1 Detailed Description

Definition at line 22 of file genericFlatMc.h.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 GenericFlatMc::GenericFlatMc ()

Definition at line 8 of file genericFlatMc.cc.

References Processor::interface_connections, NetworkComponent::name, and sending.

7.40.2.2 GenericFlatMc::~GenericFlatMc ()

Definition at line 15 of file genericFlatMc.cc.

References out_file, pending_packets, and pending_packets_time.

7.40.3 Member Function Documentation

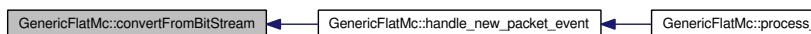
7.40.3.1 void GenericFlatMc::convertFromBitStream (Request * *req*, HighLevelPacket * *hlp*) [private]

Definition at line 318 of file genericFlatMc.cc.

References Request::address, Request::cmdType, HighLevelPacket::data, NETWORK_ADDRESS_BITS, NETWORK_COMMAND_BITS, NETWORK_THREADID_BITS, and Request::threadId.

Referenced by handle_new_packet_event().

Here is the caller graph for this function:



7.40.3.2 void GenericFlatMc::finish ()

Definition at line 69 of file genericFlatMc.cc.

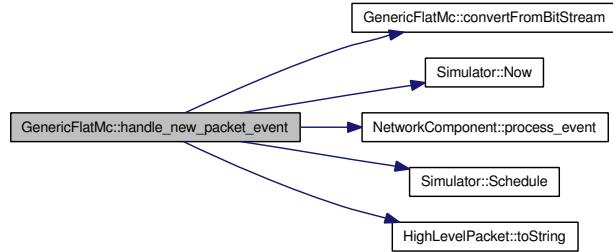
7.40.3.3 void GenericFlatMc::handle_new_packet_event (IrisEvent * *e*) [private]

Definition at line 95 of file genericFlatMc.cc.

References CACHE_WRITEBACK, Request::cmdType, convertFromBitStream(), IrisEvent::event_data, Processor::interface_connections, min_pkt_latency, MSHR_SIZE, Simulator::Now(), out_file, out_filename, OUT_PULL_EVENT, outstanding_hlp, packets_pending, pending_packets, NetworkComponent::process_event(), READY_EVENT, Simulator::Schedule(), sending, HighLevelPacket::sent_time, HighLevelPacket::toString(), IrisEvent::type, and IrisEvent::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



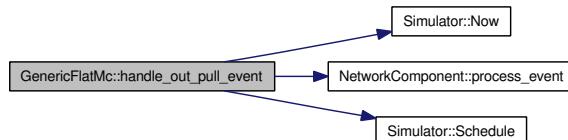
7.40.3.4 void GenericFlatMc::handle_out_pull_event (IrisEvent * e) [private]

Definition at line 190 of file genericFlatMc.cc.

References `NetworkComponent::address`, `HighLevelPacket::data`, `HighLevelPacket::data_payload_length`, `HighLevelPacket::destination`, `Processor::interface_connections`, `max_phy_link_bits`, `HighLevelPacket::msg_class`, `NEW_PACKET_EVENT`, `Simulator::Now()`, `OUT_PULL_EVENT`, `packets`, `packets_pending`, `pending_packets`, `NetworkComponent::process_event()`, `ready`, `RESPONSE_PKT`, `Simulator::Schedule()`, `sending`, `HighLevelPacket::sent_time`, `HighLevelPacket::source`, `HighLevelPacket::stat_memory_serviced_time`, `HighLevelPacket::transaction_id`, `HighLevelPacket::virtual_channel`, `waiting_at_injection`, and `HighLevelPacket::waiting_in_ni`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



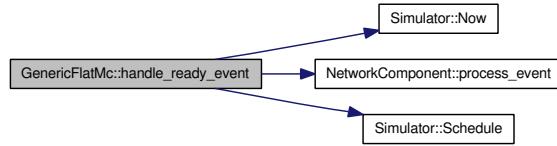
7.40.3.5 void GenericFlatMc::handle_ready_event (IrisEvent * e) [private]

Definition at line 272 of file genericFlatMc.cc.

References `_DBG_NOARG`, `max_sim_time`, `MSHR_SIZE`, `NEW_PACKET_EVENT`, `Simulator::Now()`, `OUT_PULL_EVENT`, `outstanding_hlp`, `packets_pending`, `NetworkComponent::process_event()`, `ready`, `Simulator::Schedule()`, and `sending`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.40.3.6 string GenericFlatMc::print_stats () const [virtual]

Implements **Processor** (p. 296).

Definition at line 347 of file genericFlatMc.cc.

References `min_pkt_latency`, `NetworkComponent::node_ip`, `packets`, and `waiting_at_injection`.

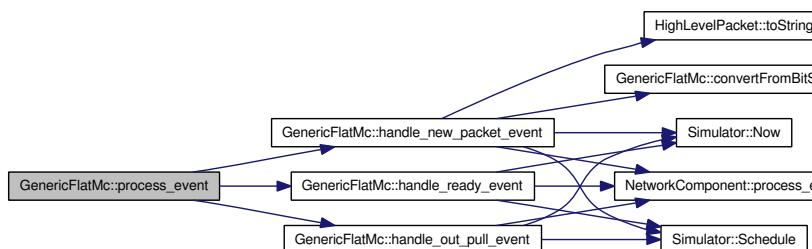
7.40.3.7 void GenericFlatMc::process_event (IrisEvent * e) [virtual]

Implements **Processor** (p. 297).

Definition at line 74 of file genericFlatMc.cc.

References `NetworkComponent::address`, `handle_new_packet_event()`, `handle_out_pull_event()`, `handle_ready_event()`, `NEW_PACKET_EVENT`, `OUT_PULL_EVENT`, `READY_EVENT`, and `IrisEvent::type`.

Here is the call graph for this function:



7.40.3.8 void GenericFlatMc::set_output_path (string outpath) [virtual]

Implements **Processor** (p. 297).

Definition at line 53 of file genericFlatMc.cc.

References NetworkComponent::address, out_file, out_filename, and timed_cout().

Here is the call graph for this function:



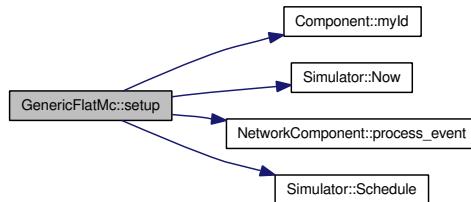
7.40.3.9 void GenericFlatMc::setup (uint no_nodes, uint vcs, uint max_sim_time) [virtual]

Implements **Processor** (p. 297).

Definition at line 24 of file genericFlatMc.cc.

References NetworkComponent::address, max_sim_time, min_pkt_latency, Component::myId(), no_nodes, Simulator::Now(), packets, packets_pending, NetworkComponent::process_event(), ready, READY_EVENT, Simulator::Schedule(), and vcs.

Here is the call graph for this function:



7.40.3.10 string GenericFlatMc::toString () const [virtual]

Reimplemented from **Processor** (p. 297).

Definition at line 306 of file genericFlatMc.cc.

References NetworkComponent::address, NetworkComponent::node_ip, and ready.

7.40.4 Member Data Documentation

7.40.4.1 unsigned long long int GenericFlatMc::max_sim_time [private]

Definition at line 28 of file genericFlatMc.h.

Referenced by handle_ready_event(), and setup().

7.40.4.2 unsigned long long int GenericFlatMc::max_time

Definition at line 51 of file genericFlatMc.h.

7.40.4.3 vector<uint> GenericFlatMc::mc_node_ip

Definition at line 58 of file genericFlatMc.h.

7.40.4.4 double GenericFlatMc::min_pkt_latency

Definition at line 48 of file genericFlatMc.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.40.4.5 uint GenericFlatMc::no_nodes [private]

Definition at line 27 of file genericFlatMc.h.

Referenced by setup().

7.40.4.6 ofstream GenericFlatMc::out_file [private]

Definition at line 31 of file genericFlatMc.h.

Referenced by handle_new_packet_event(), set_output_path(), and ~GenericFlatMc().

7.40.4.7 string GenericFlatMc::out_filename [private]

Definition at line 30 of file genericFlatMc.h.

Referenced by handle_new_packet_event(), and set_output_path().

7.40.4.8 deque< HighLevelPacket > GenericFlatMc::out_packets [private]

Definition at line 29 of file genericFlatMc.h.

7.40.4.9 HighLevelPacket* GenericFlatMc::outstanding_hlp

Definition at line 59 of file genericFlatMc.h.

Referenced by handle_new_packet_event(), and handle_ready_event().

7.40.4.10 unsigned int GenericFlatMc::packets

Definition at line 47 of file genericFlatMc.h.

Referenced by handle_out_pull_event(), print_stats(), and setup().

7.40.4.11 long int GenericFlatMc::packets_pending [private]

Definition at line 38 of file genericFlatMc.h.

Referenced by handle_new_packet_event(), handle_out_pull_event(), handle_ready_event(), and setup().

7.40.4.12 deque<HighLevelPacket*> GenericFlatMc::pending_packets [private]

Definition at line 40 of file genericFlatMc.h.

Referenced by handle_new_packet_event(), handle_out_pull_event(), and ~GenericFlatMc().

7.40.4.13 deque<unsigned long long int> GenericFlatMc::pending_packets_time [private]

Definition at line 39 of file genericFlatMc.h.

Referenced by ~GenericFlatMc().

7.40.4.14 vector< bool > GenericFlatMc::ready [private]

Definition at line 32 of file genericFlatMc.h.

Referenced by handle_out_pull_event(), handle_ready_event(), setup(), and toString().

7.40.4.15 bool GenericFlatMc::sending [private]

Definition at line 33 of file genericFlatMc.h.

Referenced by GenericFlatMc(), handle_new_packet_event(), handle_out_pull_event(), and handle_ready_event().

7.40.4.16 uint GenericFlatMc::vcs [private]

Definition at line 26 of file genericFlatMc.h.

Referenced by setup().

7.40.4.17 unsigned long long int GenericFlatMc::waiting_at_injection

Definition at line 49 of file genericFlatMc.h.

Referenced by handle_out_pull_event(), and print_stats().

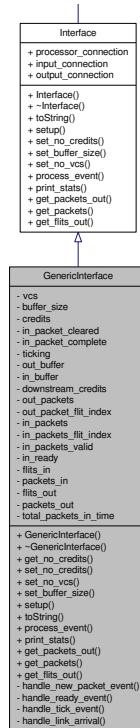
The documentation for this class was generated from the following files:

- **genericFlatMc.h**
- **genericFlatMc.cc**

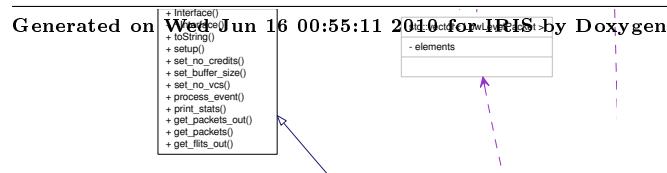
7.41 GenericInterface Class Reference

```
#include <genericInterface.h>
```

Inheritance diagram for GenericInterface:



Collaboration diagram for GenericInterface:



Public Member Functions

- `GenericInterface ()`
- `~GenericInterface ()`
- `uint get_no_credits () const`
- `void set_no_credits (int credits)`
- `void set_no_vcs (uint v)`
- `void set_buffer_size (uint b)`
- `void setup (uint v, uint cr)`
- `string toString () const`
- `void process_event (IrisEvent *e)`
- `string print_stats ()`
- `ullint get_packets_out ()`
- `ullint get_packets ()`
- `ullint get_flits_out ()`

Private Member Functions

- `void handle_new_packet_event (IrisEvent *e)`
- `void handle_ready_event (IrisEvent *e)`
- `void handle_tick_event (IrisEvent *e)`
- `void handle_link_arrival (IrisEvent *e)`

Private Attributes

- `uint vcs`
- `uint buffer_size`
- `int credits`
- `bool in_packet_cleared`
- `bool in_packet_complete`
- `bool ticking`
- `GenericBuffer out_buffer`
- `GenericBuffer in_buffer`
- `vector< int > downstream_credits`
- `vector< LowLevelPacket > out_packets`
- `vector< uint > out_packet_flt_index`
- `vector< LowLevelPacket > in_packets`
- `vector< uint > in_packets_flt_index`
- `vector< bool > in_packets_valid`
- `vector< bool > in_ready`
- `ullint flits_in`
- `ullint packets_in`
- `ullint flits_out`
- `ullint packets_out`
- `ullint total_packets_in_time`

7.41.1 Detailed Description

Definition at line 44 of file genericInterface.h.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 GenericInterface::GenericInterface ()

Definition at line 25 of file genericInterface.cc.

References NetworkComponent::name, and ticking.

7.41.2.2 GenericInterface::~GenericInterface ()

Definition at line 31 of file genericInterface.cc.

References in_packets, and out_packets.

7.41.3 Member Function Documentation

7.41.3.1 ullint GenericInterface::get_flits_out () [virtual]

Implements **Interface** (p. 231).

Definition at line 507 of file genericInterface.cc.

References flits_out.

7.41.3.2 uint GenericInterface::get_no_credits () const

7.41.3.3 ullint GenericInterface::get_packets () [virtual]

Implements **Interface** (p. 231).

Definition at line 519 of file genericInterface.cc.

References packets_in, and packets_out.

7.41.3.4 ullint GenericInterface::get_packets_out () [virtual]

Implements **Interface** (p. 231).

Definition at line 513 of file genericInterface.cc.

References packets_out.

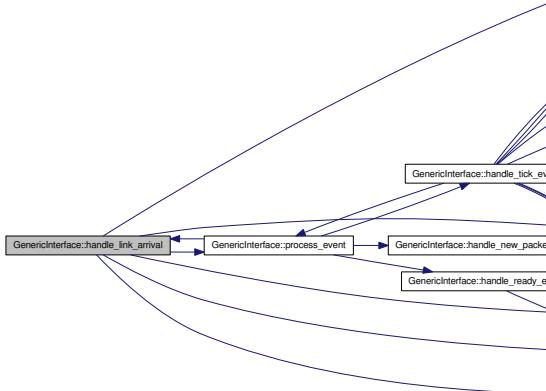
7.41.3.5 void GenericInterface::handle_link_arrival (IrisEvent * e) [private]

Definition at line 192 of file genericInterface.cc.

References _DBG, NetworkComponent::address, GenericBuffer::change_push_channel(), CREDIT_ID, do_two_stage_router, downstream_credits, IrisEvent::event_data, FLIT_ID, flits_in, HEAD, in_buffer, Interface::input_connection, LINK_ARRIVAL_EVENT, Simulator::Now(), ONE_FLIT_REQ, packets_in, process_event(), NetworkComponent::process_event(), LinkArrivalData::ptr, GenericBuffer::push(), Simulator::Schedule(), TAIL, TICK_EVENT, ticking, total_packets_in_time, IrisEvent::type, Flit::type, LinkArrivalData::type, LinkArrivalData::valid, IrisEvent::vc, and LinkArrivalData::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



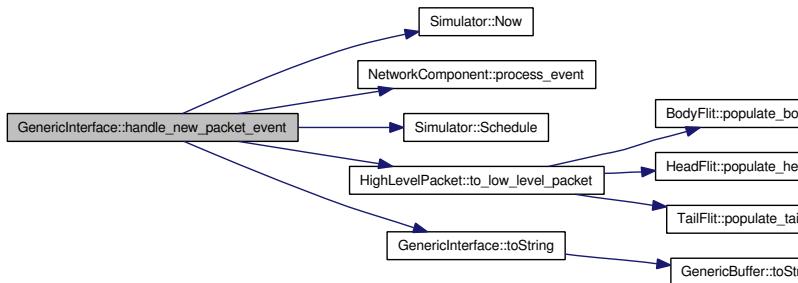
7.41.3.6 void GenericInterface::handle_new_packet_event (IrisEvent * e) [private]

Definition at line 266 of file genericInterface.cc.

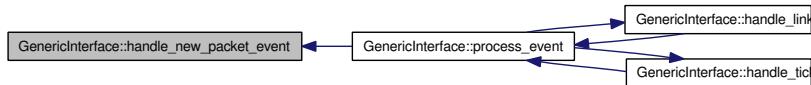
References `_DBG`, `IrisEvent::event_data`, `Simulator::Now()`, `out_packet_flist_index`, `out_packets`, `NetworkComponent::process_event()`, `Simulator::Schedule()`, `TICK_EVENT`, `ticking`, `HighLevelPacket::to_low_level_packet()`, `toString()`, `IrisEvent::vc`, and `HighLevelPacket::virtual_channel`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.41.3.7 void GenericInterface::handle_ready_event (IrisEvent * e) [private]

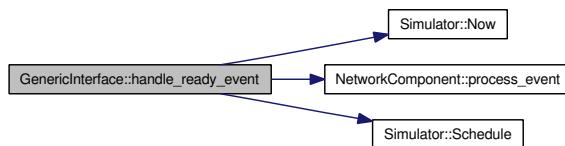
These are credits from the processor and hence do not update flow control state. Flow control state is updated at the link arrival event for a credit.

Definition at line 166 of file genericInterface.cc.

References _DBG, in_ready, Simulator::Now(), NetworkComponent::process_event(), Simulator::Schedule(), TICK_EVENT, ticking, and IrisEvent::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



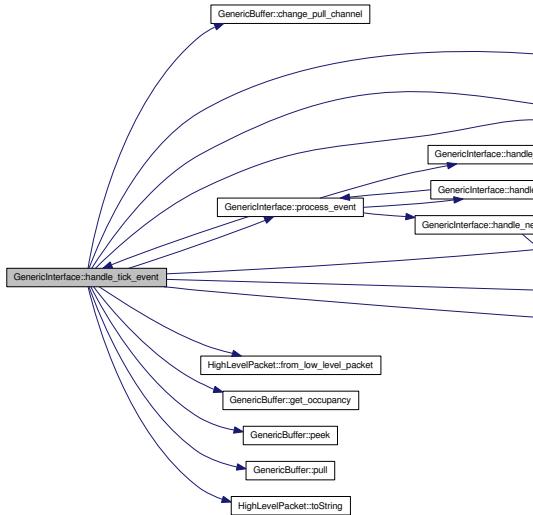
7.41.3.8 void GenericInterface::handle_tick_event (IrisEvent * e) [private]

Definition at line 296 of file genericInterface.cc.

References _DBG, NetworkComponent::address, HighLevelPacket::avg_network_latency, GenericBuffer::change_pull_channel(), GenericBuffer::change_push_channel(), CREDIT_ID, credits, do_two_stage_router, downstream_credits, IrisEvent::event_data, FLIT_ID, flits_out, HighLevelPacket::from_low_level_packet(), GenericBuffer::get_occupancy(), HEAD, HighLevelPacket::hop_count, in_buffer, in_packet_complete, in_packets, in_packets_flist_index, in_ready, Interface::input_connection, LINK_ARRIVAL_EVENT, NEW_PACKET_EVENT, NetworkComponent::node_ip, Simulator::Now(), ONE_FLIT_REQ, out_buffer, out_packet_flist_index, out_packets, Interface::output_connection, packets_out, GenericBuffer::peek(), process_event(), NetworkComponent::process_event(), Interface::processor_connection, LinkArrivalData::ptr, GenericBuffer::pull(), GenericBuffer::push(), READY_EVENT, HighLevelPacket::recv_time, HighLevelPacket::req_start_time, Simulator::Schedule(), IrisEvent::src_id, TAIL, TICK_EVENT, ticking, HighLevelPacket::toString(), toString(), IrisEvent::type, LinkArrivalData::type, Flit::type, IrisEvent::vc, LinkArrivalData::vc, and HighLevelPacket::waiting_in_ni.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.41.3.9 string GenericInterface::print_stats () [virtual]

Implements **Interface** (p. 231).

Definition at line 491 of file genericInterface.cc.

References flits_in, flits_out, NetworkComponent::node_ip, packets_in, packets_out, and total_packets_in_time.

7.41.3.10 void GenericInterface::process_event (IrisEvent * e) [virtual]

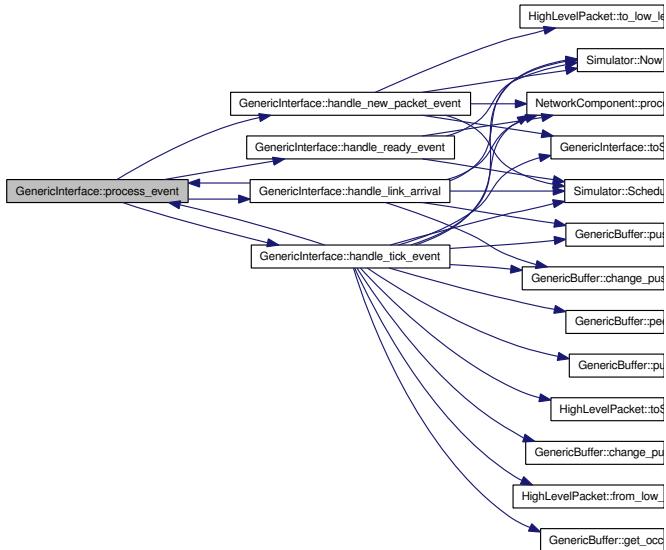
Implements **Interface** (p. 231).

Definition at line 139 of file genericInterface.cc.

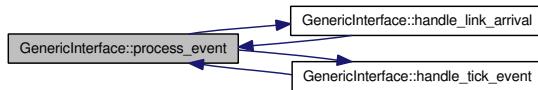
References handle_link_arrival(), handle_new_packet_event(), handle_ready_event(), handle_tick_event(), LINK_ARRIVAL_EVENT, NEW_PACKET_EVENT, READY_EVENT, TICK_EVENT, and IrisEvent::type.

Referenced by handle_link_arrival(), and handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.41.3.11 void GenericInterface::set_buffer_size (uint *b*) [virtual]

Implements **Interface** (p. 231).

Definition at line 104 of file genericInterface.cc.

References `buffer_size`.

7.41.3.12 void GenericInterface::set_no_credits (int *credits*) [virtual]

Implements **Interface** (p. 231).

Definition at line 119 of file genericInterface.cc.

References `credits`.

7.41.3.13 void GenericInterface::set_no_vcs (uint *v*) [virtual]

Implements **Interface** (p. 231).

Definition at line 98 of file genericInterface.cc.

References `vcs`.

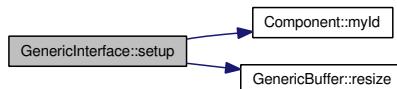
7.41.3.14 void GenericInterface::setup (uint *v*, uint *cr*) [virtual]

Implements **Interface** (p. 231).

Definition at line 39 of file genericInterface.cc.

References NetworkComponent::address, buffer_size, credits, downstream_credits, flits_in, flits_out, in_buffer, in_packet_complete, in_packets, in_packets_flit_index, in_packets_valid, in_ready, Component::myId(), out_buffer, out_packet_flit_index, out_packets, packets_in, packets_out, GenericBuffer::resize(), total_packets_in_time, and vcs.

Here is the call graph for this function:



7.41.3.15 string GenericInterface::toString () const [virtual]

Reimplemented from **Interface** (p. 231).

Definition at line 126 of file genericInterface.cc.

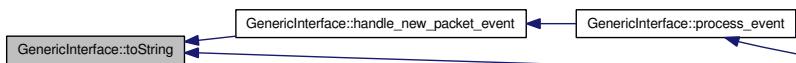
References NetworkComponent::address, in_buffer, NetworkComponent::node_ip, out_buffer, out_packets, and GenericBuffer::toString().

Referenced by handle_new_packet_event(), and handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.41.4 Member Data Documentation

7.41.4.1 uint GenericInterface::buffer_size [private]

Definition at line 67 of file genericInterface.h.

Referenced by set_buffer_size(), and setup().

7.41.4.2 int GenericInterface::credits [private]

Definition at line 68 of file genericInterface.h.

Referenced by handle_tick_event(), set_no_credits(), and setup().

7.41.4.3 vector< int > GenericInterface::downstream_credits [private]

Definition at line 75 of file genericInterface.h.

Referenced by handle_link_arrival(), handle_tick_event(), and setup().

7.41.4.4 ullint GenericInterface::flits_in [private]

Definition at line 95 of file genericInterface.h.

Referenced by handle_link_arrival(), print_stats(), and setup().

7.41.4.5 ullint GenericInterface::flits_out [private]

Definition at line 97 of file genericInterface.h.

Referenced by get_flits_out(), handle_tick_event(), print_stats(), and setup().

7.41.4.6 GenericBuffer GenericInterface::in_buffer [private]

Definition at line 74 of file genericInterface.h.

Referenced by handle_link_arrival(), handle_tick_event(), setup(), and toString().

7.41.4.7 bool GenericInterface::in_packet_cleared [private]

Definition at line 69 of file genericInterface.h.

7.41.4.8 bool GenericInterface::in_packet_complete [private]

Definition at line 70 of file genericInterface.h.

Referenced by handle_tick_event(), and setup().

7.41.4.9 vector< LowLevelPacket> GenericInterface::in_packets [private]

Definition at line 82 of file genericInterface.h.

Referenced by handle_tick_event(), setup(), and ~GenericInterface().

7.41.4.10 vector< uint > GenericInterface::in_packets_flit_index [private]

Definition at line 83 of file genericInterface.h.

Referenced by handle_tick_event(), and setup().

7.41.4.11 vector< bool> GenericInterface::in_packets_valid [private]

Definition at line 84 of file genericInterface.h.

Referenced by setup().

7.41.4.12 `vector< bool > GenericInterface::in_ready [private]`

Definition at line 86 of file genericInterface.h.

Referenced by handle_ready_event(), handle_tick_event(), and setup().

7.41.4.13 `GenericBuffer GenericInterface::out_buffer [private]`

Definition at line 73 of file genericInterface.h.

Referenced by handle_tick_event(), setup(), and toString().

7.41.4.14 `vector< uint > GenericInterface::out_packet_flit_index [private]`

Definition at line 79 of file genericInterface.h.

Referenced by handle_new_packet_event(), handle_tick_event(), and setup().

7.41.4.15 `vector< LowLevelPacket> GenericInterface::out_packets [private]`

Definition at line 78 of file genericInterface.h.

Referenced by handle_new_packet_event(), handle_tick_event(), setup(), toString(), and ~GenericInterface().

7.41.4.16 `ullint GenericInterface::packets_in [private]`

Definition at line 96 of file genericInterface.h.

Referenced by get_packets(), handle_link_arrival(), print_stats(), and setup().

7.41.4.17 `ullint GenericInterface::packets_out [private]`

Definition at line 98 of file genericInterface.h.

Referenced by get_packets(), get_packets_out(), handle_tick_event(), print_stats(), and setup().

7.41.4.18 `bool GenericInterface::ticking [private]`

Definition at line 72 of file genericInterface.h.

Referenced by GenericInterface(), handle_link_arrival(), handle_new_packet_event(), handle_ready_event(), and handle_tick_event().

7.41.4.19 `ullint GenericInterface::total_packets_in_time [private]`

Definition at line 99 of file genericInterface.h.

Referenced by handle_link_arrival(), print_stats(), and setup().

7.41.4.20 uint GenericInterface::vcs [private]

Definition at line 66 of file genericInterface.h.

Referenced by set_no_vcs(), and setup().

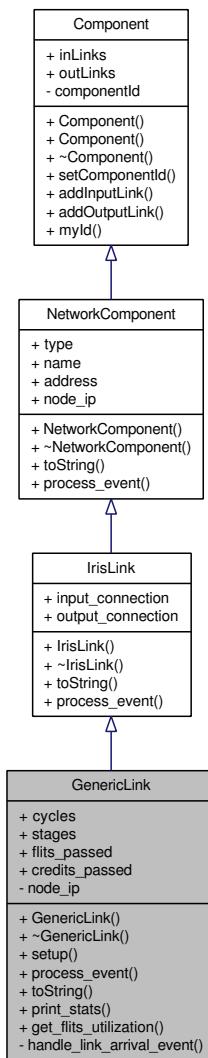
The documentation for this class was generated from the following files:

- [genericInterface.h](#)
- [genericInterface.cc](#)

7.42 GenericLink Class Reference

```
#include <genericLink.h>
```

Inheritance diagram for GenericLink:



Collaboration diagram for GenericLink:

Public Member Functions

- `GenericLink ()`
- `~GenericLink ()`
- `void setup ()`
- `void process_event (IrisEvent *e)`
- `string toString () const`
- `string print_stats () const`
- `ullint get_flits_utilization ()`

Public Attributes

- `uint cycles`
- `uint stages`
- `unsigned long long int flits_passed`
- `unsigned long long int credits_passed`

Private Member Functions

- `void handle_link_arrival_event (IrisEvent *e)`

Private Attributes

- `uint node_ip`

7.42.1 Detailed Description

Definition at line 35 of file genericLink.h.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 GenericLink::GenericLink () [inline]

Definition at line 38 of file genericLink.h.

7.42.2.2 GenericLink::~GenericLink () [inline]

Definition at line 39 of file genericLink.h.

7.42.3 Member Function Documentation

7.42.3.1 ullint GenericLink::get_flits_utilization ()

Definition at line 102 of file genericLink.cc.

References flits_passed.

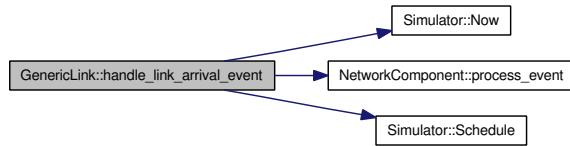
7.42.3.2 void GenericLink::handle_link_arrival_event (IrisEvent * e) [private]

Definition at line 51 of file genericLink.cc.

References `_DBG`, `NetworkComponent::address`, `credits_passed`, `IrisEvent::dst_id`, `IrisEvent::event_data`, `flits_passed`, `IrisLink::input_connection`, `Simulator::Now()`, `IrisLink::output_connection`, `NetworkComponent::process_event()`, `Simulator::Schedule()`, `IrisEvent::src_id`, `LinkArrivalData::type`, and `LinkArrivalData::vc`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.3 string GenericLink::print_stats () const

Definition at line 90 of file genericLink.cc.

References `NetworkComponent::address`, `credits_passed`, `flits_passed`, `max_sim_time`, and `toString()`.

Here is the call graph for this function:



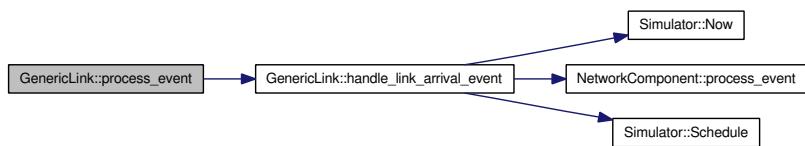
7.42.3.4 void GenericLink::process_event (IrisEvent * e) [virtual]

Implements **IrisLink** (p. 236).

Definition at line 35 of file genericLink.cc.

References `handle_link_arrival_event()`, `LINK_ARRIVAL_EVENT`, and `IrisEvent::type`.

Here is the call graph for this function:



7.42.3.5 void GenericLink::setup ()

Definition at line 25 of file genericLink.cc.

References NetworkComponent::address, credits_passed, flits_passed, Component::myId(), NetworkComponent::name, and node_ip.

Here is the call graph for this function:



7.42.3.6 string GenericLink::toString () const [virtual]

Reimplemented from **IrisLink** (p. 237).

Definition at line 76 of file genericLink.cc.

References NetworkComponent::address, IrisLink::input_connection, and IrisLink::output_connection.

Referenced by print_stats().

Here is the caller graph for this function:



7.42.4 Member Data Documentation

7.42.4.1 unsigned long long int GenericLink::credits_passed

Definition at line 48 of file genericLink.h.

Referenced by handle_link_arrival_event(), print_stats(), and setup().

7.42.4.2 uint GenericLink::cycles

Definition at line 40 of file genericLink.h.

7.42.4.3 unsigned long long int GenericLink::flits_passed

Definition at line 47 of file genericLink.h.

Referenced by get_flits_utilization(), handle_link_arrival_event(), print_stats(), and setup().

7.42.4.4 uint GenericLink::node_ip [private]

Reimplemented from **NetworkComponent** (p. 276).

Definition at line 53 of file genericLink.h.

Referenced by setup().

7.42.4.5 `uint GenericLink::stages`

Definition at line 41 of file genericLink.h.

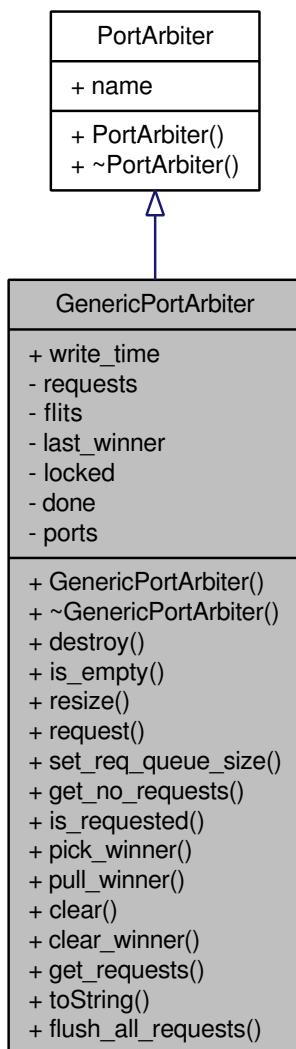
The documentation for this class was generated from the following files:

- `genericLink.h`
- `genericLink.cc`

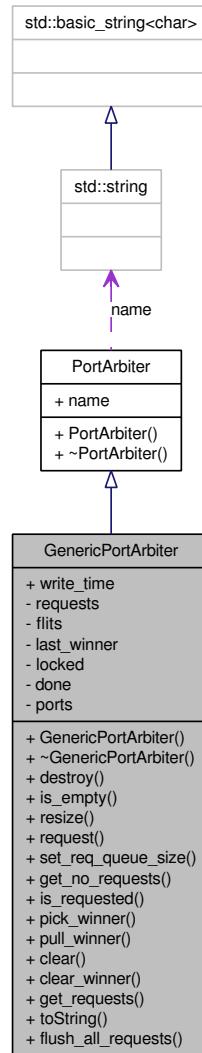
7.43 GenericPortArbiter Class Reference

```
#include <genericPortArbiter.h>
```

Inheritance diagram for GenericPortArbiter:



Collaboration diagram for GenericPortArbiter:



Public Member Functions

- **GenericPortArbiter ()**
- **~GenericPortArbiter ()**
- **void `destroy ()`**
- **bool `is_empty (uint ch)`**
- **void `resize (uint ports)`**
- **void `request (Flit *f, uint port, uint vc)`**
- **void `set_req_queue_size (uint ch)`**
- **uint `get_no_requests ()`**
- **bool `is_requested (uint port, uint ch)`**
- **uint `pick_winner (uint ch)`**
- **Flit * `pull_winner (uint ch)`**
- **void `clear (uint ch)`**
- **void `clear_winner (uint ch)`**

- `vector< uint > get_requests (uint ch)`
- `string toString () const`
- `void flush_all_requests ()`

Public Attributes

- `unsigned long long int write_time`

Private Attributes

- `vector< vector< bool > > requests`
- `vector< vector< Flit * > > flits`
- `vector< uint > last_winner`
- `vector< bool > locked`
- `vector< bool > done`
- `uint ports`

7.43.1 Detailed Description

Definition at line 30 of file genericPortArbiter.h.

7.43.2 Constructor & Destructor Documentation

7.43.2.1 GenericPortArbiter::GenericPortArbiter ()

Definition at line 26 of file genericPortArbiter.cc.

References PortArbiter::name.

7.43.2.2 GenericPortArbiter::~GenericPortArbiter ()

Definition at line 31 of file genericPortArbiter.cc.

7.43.3 Member Function Documentation

7.43.3.1 void GenericPortArbiter::clear (uint ch)

7.43.3.2 void GenericPortArbiter::clear_winner (uint ch)

7.43.3.3 void GenericPortArbiter::destroy ()

7.43.3.4 void GenericPortArbiter::flush_all_requests ()

Definition at line 36 of file genericPortArbiter.cc.

References flits, and requests.

7.43.3.5 uint GenericPortArbiter::get_no_requests ()

Definition at line 50 of file genericPortArbiter.cc.

References requests.

7.43.3.6 vector<uint> GenericPortArbiter::get_requests (uint ch)**7.43.3.7 bool GenericPortArbiter::is_empty (uint ch)**

Definition at line 81 of file genericPortArbiter.cc.

References last_winner, locked, and requests.

7.43.3.8 bool GenericPortArbiter::is_requested (uint port, uint ch)

Definition at line 187 of file genericPortArbiter.cc.

References requests.

7.43.3.9 uint GenericPortArbiter::pick_winner (uint ch)

Definition at line 113 of file genericPortArbiter.cc.

References done, last_winner, locked, and requests.

7.43.3.10 Flit * GenericPortArbiter::pull_winner (uint ch)

Definition at line 149 of file genericPortArbiter.cc.

References done, flits, HEAD, last_winner, locked, requests, TAIL, and Flit::type.

7.43.3.11 void GenericPortArbiter::request (Flit * f, uint port, uint vc)

Definition at line 97 of file genericPortArbiter.cc.

References flits, and requests.

7.43.3.12 void GenericPortArbiter::resize (uint ports)**7.43.3.13 void GenericPortArbiter::set_req_queue_size (uint ch)**

Definition at line 57 of file genericPortArbiter.cc.

References done, flits, last_winner, locked, ports, and requests.

7.43.3.14 string GenericPortArbiter::toString () const

Definition at line 201 of file genericPortArbiter.cc.

References requests.

7.43.4 Member Data Documentation

7.43.4.1 `vector<bool> GenericPortArbiter::done [private]`

Definition at line 58 of file genericPortArbiter.h.

Referenced by `pick_winner()`, `pull_winner()`, and `set_req_queue_size()`.

7.43.4.2 `vector< vector<Flit*>> GenericPortArbiter::flits [private]`

Definition at line 55 of file genericPortArbiter.h.

Referenced by `flush_all_requests()`, `pull_winner()`, `request()`, and `set_req_queue_size()`.

7.43.4.3 `vector< uint> GenericPortArbiter::last_winner [private]`

Definition at line 56 of file genericPortArbiter.h.

Referenced by `is_empty()`, `pick_winner()`, `pull_winner()`, and `set_req_queue_size()`.

7.43.4.4 `vector<bool> GenericPortArbiter::locked [private]`

Definition at line 57 of file genericPortArbiter.h.

Referenced by `is_empty()`, `pick_winner()`, `pull_winner()`, and `set_req_queue_size()`.

7.43.4.5 `uint GenericPortArbiter::ports [private]`

Definition at line 59 of file genericPortArbiter.h.

Referenced by `set_req_queue_size()`.

7.43.4.6 `vector< vector<bool>> GenericPortArbiter::requests [private]`

Definition at line 54 of file genericPortArbiter.h.

Referenced by `flush_all_requests()`, `get_no_requests()`, `is_empty()`, `is_requested()`, `pick_winner()`, `pull_winner()`, `request()`, `set_req_queue_size()`, and `toString()`.

7.43.4.7 `unsigned long long int GenericPortArbiter::write_time`

Definition at line 48 of file genericPortArbiter.h.

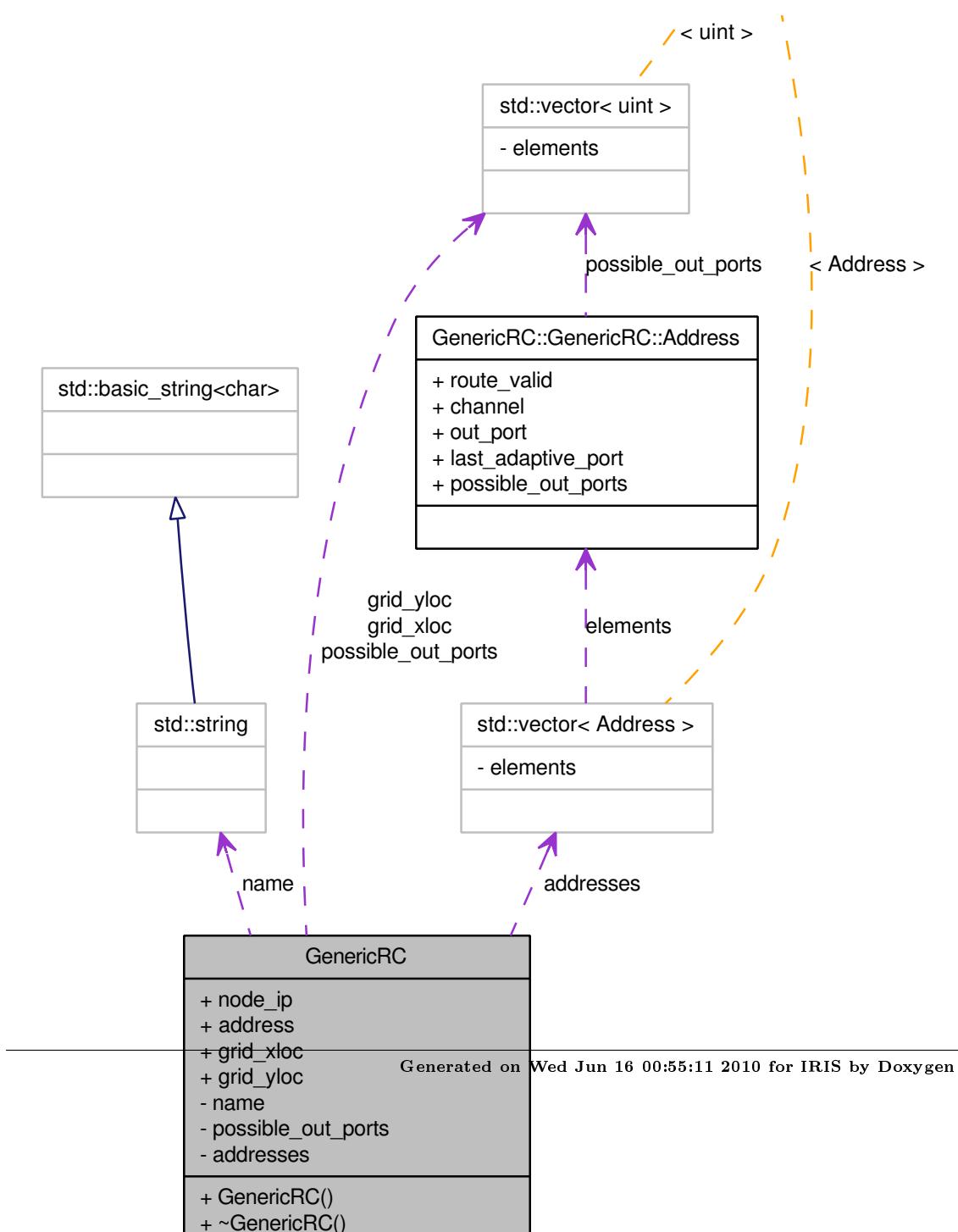
The documentation for this class was generated from the following files:

- `genericPortArbiter.h`
- `genericPortArbiter.cc`

7.44 GenericRC Class Reference

```
#include <genericRC.h>
```

Collaboration diagram for GenericRC:



Classes

- class **Address**

Public Member Functions

- **GenericRC ()**
- **~GenericRC ()**
- void **push** (**Flit** *f, unsigned int vc)
- unsigned int **get_output_port** (unsigned int channel)
- unsigned int **speculate_port** (**Flit** *f, unsigned int ch)
- unsigned int **speculate_channel** (**Flit** *f, unsigned int ch)
- unsigned int **get_virtual_channel** (unsigned int ch)
- void **resize** (unsigned int ch)
- unsigned int **get_no_channels** ()
- bool **is_empty** ()
- void **route_negative_first** (**HeadFlit** *hf)
- void **route_west_first** (**HeadFlit** *hf)
- void **route_north_last** (**HeadFlit** *hf)
- void **route_north_last_non_minimal** (**HeadFlit** *hf)
- string **toString** () const

Public Attributes

- uint **node_ip**
- uint **address**
- vector< uint > **grid_xloc**
- vector< uint > **grid_yloc**

Private Member Functions

- uint **route_x_y** (uint addr)
- uint **route_valiant** (**HeadFlit** *hf)

Private Attributes

- string **name**
- vector< uint > **possible_out_ports**
- vector< **Address** > **addresses**

7.44.1 Detailed Description

Definition at line 37 of file genericRC.h.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 GenericRC::GenericRC ()

Definition at line 25 of file genericRC.cc.

References address, name, and node_ip.

7.44.2.2 GenericRC::~GenericRC () [inline]

Definition at line 41 of file genericRC.h.

7.44.3 Member Function Documentation

7.44.3.1 uint GenericRC::get_no_channels ()

Definition at line 543 of file genericRC.cc.

References addresses.

7.44.3.2 uint GenericRC::get_output_port (unsigned int *channel*)

Definition at line 514 of file genericRC.cc.

References addresses, and possible_out_ports.

7.44.3.3 uint GenericRC::get_virtual_channel (unsigned int *ch*)

Definition at line 526 of file genericRC.cc.

References addresses.

7.44.3.4 bool GenericRC::is_empty ()

Definition at line 549 of file genericRC.cc.

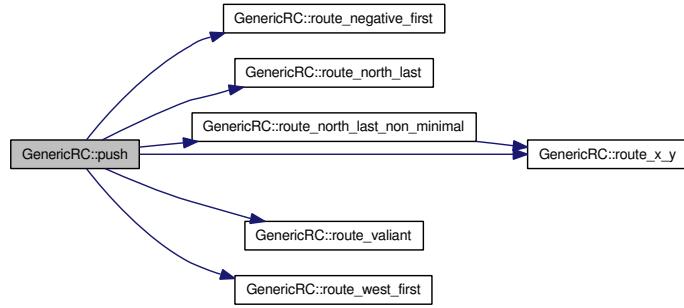
References addresses.

7.44.3.5 void GenericRC::push (Flit * *f*, unsigned int *vc*)

Definition at line 404 of file genericRC.cc.

References _DBG, _DBG_NOARG, addresses, BODY, HeadFlit::dst_address, HEAD, NEGATIVE_FIRST, NORTH_LAST, NORTH_LAST_NON_MINIMAL, possible_out_ports, rc_method, route_negative_first(), route_north_last(), route_north_last_non_minimal(), route_valiant(), route_west_first(), route_x_y(), TAIL, Flit::type, VALIANT, Flit::vc, and WEST_FIRST.

Here is the call graph for this function:



7.44.3.6 void GenericRC::resize (unsigned int *ch*)

Definition at line 532 of file genericRC.cc.

References addresses.

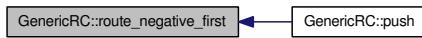
7.44.3.7 void GenericRC::route_negative_first (HeadFlit * *hf*)

Definition at line 109 of file genericRC.cc.

References _DBG, HeadFlit::dst_address, grid_xloc, grid_yloc, node_ip, and possible_out_ports.

Referenced by `push()`.

Here is the caller graph for this function:



7.44.3.8 void GenericRC::route_north_last (HeadFlit * *hf*)

Definition at line 238 of file genericRC.cc.

References _DBG, HeadFlit::dst_address, grid_xloc, grid_yloc, node_ip, and possible_out_ports.

Referenced by `push()`.

Here is the caller graph for this function:



7.44.3.9 void GenericRC::route_north_last_non_minimal (HeadFlit * *hf*)

Definition at line 318 of file genericRC.cc.

References _DBG, HeadFlit::dst_address, grid_size, grid_xloc, grid_yloc, HeadFlit::import, no_nodes, node_ip, possible_out_ports, and route_x_y().

Referenced by push().

Here is the call graph for this function:



Here is the caller graph for this function:



7.44.3.10 uint GenericRC::route_valiant (HeadFlit * hf) [private]

Definition at line 66 of file genericRC.cc.

Referenced by push().

Here is the caller graph for this function:



7.44.3.11 void GenericRC::route_west_first (HeadFlit * hf)

Definition at line 171 of file genericRC.cc.

References _DBG, HeadFlit::dst_address, grid_xloc, grid_yloc, node_ip, and possible_out_ports.

Referenced by push().

Here is the caller graph for this function:



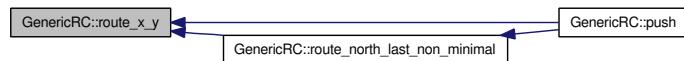
7.44.3.12 uint GenericRC::route_x_y (uint addr) [private]

Definition at line 33 of file genericRC.cc.

References _DBG, grid_xloc, grid_yloc, and node_ip.

Referenced by push(), and route_north_last_non_minimal().

Here is the caller graph for this function:



7.44.3.13 `unsigned int GenericRC::speculate_channel (Flit * f, unsigned int ch)`

7.44.3.14 `unsigned int GenericRC::speculate_port (Flit * f, unsigned int ch)`

7.44.3.15 `string GenericRC::toString () const`

Definition at line 560 of file genericRC.cc.

References addresses.

7.44.4 Member Data Documentation

7.44.4.1 `uint GenericRC::address`

Definition at line 56 of file genericRC.h.

Referenced by GenericRC().

7.44.4.2 `vector<Address> GenericRC::addresses [private]`

Definition at line 88 of file genericRC.h.

Referenced by get_no_channels(), get_output_port(), get_virtual_channel(), is_empty(), push(), resize(), and toString().

7.44.4.3 `vector< uint > GenericRC::grid_xloc`

Definition at line 57 of file genericRC.h.

Referenced by route_negative_first(), route_north_last(), route_north_last_non_minimal(), route_west_first(), and route_x_y().

7.44.4.4 `vector< uint > GenericRC::grid_yloc`

Definition at line 58 of file genericRC.h.

Referenced by route_negative_first(), route_north_last(), route_north_last_non_minimal(), route_west_first(), and route_x_y().

7.44.4.5 `string GenericRC::name [private]`

Definition at line 63 of file genericRC.h.

Referenced by GenericRC().

7.44.4.6 `uint GenericRC::node_ip`

Definition at line 55 of file genericRC.h.

Referenced by GenericRC(), route_negative_first(), route_north_last(), route_north_last_non_minimal(), route_west_first(), and route_x_y().

7.44.4.7 `vector< uint > GenericRC::possible_out_ports [private]`

Definition at line 66 of file genericRC.h.

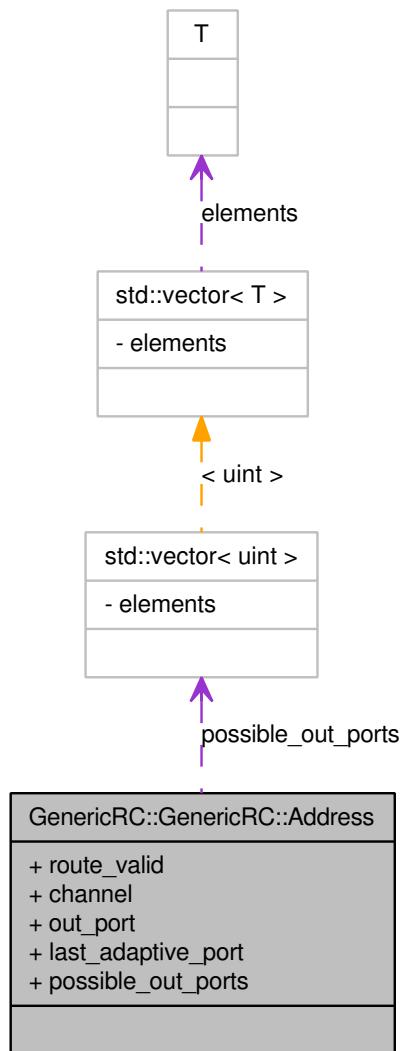
Referenced by `get_output_port()`, `push()`, `route_negative_first()`, `route_north_last()`, `route_north_last_non_minimal()`, and `route_west_first()`.

The documentation for this class was generated from the following files:

- `genericRC.h`
- `genericRC.cc`

7.45 GenericRC::GenericRC::Address Class Reference

Collaboration diagram for GenericRC::GenericRC::Address:



Public Attributes

- `bool route_valid`
- `unsigned int channel`
- `unsigned int out_port`
- `uint last_adaptive_port`
- `vector< uint > possible_out_ports`

7.45.1 Detailed Description

Definition at line 74 of file genericRC.h.

7.45.2 Member Data Documentation

7.45.2.1 `unsigned int GenericRC::GenericRC::Address::channel`

Definition at line 78 of file genericRC.h.

7.45.2.2 `uint GenericRC::GenericRC::Address::last_adaptive_port`

Definition at line 80 of file genericRC.h.

7.45.2.3 `unsigned int GenericRC::GenericRC::Address::out_port`

Definition at line 79 of file genericRC.h.

7.45.2.4 `vector< uint > GenericRC::GenericRC::Address::possible_out_ports`

Definition at line 81 of file genericRC.h.

7.45.2.5 `bool GenericRC::GenericRC::Address::route_valid`

Definition at line 77 of file genericRC.h.

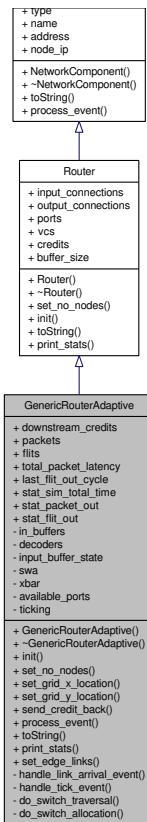
The documentation for this class was generated from the following file:

- `genericRC.h`

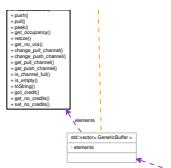
7.46 GenericRouterAdaptive Class Reference

```
#include <genericRouterAdaptive.h>
```

Inheritance diagram for GenericRouterAdaptive:



Collaboration diagram for GenericRouterAdaptive:



Public Member Functions

- **GenericRouterAdaptive ()**
- **~GenericRouterAdaptive ()**
- **void init (uint ports, uint vcs, uint credits, uint buffer_size)**

- void **set_no_nodes** (uint nodes)

These functions are mainly for DOR routing and are separated so as to not force DOR modelling in all designs.

- void **set_grid_x_location** (uint a, uint b, uint c)
- void **set_grid_y_location** (uint a, uint b, uint c)
- void **send_credit_back** (uint i)
- void **process_event** (IrisEvent *e)
- string **toString** () const
- string **print_stats** ()
- void **set_edge_links** ()

Public Attributes

- vector< vector< uint > > **downstream_credits**
- uint **packets**
- uint **flits**
- double **total_packet_latency**
- double **last_flit_out_cycle**
- unsigned long long int **stat_sim_total_time**
- vector< vector< uint > > **stat_packet_out**
- vector< vector< uint > > **stat_flit_out**

Private Member Functions

- void **handle_link_arrival_event** (IrisEvent *e)
Event handle for the LINK_ARRIVAL_EVENT event. Entry from DES kernel.
- void **handle_tick_event** (IrisEvent *e)
Event handle for the TICK_EVENT. Entry from DES kernel.
- void **do_switch_traversal** ()
- void **do_switch_allocation** ()

Private Attributes

- vector< **GenericBuffer** > **in_buffers**
- vector< **GenericRC** > **decoders**
- vector< **InputBufferState** > **input_buffer_state**
- **PToPSwitchArbiter** **swa**
- **GenericCrossbar** **xbar**
- vector< uint > **available_ports**
- bool **ticking**

7.46.1 Detailed Description

Definition at line 37 of file genericRouterAdaptive.h.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 GenericRouterAdaptive::GenericRouterAdaptive ()

Definition at line 39 of file genericRouterAdaptive.cc.

References NetworkComponent::name, and ticking.

7.46.2.2 GenericRouterAdaptive::~GenericRouterAdaptive ()

Definition at line 45 of file genericRouterAdaptive.cc.

7.46.3 Member Function Documentation

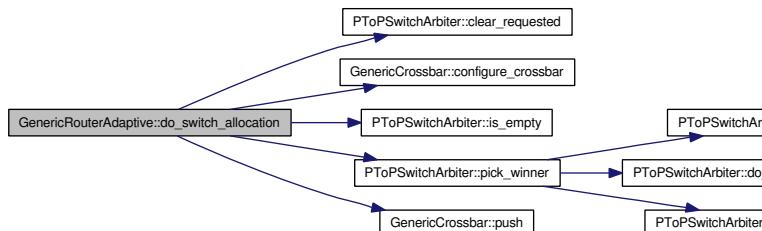
7.46.3.1 void GenericRouterAdaptive::do_switch_allocation () [private]

Definition at line 482 of file genericRouterAdaptive.cc.

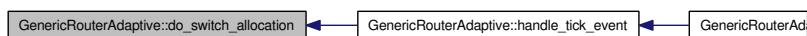
References _DBG, PToPSwitchArbiter::clear_requested(), GenericCrossbar::configure_crossbar(), FULL, input_buffer_state, PToPSwitchArbiter::is_empty(), PToPSwitchArbiter::pick_winner(), SA_unit::port, Router::ports, GenericCrossbar::push(), ST, swa, SWA_REQUESTED, ticking, Router::vcs, and xbar.

Referenced by handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.46.3.2 void GenericRouterAdaptive::do_switch_traversal () [private]

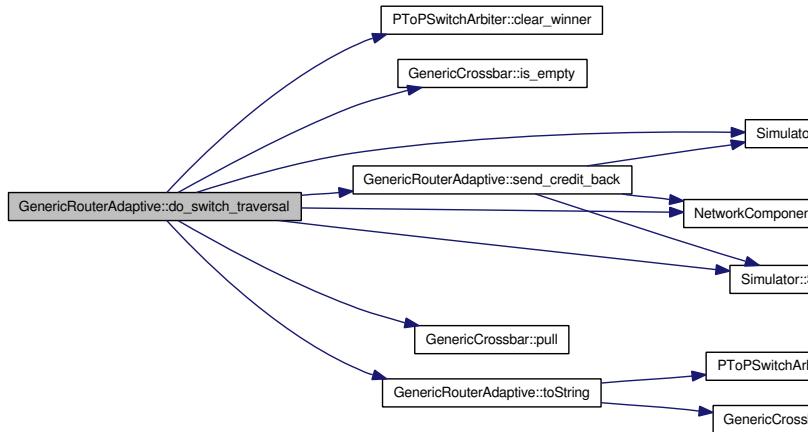
Definition at line 354 of file genericRouterAdaptive.cc.

References _DBG, _DBG_NOARG, NetworkComponent::address, TailFlit::avg_network_latency, HeadFlit::avg_network_latency, BODY, PToPSwitchArbiter::clear_winner(), do_two_stage_router, downstream_credits, EMPTY, FLIT_ID, HEAD, TailFlit::hop_count, HeadFlit::hop_count, in_buffers, input_buffer_state, GenericCrossbar::is_empty(), last_flit_out_cycle, LINK_ARRIVAL_EVENT, Simulator::Now(), ONE_FLIT_REQ, Router::output_connections, Router::ports, NetworkComponent::process_event(), LinkArrivalData::ptr, GenericCrossbar::pull(), Simulator::Schedule(), send_credit_back(), ST, stat_flit_out, stat_packet_

out, swa, TAIL, ticking, `toString()`, `total_packet_latency`, `Flit::type`, `LinkArrivalData::type`, `LinkArrivalData::vc`, `Router::vcs`, and `xbar`.

Referenced by `handle_tick_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.46.3.3 void GenericRouterAdaptive::handle_link_arrival_event (IrisEvent * e) [private]

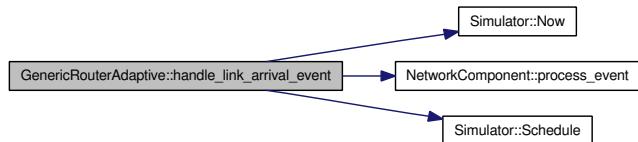
Event handle for the `LINK_ARRIVAL_EVENT` event. Entry from DES kernel.

Definition at line 233 of file `genericRouterAdaptive.cc`.

References `_DBG`, `available_ports`, `CREDIT_ID`, `decoders`, `downstream_credits`, `IrisEvent::event_data`, `FLIT_ID`, `flits`, `FULL`, `HEAD`, `in_buffers`, `HeadFlit::import`, `input_buffer_state`, `Router::input_connections`, `HeadFlit::length`, `HeadFlit::msg_class`, `Simulator::Now()`, `ONE_FLIT_REQ`, `Router::output_connections`, `packets`, `Router::ports`, `NetworkComponent::process_event()`, `LinkArrivalData::ptr`, `Simulator::Schedule()`, `IrisEvent::src_id`, `TAIL`, `TICK_EVENT`, `ticking`, `Flit::type`, `LinkArrivalData::type`, `IrisEvent::vc`, `LinkArrivalData::vc`, and `Router::vcs`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.46.3.4 void GenericRouterAdaptive::handle_tick_event (IrisEvent * e) [private]

Event handle for the TICK_EVENT. Entry from DES kernel.

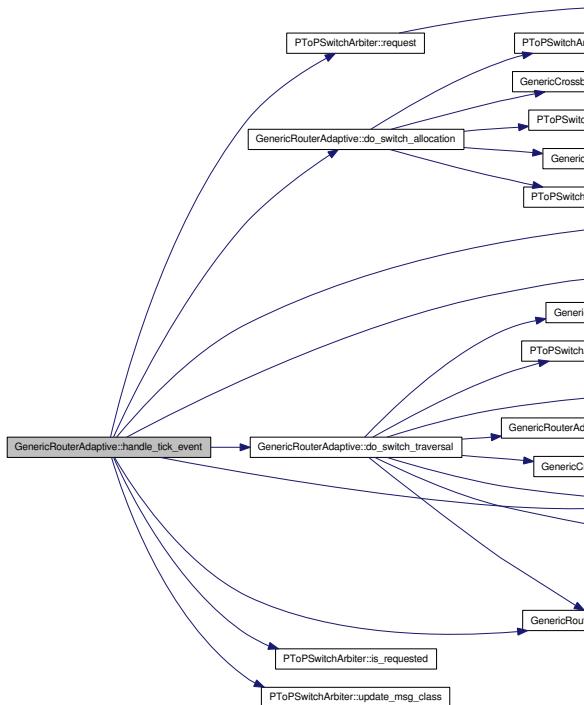
Body and tail flits get written in link arrival and since the message state may already been pushed to ST because of the header we want to ensure that all flits go thru an IB and ST stage. Hence ST is done on the flits_in_ib information and not buffer occupancy.

Definition at line 539 of file genericRouterAdaptive.cc.

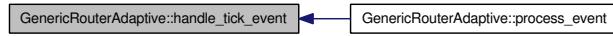
References _DBG, _DBG_NOARG, Router::credits, do_switch_allocation(), do_switch_traversal(), downstream_credits, FULL, IB, in_buffers, input_buffer_state, PToPSwitchArbiter::is_requested(), Simulator::Now(), Router::ports, NetworkComponent::process_event(), PToPSwitchArbiter::request(), Simulator::Schedule(), ST, swa, SWA_REQUESTED, TICK_EVENT, ticking, toString(), PToPSwitchArbiter::update_msg_class(), IrisEvent::vc, and Router::vcs.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



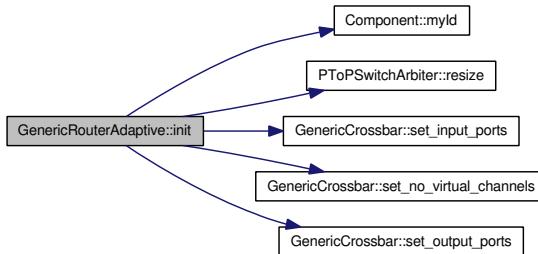
7.46.3.5 void GenericRouterAdaptive::init (uint ports, uint vcs, uint credits, uint buffer_size) [virtual]

Implements **Router** (p. 328).

Definition at line 50 of file genericRouterAdaptive.cc.

References PToPSwitchArbiter::address, NetworkComponent::address, Router::buffer_size, Router::credits, decoders, downstream_credits, flits, in_buffers, input_buffer_state, Component::myId(), PToPSwitchArbiter::node_ip, NetworkComponent::node_ip, packets, Router::ports, PToPSwitchArbiter::resize(), GenericCrossbar::set_input_ports(), GenericCrossbar::set_no_virtual_channels(), GenericCrossbar::set_output_ports(), stat_flt_out, stat_packet_out, swa, total_packet_latency, Router::vcs, and xbar.

Here is the call graph for this function:



7.46.3.6 string GenericRouterAdaptive::print_stats () [virtual]

Implements **Router** (p. 329).

Definition at line 159 of file genericRouterAdaptive.cc.

References flits, last_flt_out_cycle, NetworkComponent::node_ip, packets, Router::ports, stat_flt_out, stat_packet_out, and total_packet_latency.

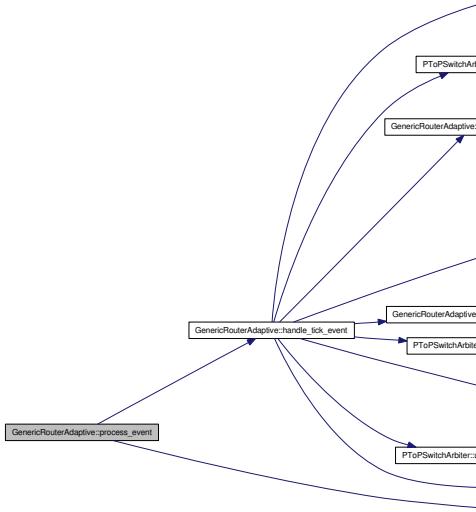
7.46.3.7 void GenericRouterAdaptive::process_event (IrisEvent * e) [virtual]

Implements **NetworkComponent** (p. 274).

Definition at line 141 of file genericRouterAdaptive.cc.

References _DBG, handle_link_arrival_event(), handle_tick_event(), LINK_ARRIVAL_EVENT, TICK_EVENT, and IrisEvent::type.

Here is the call graph for this function:



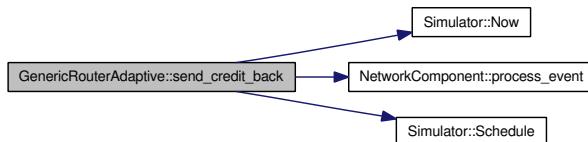
7.46.3.8 void GenericRouterAdaptive::send_credit_back (uint i)

Definition at line 702 of file genericRouterAdaptive.cc.

References _DBG, NetworkComponent::address, CREDIT_ID, do_two_stage_router, input_buffer_state, Router::input_connections, LINK_ARRIVAL_EVENT, Simulator::Now(), NetworkComponent::process_event(), Simulator::Schedule(), LinkArrivalData::type, and LinkArrivalData::vc.

Referenced by do_switch_traversal().

Here is the call graph for this function:



Here is the caller graph for this function:



7.46.3.9 void GenericRouterAdaptive::set_edge_links ()

Definition at line 131 of file genericRouterAdaptive.cc.

References available_ports, Router::input_connections, and Router::ports.

7.46.3.10 void GenericRouterAdaptive::set_grid_x_location (uint a, uint b, uint c)

Definition at line 119 of file genericRouterAdaptive.cc.

References decoders.

7.46.3.11 void GenericRouterAdaptive::set_grid_y_location (uint a, uint b, uint c)

Definition at line 125 of file genericRouterAdaptive.cc.

References decoders.

7.46.3.12 void GenericRouterAdaptive::set_no_nodes (uint nodes) [virtual]

These functions are mainly for DOR routing and are separated so as to not force DOR modelling in all designs.

Implements **Router** (p. 329).

Definition at line 109 of file genericRouterAdaptive.cc.

References decoders.

7.46.3.13 string GenericRouterAdaptive::toString () const [virtual]

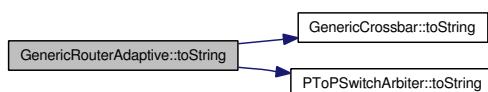
Reimplemented from **Router** (p. 329).

Definition at line 679 of file genericRouterAdaptive.cc.

References NetworkComponent::address, decoders, in_buffers, NetworkComponent::node_ip, swa, GenericCrossbar::toString(), PToPSwitchArbiter::toString(), and xbar.

Referenced by do_switch_traversal(), and handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.46.4 Member Data Documentation

7.46.4.1 vector<uint> GenericRouterAdaptive::available_ports [private]

Definition at line 73 of file genericRouterAdaptive.h.

Referenced by handle_link_arrival_event(), and set_edge_links().

7.46.4.2 vector<GenericRC> GenericRouterAdaptive::decoders [private]

Definition at line 69 of file genericRouterAdaptive.h.

Referenced by handle_link_arrival_event(), init(), set_grid_x_location(), set_grid_y_location(), set_no_nodes(), and toString().

7.46.4.3 vector< vector<uint> > GenericRouterAdaptive::downstream_credits

Definition at line 54 of file genericRouterAdaptive.h.

Referenced by do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), and init().

7.46.4.4 uint GenericRouterAdaptive::flits

Definition at line 58 of file genericRouterAdaptive.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.46.4.5 vector<GenericBuffer> GenericRouterAdaptive::in_buffers [private]

Definition at line 68 of file genericRouterAdaptive.h.

Referenced by do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), init(), and toString().

7.46.4.6 vector<InputBufferState> GenericRouterAdaptive::input_buffer_state [private]

Definition at line 70 of file genericRouterAdaptive.h.

Referenced by do_switch_allocation(), do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), init(), and send_credit_back().

7.46.4.7 double GenericRouterAdaptive::last_flit_out_cycle

Definition at line 60 of file genericRouterAdaptive.h.

Referenced by do_switch_traversal(), and print_stats().

7.46.4.8 uint GenericRouterAdaptive::packets

Definition at line 57 of file genericRouterAdaptive.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.46.4.9 vector< vector<uint> > GenericRouterAdaptive::stat_flit_out

Definition at line 63 of file genericRouterAdaptive.h.

Referenced by do_switch_traversal(), init(), and print_stats().

7.46.4.10 vector< vector<uint> > GenericRouterAdaptive::stat_packet_out

Definition at line 62 of file genericRouterAdaptive.h.

Referenced by do_switch_traversal(), init(), and print_stats().

7.46.4.11 unsigned long long int GenericRouterAdaptive::stat_sim_total_time

Definition at line 61 of file genericRouterAdaptive.h.

7.46.4.12 PToPSwitchArbiter GenericRouterAdaptive::swa [private]

Definition at line 71 of file genericRouterAdaptive.h.

Referenced by do_switch_allocation(), do_switch_traversal(), handle_tick_event(), init(), and toString().

7.46.4.13 bool GenericRouterAdaptive::ticking [private]

Definition at line 75 of file genericRouterAdaptive.h.

Referenced by do_switch_allocation(), do_switch_traversal(), GenericRouterAdaptive(), handle_link_arrival_event(), and handle_tick_event().

7.46.4.14 double GenericRouterAdaptive::total_packet_latency

Definition at line 59 of file genericRouterAdaptive.h.

Referenced by do_switch_traversal(), init(), and print_stats().

7.46.4.15 GenericCrossbar GenericRouterAdaptive::xbar [private]

Definition at line 72 of file genericRouterAdaptive.h.

Referenced by do_switch_allocation(), do_switch_traversal(), init(), and toString().

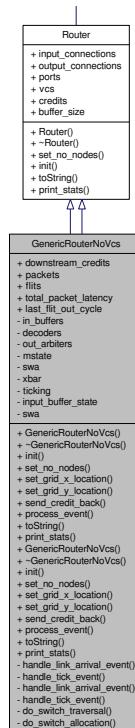
The documentation for this class was generated from the following files:

- **genericRouterAdaptive.h**
- **genericRouterAdaptive.cc**

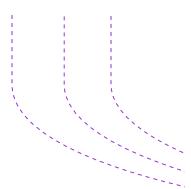
7.47 GenericRouterNoVcs Class Reference

```
#include <genericRouterNoVcs.h>
```

Inheritance diagram for GenericRouterNoVcs:



Collaboration diagram for GenericRouterNoVcs:



Public Member Functions

- **GenericRouterNoVcs ()**

- `~GenericRouterNoVcs ()`
- `void init (uint ports, uint vcs, uint credits, uint buffer_size)`
- `void set_no_nodes (uint nodes)`

These functions are mainly for DOR routing and are separated so as to not force DOR modelling in all designs.

- `void set_grid_x_location (uint a, uint b, uint c)`
- `void set_grid_y_location (uint a, uint b, uint c)`
- `void send_credit_back (uint i)`
- `void process_event (IrisEvent *e)`
- `string toString () const`
- `string print_stats ()`
- `GenericRouterNoVcs ()`
- `~GenericRouterNoVcs ()`
- `void init (uint ports, uint vcs, uint credits, uint buffer_size)`
- `void set_no_nodes (uint nodes)`
- `void set_grid_x_location (uint a, uint b, uint c)`
- `void set_grid_y_location (uint a, uint b, uint c)`
- `void send_credit_back (uint i)`
- `void process_event (IrisEvent *e)`
- `string toString () const`
- `string print_stats ()`

Public Attributes

- `vector< vector< uint > > downstream_credits`
- `uint packets`
- `uint flits`
- `double total_packet_latency`
- `double last_flit_out_cycle`

Private Member Functions

- `void handle_link_arrival_event (IrisEvent *e)`

Event handle for the LINK_ARRIVAL_EVENT event. Entry from DES kernel.
- `void handle_tick_event (IrisEvent *e)`

Event handle for the TICK_EVENT. Entry from DES kernel.
- `void handle_link_arrival_event (IrisEvent *e)`
- `void handle_tick_event (IrisEvent *e)`
- `void do_switch_traversal ()`
- `void do_switch_allocation ()`

Private Attributes

- `vector< GenericBuffer > in_buffers`
- `vector< GenericRC > decoders`
- `vector< GenericVcArbiter > out_arbiters`
- `vector< MessageState > mstate`
- `MyArbiter swa`
- `GenericCrossbar xbar`
- `bool ticking`
- `vector< InputBufferState > input_buffer_state`
- `PToPSwitchArbiter swa`

7.47.1 Detailed Description

Definition at line 55 of file backup/genericRouterNoVcs.h.

7.47.2 Constructor & Destructor Documentation

7.47.2.1 GenericRouterNoVcs::GenericRouterNoVcs ()

Definition at line 47 of file backup/genericRouterNoVcs.cc.

References NetworkComponent::name, and ticking.

7.47.2.2 GenericRouterNoVcs::~GenericRouterNoVcs ()

Definition at line 53 of file backup/genericRouterNoVcs.cc.

7.47.2.3 GenericRouterNoVcs::GenericRouterNoVcs ()

7.47.2.4 GenericRouterNoVcs::~GenericRouterNoVcs ()

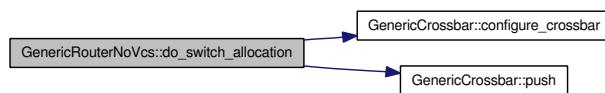
7.47.3 Member Function Documentation

7.47.3.1 void GenericRouterNoVcs::do_switch_allocation () [private]

Definition at line 367 of file genericRouterNoVcs.cc.

References _DBG, GenericCrossbar::configure_crossbar(), input_buffer_state, SA_unit::port, GenericCrossbar::push(), ST, swa, SWA_REQUESTED, ticking, and xbar.

Here is the call graph for this function:

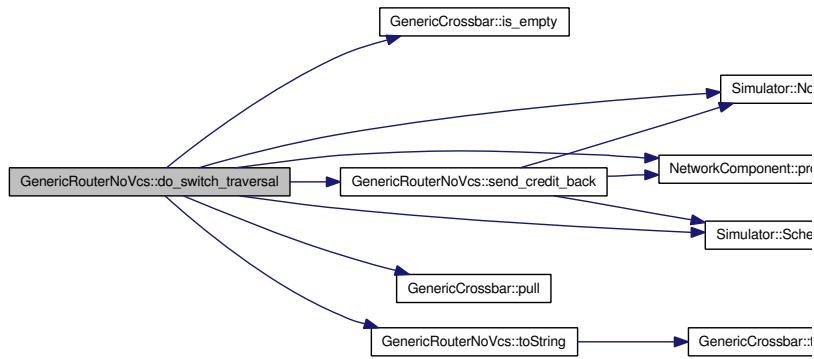


7.47.3.2 void GenericRouterNoVcs::do_switch_traversal () [private]

Definition at line 267 of file genericRouterNoVcs.cc.

References _DBG, _DBG_NOARG, NetworkComponent::address, BODY, downstream_credits, EMPTY, FLIT_ID, HEAD, in_buffers, input_buffer_state, GenericCrossbar::is_empty(), LINK_ARRIVAL_EVENT, Simulator::Now(), ONE_FLIT_REQ, Router::output_connections, NetworkComponent::process_event(), LinkArrivalData::ptr, GenericCrossbar::pull(), Simulator::Schedule(), send_credit_back(), ST, swa, TAIL, toString(), total_packet_latency, Flit::type, LinkArrivalData::type, LinkArrivalData::vc, and xbar.

Here is the call graph for this function:



7.47.3.3 void GenericRouterNoVcs::handle_link_arrival_event (IrisEvent * e) [private]

7.47.3.4 void GenericRouterNoVcs::handle_link_arrival_event (IrisEvent * e) [private]

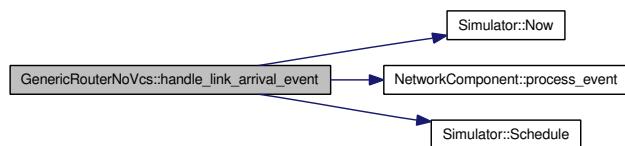
Event handle for the LINK_ARRIVAL_EVENT event. Entry from DES kernel.

Definition at line 171 of file backup/genericRouterNoVcs.cc.

References _DBG, CREDIT_ID, decoders, downstream_credits, IrisEvent::event_data, FLIT_ID, flits, FULL, HEAD, in_buffers, Router::input_connections, HeadFlit::length, mstate, Simulator::Now(), Router::output_connections, packets, Router::ports, NetworkComponent::process_event(), LinkArrivalData::ptr, Simulator::Schedule(), IrisEvent::src_id, TAIL, TICK_EVENT, ticking, Flit::type, LinkArrivalData::type, IrisEvent::vc, LinkArrivalData::vc, and Router::vcs.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.47.3.5 void GenericRouterNoVcs::handle_link_arrival_event (IrisEvent * e) [private]

7.47.3.6 void GenericRouterNoVcs::handle_tick_event (IrisEvent * e) [private]

Event handle for the TICK_EVENT. Entry from DES kernel.

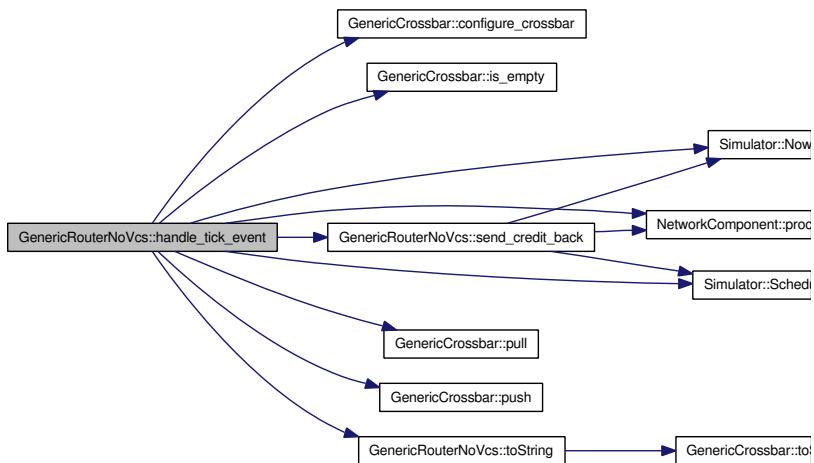
Body and tail flits get written in link arrival and since the message state may already been pushed to ST because of the header we want to ensure that all flits go thru an IB and ST stage. Hence ST is done on the flits_in_ib information and not buffer occupancy.

Sending credits back for body+tail: Condition being HEAD in ST and having downstream credits
Definition at line 276 of file backup/genericRouterNoVcs.cc.

References _DBG, _DBG_NOARG, NetworkComponent::address, BODY, GenericCrossbar::configure_crossbar(), downstream_credits, EMPTY, FLIT_ID, FULL, HEAD, IB, in_buffers, GenericCrossbar::is_empty(), last_flit_out_cycle, LINK_ARRIVAL_EVENT, mstate, Simulator::Now(), Router::output_connections, Router::ports, NetworkComponent::process_event(), LinkArrivalData::ptr, GenericCrossbar::pull(), GenericCrossbar::push(), Simulator::Schedule(), send_credit_back(), ST, swa, SWA_REQUESTED, TAIL, TICK_EVENT, ticking, toString(), total_packet_latency, LinkArrivalData::type, Flit::type, IrisEvent::vc, LinkArrivalData::vc, Router::vcs, and xbar.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.47.3.7 void GenericRouterNoVcs::init (uint *ports*, uint *vcs*, uint *credits*, uint *buffer_size*) [virtual]

Implements **Router** (p. 328).

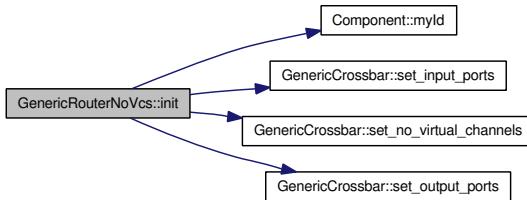
7.47.3.8 void GenericRouterNoVcs::init (uint *ports*, uint *vcs*, uint *credits*, uint *buffer_size*) [virtual]

Implements **Router** (p. 328).

Definition at line 58 of file backup/genericRouterNoVcs.cc.

References _DBG, NetworkComponent::address, Router::buffer_size, Router::credits, decoders, downstream_credits, flits, in_buffers, mstate, Component::myId(), NetworkComponent::node_ip, packets, Router::ports, GenericCrossbar::set_input_ports(), GenericCrossbar::set_no_virtual_channels(), GenericCrossbar::set_output_ports(), swa, total_packet_latency, Router::vcs, and xbar.

Here is the call graph for this function:



7.47.3.9 string GenericRouterNoVcs::print_stats () [virtual]

Implements **Router** (p. 329).

7.47.3.10 string GenericRouterNoVcs::print_stats () [virtual]

Implements **Router** (p. 329).

Definition at line 156 of file backup/genericRouterNoVcs.cc.

References flits, last_flit_out_cycle, NetworkComponent::node_ip, packets, and total_packet_latency.

7.47.3.11 void GenericRouterNoVcs::process_event (IrisEvent * *e*) [virtual]

Implements **NetworkComponent** (p. 274).

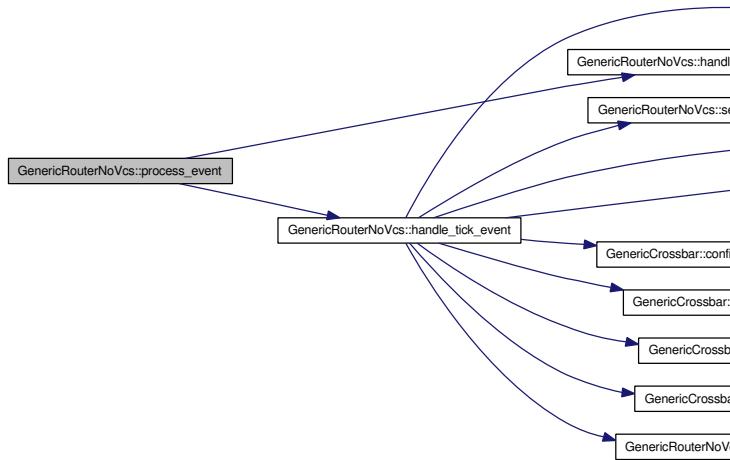
7.47.3.12 void GenericRouterNoVcs::process_event (IrisEvent * *e*) [virtual]

Implements **NetworkComponent** (p. 274).

Definition at line 138 of file backup/genericRouterNoVcs.cc.

References _DBG, handle_link_arrival_event(), handle_tick_event(), LINK_ARRIVAL_EVENT, TICK_EVENT, and IrisEvent::type.

Here is the call graph for this function:



7.47.3.13 void GenericRouterNoVcs::send_credit_back (uint i)

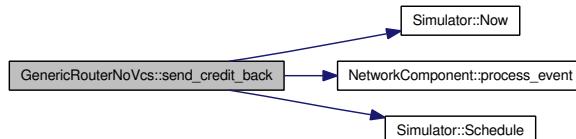
7.47.3.14 void GenericRouterNoVcs::send_credit_back (uint i)

Definition at line 582 of file backup/genericRouterNoVcs.cc.

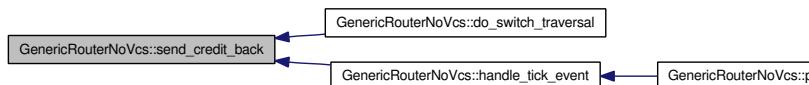
References `DBG`, `NetworkComponent::address`, `CREDIT_ID`, `Router::input_connections`, `LINK_ARRIVAL_EVENT`, `mstate`, `Simulator::Now()`, `VirtualChannelDescription::port`, `NetworkComponent::process_event()`, `Simulator::Schedule()`, `LinkArrivalData::type`, `LinkArrivalData::vc`, and `VirtualChannelDescription::vc`.

Referenced by `do_switch_traversal()`, and `handle_tick_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.47.3.15 void GenericRouterNoVcs::set_grid_x_location (uint a, uint b, uint c)

7.47.3.16 void GenericRouterNoVcs::set_grid_x_location (uint a, uint b, uint c)

Definition at line 124 of file backup/genericRouterNoVcs.cc.

References decoders.

7.47.3.17 void GenericRouterNoVcs::set_grid_y_location (uint a, uint b, uint c)

7.47.3.18 void GenericRouterNoVcs::set_grid_y_location (uint a, uint b, uint c)

Definition at line 130 of file backup/genericRouterNoVcs.cc.

References decoders.

7.47.3.19 void GenericRouterNoVcs::set_no_nodes (uint nodes) [virtual]

Implements **Router** (p. 329).

7.47.3.20 void GenericRouterNoVcs::set_no_nodes (uint nodes) [virtual]

These functions are mainly for DOR routing and are separated so as to not force DOR modelling in all designs.

Implements **Router** (p. 329).

Definition at line 114 of file backup/genericRouterNoVcs.cc.

References decoders.

7.47.3.21 string GenericRouterNoVcs::toString () const [virtual]

Reimplemented from **Router** (p. 329).

7.47.3.22 string GenericRouterNoVcs::toString () const [virtual]

Reimplemented from **Router** (p. 329).

Definition at line 516 of file backup/genericRouterNoVcs.cc.

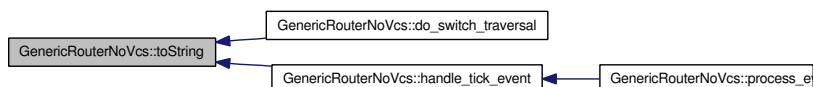
References NetworkComponent::address, decoders, in_buffers, NetworkComponent::node_ip, swa, GenericCrossbar::toString(), and xbar.

Referenced by do_switch_traversal(), and handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.47.4 Member Data Documentation

7.47.4.1 `vector< GenericRC > GenericRouterNoVcs::decoders [private]`

Definition at line 83 of file backup/genericRouterNoVcs.h.

Referenced by handle_link_arrival_event(), init(), set_grid_x_location(), set_grid_y_location(), set_no_nodes(), and toString().

7.47.4.2 `vector< vector< uint > > GenericRouterNoVcs::downstream_credits`

Definition at line 71 of file backup/genericRouterNoVcs.h.

Referenced by do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), and init().

7.47.4.3 `uint GenericRouterNoVcs::flits`

Definition at line 75 of file backup/genericRouterNoVcs.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.47.4.4 `vector< GenericBuffer > GenericRouterNoVcs::in_buffers [private]`

Definition at line 82 of file backup/genericRouterNoVcs.h.

Referenced by do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), init(), and toString().

7.47.4.5 `vector<InputBufferState> GenericRouterNoVcs::input_buffer_state [private]`

Definition at line 64 of file genericRouterNoVcs.h.

Referenced by do_switch_allocation(), and do_switch_traversal().

7.47.4.6 `double GenericRouterNoVcs::last_flit_out_cycle`

Definition at line 77 of file backup/genericRouterNoVcs.h.

Referenced by handle_tick_event(), and print_stats().

7.47.4.7 `vector<MessageState> GenericRouterNoVcs::mstate [private]`

Definition at line 85 of file backup/genericRouterNoVcs.h.

Referenced by handle_link_arrival_event(), handle_tick_event(), init(), and send_credit_back().

7.47.4.8 `vector<GenericVcArbiter> GenericRouterNoVcs::out_arbiters [private]`

Definition at line 84 of file backup/genericRouterNoVcs.h.

7.47.4.9 uint GenericRouterNoVcs::packets

Definition at line 74 of file backup/genericRouterNoVcs.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.47.4.10 PToPSwitchArbiter GenericRouterNoVcs::swa [private]

Definition at line 65 of file genericRouterNoVcs.h.

7.47.4.11 MyArbiter GenericRouterNoVcs::swa [private]

Definition at line 86 of file backup/genericRouterNoVcs.h.

Referenced by do_switch_allocation(), do_switch_traversal(), handle_tick_event(), init(), and toString().

7.47.4.12 bool GenericRouterNoVcs::ticking [private]

Definition at line 89 of file backup/genericRouterNoVcs.h.

Referenced by do_switch_allocation(), GenericRouterNoVcs(), handle_link_arrival_event(), and handle_tick_event().

7.47.4.13 double GenericRouterNoVcs::total_packet_latency

Definition at line 76 of file backup/genericRouterNoVcs.h.

Referenced by do_switch_traversal(), handle_tick_event(), init(), and print_stats().

7.47.4.14 GenericCrossbar GenericRouterNoVcs::xbar [private]

Definition at line 87 of file backup/genericRouterNoVcs.h.

Referenced by do_switch_allocation(), do_switch_traversal(), handle_tick_event(), init(), and toString().

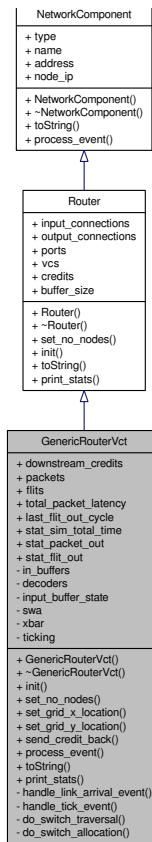
The documentation for this class was generated from the following files:

- [backup/genericRouterNoVcs.h](#)
- [genericRouterNoVcs.h](#)
- [backup/genericRouterNoVcs.cc](#)
- [genericRouterNoVcs.cc](#)

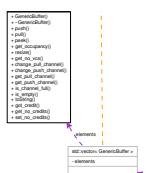
7.48 GenericRouterVct Class Reference

```
#include <genericRouterVct.h>
```

Inheritance diagram for GenericRouterVct:



Collaboration diagram for GenericRouterVct:



Public Member Functions

- `GenericRouterVct ()`
- `~GenericRouterVct ()`
- `void init (uint ports, uint vcs, uint credits, uint buffer_size)`

- void **set_no_nodes** (uint nodes)

These functions are mainly for DOR routing and are separated so as to not force DOR modelling in all designs.

- void **set_grid_x_location** (uint a, uint b, uint c)
- void **set_grid_y_location** (uint a, uint b, uint c)
- void **send_credit_back** (uint i)
- void **process_event** (IrisEvent *e)
- string **toString** () const
- string **print_stats** ()

Public Attributes

- vector< vector< uint > > **downstream_credits**
- uint **packets**
- uint **flits**
- double **total_packet_latency**
- double **last_flit_out_cycle**
- unsigned long long int **stat_sim_total_time**
- vector< vector< uint > > **stat_packet_out**
- vector< vector< uint > > **stat_flit_out**

Private Member Functions

- void **handle_link_arrival_event** (IrisEvent *e)

Event handle for the LINK_ARRIVAL_EVENT event. Entry from DES kernel.
- void **handle_tick_event** (IrisEvent *e)

Event handle for the TICK_EVENT. Entry from DES kernel.
- void **do_switch_traversal** ()
- void **do_switch_allocation** ()

Private Attributes

- vector< GenericBuffer > **in_buffers**
- vector< GenericRC > **decoders**
- vector< InputBufferState > **input_buffer_state**
- PToPSwitchArbiter **swa**
- GenericCrossbar **xbar**
- bool **ticking**

7.48.1 Detailed Description

Definition at line 36 of file genericRouterVct.h.

7.48.2 Constructor & Destructor Documentation

7.48.2.1 GenericRouterVct::GenericRouterVct ()

Definition at line 39 of file genericRouterVct.cc.

References NetworkComponent::name, and ticking.

7.48.2.2 GenericRouterVct::~GenericRouterVct ()

Definition at line 45 of file genericRouterVct.cc.

7.48.3 Member Function Documentation

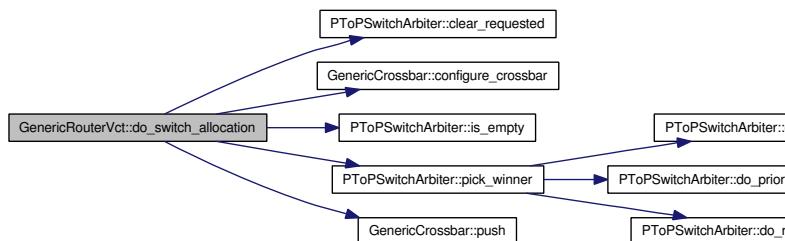
7.48.3.1 void GenericRouterVct::do_switch_allocation () [private]

Definition at line 458 of file genericRouterVct.cc.

References _DBG, PToPSwitchArbiter::clear_requested(), GenericCrossbar::configure_crossbar(), FULL, input_buffer_state, PToPSwitchArbiter::is_empty(), PToPSwitchArbiter::pick_winner(), SA_unit::port, Router::ports, GenericCrossbar::push(), ST, swa, SWA_REQUESTED, ticking, Router::vcs, and xbar.

Referenced by handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.48.3.2 void GenericRouterVct::do_switch_traversal () [private]

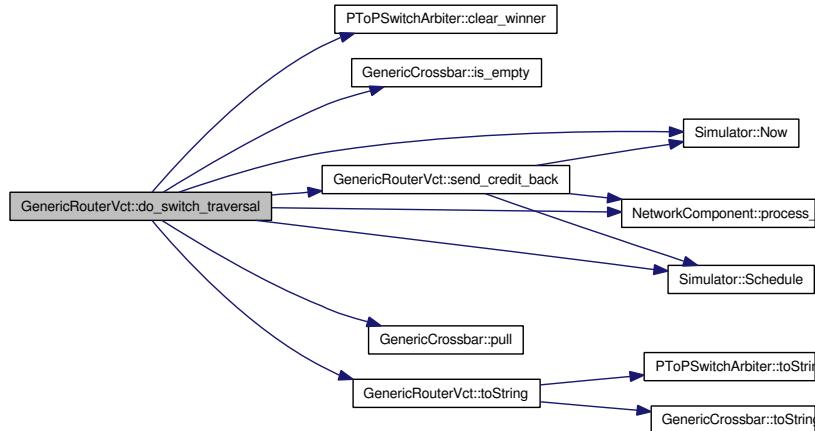
Definition at line 331 of file genericRouterVct.cc.

References _DBG, _DBG_NOARG, NetworkComponent::address, TailFlit::avg_network_latency, HeadFlit::avg_network_latency, BODY, PToPSwitchArbiter::clear_winner(), do_two_stage_router, downstream_credits, EMPTY, FLIT_ID, HEAD, TailFlit::hop_count, HeadFlit::hop_count, in_buffers, input_buffer_state, GenericCrossbar::is_empty(), last_flit_out_cycle, LINK_ARRIVAL_EVENT, Simulator::Now(), ONE_FLIT_REQ, Router::output_connections, Router::ports, NetworkComponent::process_event(), LinkArrivalData::ptr, GenericCrossbar::pull(), Simulator::Schedule(), send_credit_back(), ST, stat_flit_out, stat_packet_

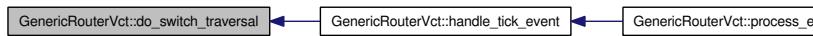
out, swa, TAIL, ticking, `toString()`, `total_packet_latency`, `Flit::type`, `LinkArrivalData::type`, `LinkArrivalData::vc`, `Router::vcs`, and `xbar`.

Referenced by `handle_tick_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.48.3.3 void GenericRouterVct::handle_link_arrival_event (IrisEvent * e) [private]

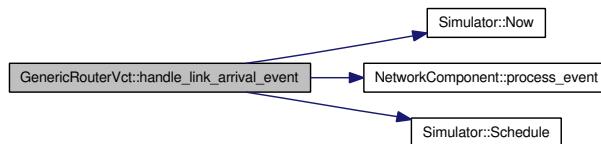
Event handle for the `LINK_ARRIVAL_EVENT` event. Entry from DES kernel.

Definition at line 223 of file `genericRouterVct.cc`.

References `_DBG`, `CREDIT_ID`, decoders, `downstream_credits`, `IrisEvent::event_data`, `FLIT_ID`, flits, `FULL`, `HEAD`, `in_buffers`, `input_buffer_state`, `Router::input_connections`, `HeadFlit::length`, `Simulator::Now()`, `ONE_FLIT_REQ`, `Router::output_connections`, packets, `Router::ports`, `NetworkComponent::process_event()`, `LinkArrivalData::ptr`, `Simulator::Schedule()`, `IrisEvent::src_id`, `TAIL`, `TICK_EVENT`, `ticking`, `Flit::type`, `LinkArrivalData::type`, `IrisEvent::vc`, `LinkArrivalData::vc`, and `Router::vcs`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.48.3.4 void GenericRouterVct::handle_tick_event (IrisEvent * e) [private]

Event handle for the TICK_EVENT. Entry from DES kernel.

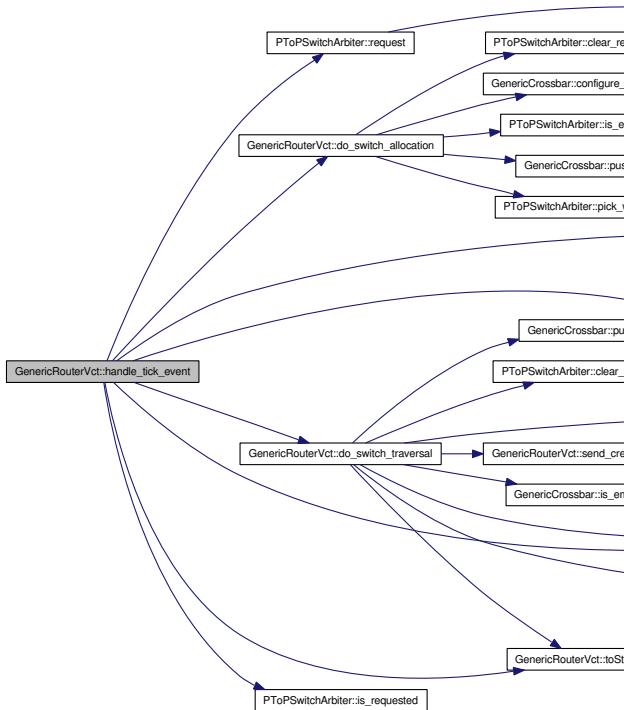
Body and tail flits get written in link arrival and since the message state may already been pushed to ST because of the header we want to ensure that all flits go thru an IB and ST stage. Hence ST is done on the flits_in_ib information and not buffer occupancy.

Definition at line 515 of file genericRouterVct.cc.

References _DBG, _DBG_NOARG, Router::credits, do_switch_allocation(), do_switch_traversal(), downstream_credits, FULL, IB, in_buffers, input_buffer_state, PToPSwitchArbiter::is_requested(), Simulator::Now(), Router::ports, NetworkComponent::process_event(), PToPSwitchArbiter::request(), Simulator::Schedule(), ST, swa, SWA_REQUESTED, TICK_EVENT, ticking, toString(), IrisEvent::vc, and Router::vcs.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



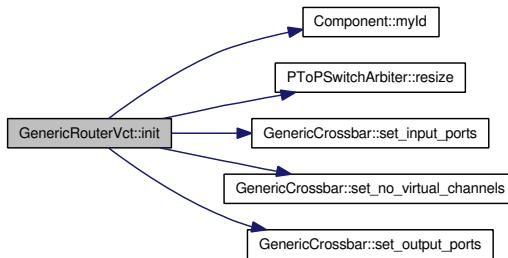
7.48.3.5 void GenericRouterVct::init (uint ports, uint vcs, uint credits, uint buffer_size) [virtual]

Implements **Router** (p. 328).

Definition at line 50 of file genericRouterVct.cc.

References NetworkComponent::address, Router::buffer_size, Router::credits, decoders, downstream_credits, flits, in_buffers, input_buffer_state, Component::myId(), NetworkComponent::node_ip, packets, Router::ports, PToPSwitchArbiter::resize(), GenericCrossbar::set_input_ports(), GenericCrossbar::set_no_virtual_channels(), GenericCrossbar::set_output_ports(), stat_flit_out, stat_packet_out, swa, total_packet_latency, Router::vcs, and xbar.

Here is the call graph for this function:



7.48.3.6 string GenericRouterVct::print_stats () [virtual]

Implements **Router** (p. 329).

Definition at line 149 of file genericRouterVct.cc.

References flits, last_flit_out_cycle, NetworkComponent::node_ip, packets, Router::ports, stat_flit_out, stat_packet_out, and total_packet_latency.

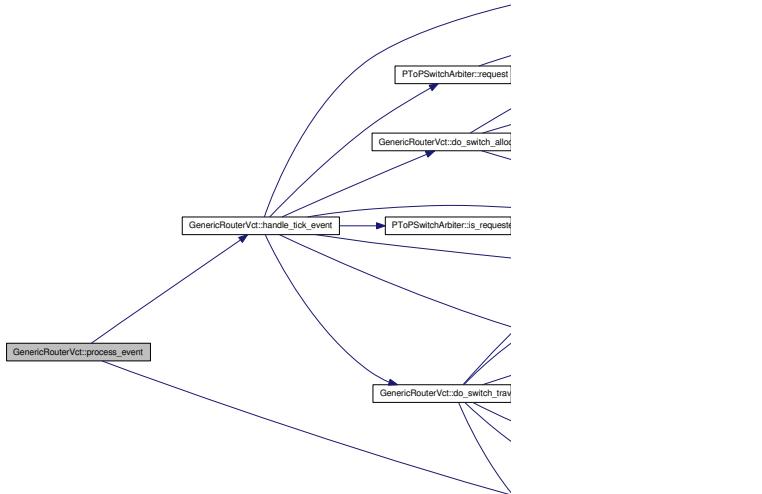
7.48.3.7 void GenericRouterVct::process_event (IrisEvent * e) [virtual]

Implements **NetworkComponent** (p. 274).

Definition at line 131 of file genericRouterVct.cc.

References _DBG, handle_link_arrival_event(), handle_tick_event(), LINK_ARRIVAL_EVENT, TICK_EVENT, and IrisEvent::type.

Here is the call graph for this function:



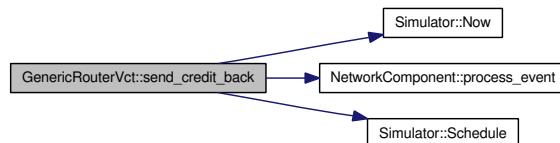
7.48.3.8 void GenericRouterVct::send_credit_back (uint i)

Definition at line 664 of file genericRouterVct.cc.

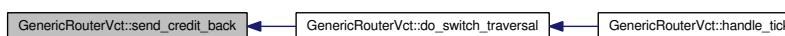
References `_DBG`, `NetworkComponent::address`, `CREDIT_ID`, `do_two_stage_router`, `input_buffer_state`, `Router::input_connections`, `LINK_ARRIVAL_EVENT`, `Simulator::Now()`, `NetworkComponent::process_event()`, `Simulator::Schedule()`, `LinkArrivalData::type`, and `LinkArrivalData::vc`.

Referenced by `do_switch_traversal()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.48.3.9 void GenericRouterVct::set_grid_x_location (uint a, uint b, uint c)

Definition at line 117 of file genericRouterVct.cc.

References decoders.

7.48.3.10 void GenericRouterVct::set_grid_y_location (uint a, uint b, uint c)

Definition at line 123 of file genericRouterVct.cc.

References decoders.

7.48.3.11 void GenericRouterVct::set_no_nodes (uint nodes) [virtual]

These functions are mainly for DOR routing and are separated so as to not force DOR modelling in all designs.

Implements **Router** (p. 329).

Definition at line 107 of file genericRouterVct.cc.

References decoders.

7.48.3.12 string GenericRouterVct::toString () const [virtual]

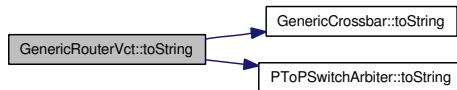
Reimplemented from **Router** (p. 329).

Definition at line 641 of file genericRouterVct.cc.

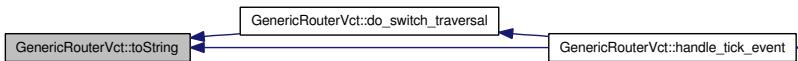
References NetworkComponent::address, decoders, in_buffers, NetworkComponent::node_ip, swa, GenericCrossbar::toString(), PToPSwitchArbiter::toString(), and xbar.

Referenced by do_switch_traversal(), and handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:

**7.48.4 Member Data Documentation****7.48.4.1 vector<GenericRC> GenericRouterVct::decoders [private]**

Definition at line 67 of file genericRouterVct.h.

Referenced by handle_link_arrival_event(), init(), set_grid_x_location(), set_grid_y_location(), set_no_nodes(), and toString().

7.48.4.2 vector< vector<uint> > GenericRouterVct::downstream_credits

Definition at line 52 of file genericRouterVct.h.

Referenced by do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), and init().

7.48.4.3 uint GenericRouterVct::flits

Definition at line 56 of file genericRouterVct.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.48.4.4 vector<GenericBuffer> GenericRouterVct::in_buffers [private]

Definition at line 66 of file genericRouterVct.h.

Referenced by do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), init(), and toString().

7.48.4.5 vector<InputBufferState> GenericRouterVct::input_buffer_state [private]

Definition at line 68 of file genericRouterVct.h.

Referenced by do_switch_allocation(), do_switch_traversal(), handle_link_arrival_event(), handle_tick_event(), init(), and send_credit_back().

7.48.4.6 double GenericRouterVct::last_flit_out_cycle

Definition at line 58 of file genericRouterVct.h.

Referenced by do_switch_traversal(), and print_stats().

7.48.4.7 uint GenericRouterVct::packets

Definition at line 55 of file genericRouterVct.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.48.4.8 vector< vector<uint> > GenericRouterVct::stat_flit_out

Definition at line 61 of file genericRouterVct.h.

Referenced by do_switch_traversal(), init(), and print_stats().

7.48.4.9 vector< vector<uint> > GenericRouterVct::stat_packet_out

Definition at line 60 of file genericRouterVct.h.

Referenced by do_switch_traversal(), init(), and print_stats().

7.48.4.10 unsigned long long int GenericRouterVct::stat_sim_total_time

Definition at line 59 of file genericRouterVct.h.

7.48.4.11 PToPSwitchArbiter GenericRouterVct::swa [private]

Definition at line 69 of file genericRouterVct.h.

Referenced by do_switch_allocation(), do_switch_traversal(), handle_tick_event(), init(), and toString().

7.48.4.12 bool GenericRouterVct::ticking [private]

Definition at line 72 of file genericRouterVct.h.

Referenced by do_switch_allocation(), do_switch_traversal(), GenericRouterVct(), handle_link_arrival_event(), and handle_tick_event().

7.48.4.13 double GenericRouterVct::total_packet_latency

Definition at line 57 of file genericRouterVct.h.

Referenced by do_switch_traversal(), init(), and print_stats().

7.48.4.14 GenericCrossbar GenericRouterVct::xbar [private]

Definition at line 70 of file genericRouterVct.h.

Referenced by do_switch_allocation(), do_switch_traversal(), init(), and toString().

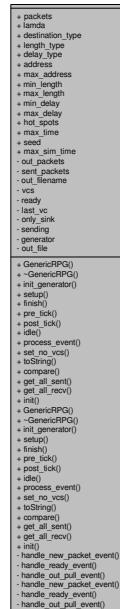
The documentation for this class was generated from the following files:

- **genericRouterVct.h**
- **genericRouterVct.cc**

7.49 GenericRPG Class Reference

```
#include <genericRPG.h>
```

Inheritance diagram for GenericRPG:



Collaboration diagram for GenericRPG:

Public Member Functions

- **GenericRPG ()**
- **~GenericRPG ()**
- **void init_generator ()**
- **void setup ()**
- **void finish ()**
- **void pre_tick ()**
- **void post_tick ()**
- **void idle ()**
- **void process_event (IrisEvent *e)**
- **void set_no_vcs (uint v)**
- **string toString () const**
- **bool compare ()**
- **set< HighLevelPacket > get_all_sent ()**
- **set< HighLevelPacket > get_all_recv ()**
- **void init ()**
- **GenericRPG ()**
- **~GenericRPG ()**
- **void init_generator ()**
- **void setup (uint no_nodes, uint vcs, uint max_sim_time)**
- **void finish ()**
- **void pre_tick ()**
- **void post_tick ()**
- **void idle ()**
- **void process_event (IrisEvent *e)**
- **void set_no_vcs (uint v)**
- **string toString () const**
- **bool compare ()**
- **set< HighLevelPacket > get_all_sent ()**
- **set< HighLevelPacket > get_all_recv ()**
- **void init ()**

Public Attributes

- **unsigned int packets**
- **double lamda**
- **libRandom::randomNumberGenerator::distribution destination_type**
- **libRandom::randomNumberGenerator::distribution length_type**
- **libRandom::randomNumberGenerator::distribution delay_type**
- **unsigned int address**
- **unsigned int max_address**
- **unsigned int min_length**
- **unsigned int max_length**
- **unsigned int min_delay**
- **unsigned int max_delay**
- **unsigned int hot_spots**
- **unsigned long long int max_time**
- **unsigned int seed**
- **uint max_sim_time**

Private Member Functions

- void **handle_new_packet_event** (IrisEvent *e)
- void **handle_ready_event** (IrisEvent *e)
- void **handle_out_pull_event** (IrisEvent *e)
- void **handle_new_packet_event** (IrisEvent *e)
- void **handle_ready_event** (IrisEvent *e)
- void **handle_out_pull_event** (IrisEvent *e)

Private Attributes

- deque< **HighLevelPacket** > **out_packets**
- deque< **HighLevelPacket** > **sent_packets**
- string **out_filename**
- uint **vcs**
- vector< bool > **ready**
- unsigned int **last_vc**
- bool **only_sink**
- bool **sending**
- libRandom::randomNumberGenerator **generator**
- ofstream **out_file**

7.49.1 Detailed Description

Definition at line 45 of file backup/genericRPG.h.

7.49.2 Constructor & Destructor Documentation

7.49.2.1 GenericRPG::GenericRPG ()

Definition at line 7 of file backup/genericRPG.cc.

References Processor::interface_connections, last_vc, NetworkComponent::name, seed, and sending.

7.49.2.2 GenericRPG::~GenericRPG ()

Definition at line 18 of file backup/genericRPG.cc.

7.49.2.3 GenericRPG::GenericRPG ()

7.49.2.4 GenericRPG::~GenericRPG ()

7.49.3 Member Function Documentation

7.49.3.1 bool GenericRPG::compare ()

7.49.3.2 bool GenericRPG::compare ()

7.49.3.3 void GenericRPG::finish ()

7.49.3.4 void GenericRPG::finish ()

Definition at line 119 of file backup/genericRPG.cc.

7.49.3.5 set< HighLevelPacket > GenericRPG::get_all_recv ()

7.49.3.6 set< HighLevelPacket > GenericRPG::get_all_recv ()

7.49.3.7 set< HighLevelPacket > GenericRPG::get_all_sent ()

7.49.3.8 set< HighLevelPacket > GenericRPG::get_all_sent ()

7.49.3.9 void GenericRPG::handle_new_packet_event (IrisEvent * e) [private]

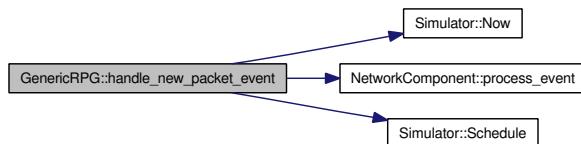
7.49.3.10 void GenericRPG::handle_new_packet_event (IrisEvent * e) [private]

Definition at line 147 of file backup/genericRPG.cc.

References `_DBG`, `IrisEvent::event_data`, `Processor::interface_connections`, `Simulator::Now()`, `NetworkComponent::process_event()`, `READY_EVENT`, `Simulator::Schedule()`, `HighLevelPacket::sent_time`, `VirtualChannelDescription::vc`, and `HighLevelPacket::virtual_channel`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.49.3.11 void GenericRPG::handle_out_pull_event (IrisEvent * e) [private]

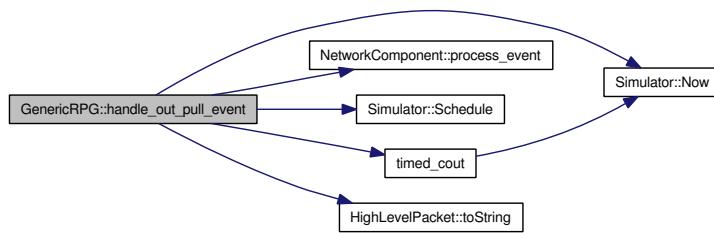
7.49.3.12 void GenericRPG::handle_out_pull_event (IrisEvent * e) [private]

Definition at line 181 of file backup/genericRPG.cc.

References address, HighLevelPacket::data, HighLevelPacket::data_payload_length, HighLevelPacket::destination, generator, Processor::interface_connections, last_vc, max_phy_link_bits, MAX_SIM_TIME, NEW_PACKET_EVENT, NetworkComponent::node_ip, Simulator::Now(), only_sink, OUT_PULL_EVENT, packets, NetworkComponent::process_event(), ready, Simulator::Schedule(), seed, sending, HighLevelPacket::sent_time, HighLevelPacket::source, timed_cout(), HighLevelPacket::toString(), HighLevelPacket::transaction_id, IrisEvent::vc, and HighLevelPacket::virtual_channel.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.49.3.13 void GenericRPG::handle_ready_event (IrisEvent * e) [private]

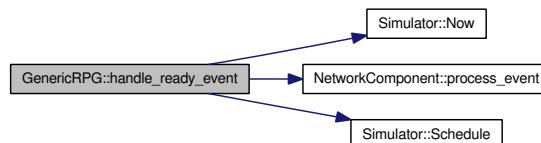
7.49.3.14 void GenericRPG::handle_ready_event (IrisEvent * e) [private]

Definition at line 273 of file backup/genericRPG.cc.

References _DBG_NOARG, IrisEvent::event_data, MAX_SIM_TIME, Simulator::Now(), OUT_PULL_EVENT, NetworkComponent::process_event(), ready, Simulator::Schedule(), sending, IrisEvent::vc, and VirtualChannelDescription::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.49.3.15 void GenericRPG::idle ()

7.49.3.16 void GenericRPG::idle ()

7.49.3.17 void GenericRPG::init ()

Reimplemented from **Processor** (p. 296).

7.49.3.18 void GenericRPG::init ()

Reimplemented from **Processor** (p. 296).

Definition at line 299 of file backup/genericRPG.cc.

7.49.3.19 void GenericRPG::init_generator ()

7.49.3.20 void GenericRPG::init_generator ()

Definition at line 23 of file backup/genericRPG.cc.

References delay_type, destination_type, generator, HOT_SPOTS, length_type, MAX_ADDRESS, MAX_DELAY, MAX_LENGTH, MIN_DELAY, MIN_LENGTH, and seed.

Referenced by setup().

Here is the caller graph for this function:



7.49.3.21 void GenericRPG::post_tick ()

7.49.3.22 void GenericRPG::post_tick ()

7.49.3.23 void GenericRPG::pre_tick ()

7.49.3.24 void GenericRPG::pre_tick ()

7.49.3.25 void GenericRPG::process_event (IrisEvent * e) [virtual]

Implements **Processor** (p. 297).

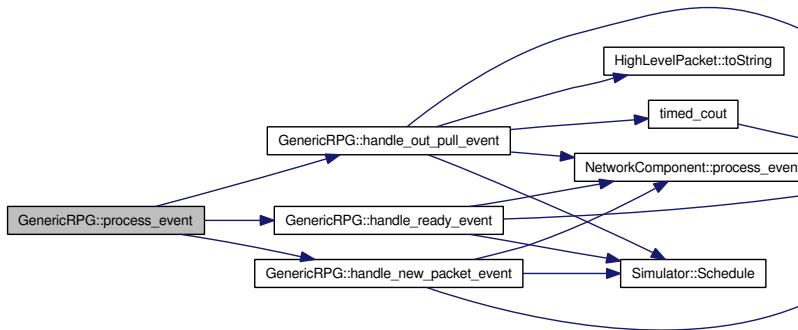
7.49.3.26 void GenericRPG::process_event (IrisEvent * e) [virtual]

Implements **Processor** (p. 297).

Definition at line 126 of file backup/genericRPG.cc.

References address, handle_new_packet_event(), handle_out_pull_event(), handle_ready_event(), NEW_PACKET_EVENT, OUT_PULL_EVENT, READY_EVENT, and IrisEvent::type.

Here is the call graph for this function:



7.49.3.27 void GenericRPG::set_no_vcs (uint v)

7.49.3.28 void GenericRPG::set_no_vcs (uint v)

Definition at line 42 of file backup/genericRPG.cc.

References vcs.

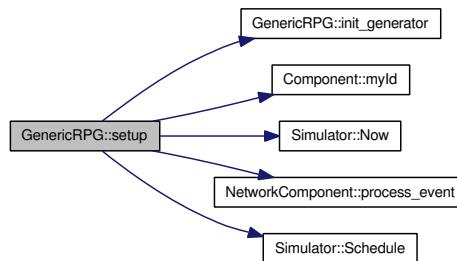
7.49.3.29 void GenericRPG::setup (uint no_nodes, uint vcs, uint max_sim_time) [virtual]

Implements **Processor** (p. 297).

Definition at line 48 of file genericRPG.cc.

References address, delay_type, destination_type, init_generator(), length_type, max_sim_time, Component::myId(), Simulator::Now(), only_sink, packets, NetworkComponent::process_event(), ready, READY_EVENT, run_destination_type, Simulator::Schedule(), VirtualChannelDescription::vc, and vcs.

Here is the call graph for this function:

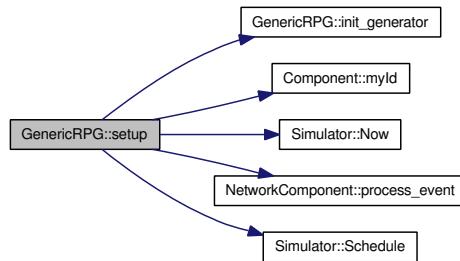


7.49.3.30 void GenericRPG::setup ()

Definition at line 48 of file backup/genericRPG.cc.

References address, delay_type, destination_type, init_generator(), length_type, Component::myId(), Simulator::Now(), only_sink, packets, NetworkComponent::process_event(), ready, READY_EVENT, run_destination_type, Simulator::Schedule(), VirtualChannelDescription::vc, and vcs.

Here is the call graph for this function:



7.49.3.31 string GenericRPG::toString () const [virtual]

Reimplemented from **Processor** (p. 297).

7.49.3.32 string GenericRPG::toString () const [virtual]

Reimplemented from **Processor** (p. 297).

Definition at line 306 of file backup/genericRPG.cc.

7.49.4 Member Data Documentation

7.49.4.1 unsigned int GenericRPG::address

Reimplemented from **NetworkComponent** (p. 275).

Definition at line 73 of file backup/genericRPG.h.

Referenced by handle_out_pull_event(), process_event(), and setup().

7.49.4.2 libRandom::randomNumberGenerator::distribution GenericRPG::delay_type

Definition at line 72 of file backup/genericRPG.h.

Referenced by init_generator(), and setup().

7.49.4.3 libRandom::randomNumberGenerator::distribution GenericRPG::destination_type

Definition at line 70 of file backup/genericRPG.h.

Referenced by init_generator(), and setup().

7.49.4.4 libRandom::randomNumberGenerator GenericRPG::generator [private]

Definition at line 61 of file backup/genericRPG.h.

Referenced by handle_out_pull_event(), and init_generator().

7.49.4.5 unsigned int GenericRPG::hot_spots

Definition at line 79 of file backup/genericRPG.h.

7.49.4.6 double GenericRPG::lamda

Definition at line 69 of file backup/genericRPG.h.

7.49.4.7 unsigned int GenericRPG::last_vc [private]

Definition at line 55 of file backup/genericRPG.h.

Referenced by GenericRPG(), and handle_out_pull_event().

7.49.4.8 libRandom::randomNumberGenerator::distribution GenericRPG::length_type

Definition at line 71 of file backup/genericRPG.h.

Referenced by init_generator(), and setup().

7.49.4.9 unsigned int GenericRPG::max_address

Definition at line 74 of file backup/genericRPG.h.

7.49.4.10 unsigned int GenericRPG::max_delay

Definition at line 78 of file backup/genericRPG.h.

7.49.4.11 unsigned int GenericRPG::max_length

Definition at line 76 of file backup/genericRPG.h.

7.49.4.12 uint GenericRPG::max_sim_time

Definition at line 68 of file genericRPG.h.

Referenced by setup().

7.49.4.13 unsigned long long int GenericRPG::max_time

Definition at line 80 of file backup/genericRPG.h.

7.49.4.14 unsigned int GenericRPG::min_delay

Definition at line 77 of file backup/genericRPG.h.

7.49.4.15 unsigned int GenericRPG::min_length

Definition at line 75 of file backup/genericRPG.h.

7.49.4.16 bool GenericRPG::only_sink [private]

Definition at line 56 of file backup/genericRPG.h.

Referenced by handle_out_pull_event(), and setup().

7.49.4.17 ofstream GenericRPG::out_file [private]

Definition at line 53 of file genericRPG.h.

7.49.4.18 string GenericRPG::out_filename [private]

Definition at line 51 of file backup/genericRPG.h.

7.49.4.19 deque< HighLevelPacket > GenericRPG::out_packets [private]

Definition at line 49 of file backup/genericRPG.h.

7.49.4.20 unsigned int GenericRPG::packets

Definition at line 67 of file backup/genericRPG.h.

Referenced by handle_out_pull_event(), and setup().

7.49.4.21 vector< bool > GenericRPG::ready [private]

Definition at line 54 of file backup/genericRPG.h.

Referenced by handle_out_pull_event(), handle_ready_event(), and setup().

7.49.4.22 unsigned int GenericRPG::seed

Definition at line 81 of file backup/genericRPG.h.

Referenced by GenericRPG(), handle_out_pull_event(), and init_generator().

7.49.4.23 bool GenericRPG::sending [private]

Definition at line 57 of file backup/genericRPG.h.

Referenced by GenericRPG(), handle_out_pull_event(), and handle_ready_event().

7.49.4.24 deque< HighLevelPacket > GenericRPG::sent_packets [private]

Definition at line 50 of file backup/genericRPG.h.

7.49.4.25 uint GenericRPG::vcs [private]

Definition at line 52 of file backup/genericRPG.h.

Referenced by set_no_vcs(), and setup().

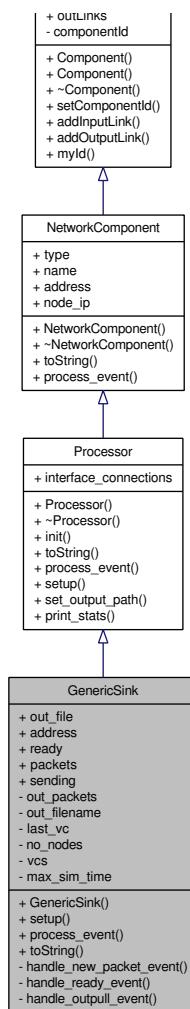
The documentation for this class was generated from the following files:

- [backup/genericRPG.h](#)
- [genericRPG.h](#)
- [backup/genericRPG.cc](#)
- [genericRPG.cc](#)

7.50 GenericSink Class Reference

```
#include <genericSink.h>
```

Inheritance diagram for GenericSink:



Collaboration diagram for GenericSink:

Public Member Functions

- **GenericSink ()**
- void **setup (uint v, uint time, uint no_nodes)**
- void **process_event (IrisEvent *e)**
- string **toString () const**

Public Attributes

- ofstream **out_file**
- uint **address**
- vector< bool > **ready**
- unsigned int **packets**
- bool **sending**

Private Member Functions

- void **handle_new_packet_event (IrisEvent *e)**
- void **handle_ready_event (IrisEvent *e)**
- void **handle_outpull_event (IrisEvent *e)**

Private Attributes

- deque< **HighLevelPacket * > out_packets**
- string **out_filename**
- uint **last_vc**
- uint **no_nodes**
- uint **vcs**
- uint **max_sim_time**

7.50.1 Detailed Description

Definition at line 37 of file genericSink.h.

7.50.2 Constructor & Destructor Documentation

7.50.2.1 GenericSink::GenericSink ()

Definition at line 33 of file genericSink.cc.

References NetworkComponent::name, and out_filename.

7.50.3 Member Function Documentation

7.50.3.1 void GenericSink::handle_new_packet_event (IrisEvent * e) [private]

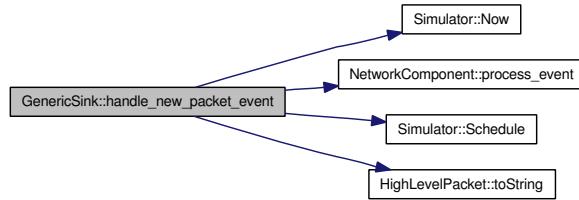
Definition at line 92 of file genericSink.cc.

References IrisEvent::event_data, Processor::interface_connections, Simulator::Now(), out_file, NetworkComponent::process_event(), READY_EVENT, Simulator::Schedule(),

HighLevelPacket::sent_time, HighLevelPacket::toString(), VirtualChannelDescription::vc, and HighLevelPacket::virtual_channel.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



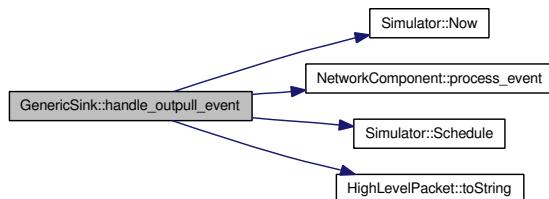
7.50.3.2 void GenericSink::handle_outpull_event (IrisEvent * e) [private]

Definition at line 115 of file genericSink.cc.

References _DBG, address, HighLevelPacket::destination, Processor::interface_connections, last_vc, NEW_PACKET_EVENT, no_nodes, NetworkComponent::node_ip, Simulator::Now(), out_file, packets, NetworkComponent::process_event(), ready, Simulator::Schedule(), sending, HighLevelPacket::sent_time, HighLevelPacket::source, HighLevelPacket::toString(), HighLevelPacket::transaction_id, and HighLevelPacket::virtual_channel.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



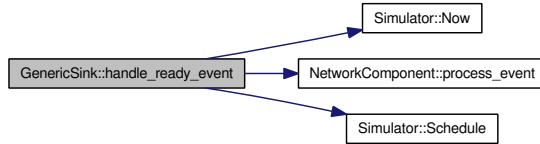
7.50.3.3 void GenericSink::handle_ready_event (IrisEvent * e) [private]

Definition at line 177 of file genericSink.cc.

References IrisEvent::event_data, Simulator::Now(), OUT_PULL_EVENT, NetworkComponent::process_event(), ready, Simulator::Schedule(), sending, IrisEvent::vc, and VirtualChannelDescription::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



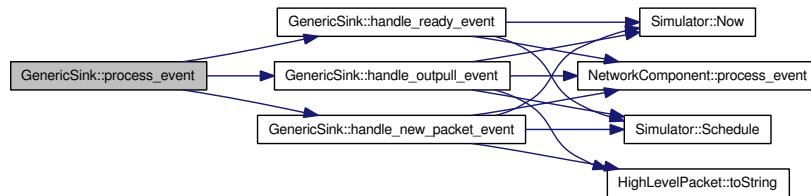
7.50.3.4 void GenericSink::process_event (IrisEvent * e) [virtual]

Implements **Processor** (p. 297).

Definition at line 70 of file genericSink.cc.

References handle_new_packet_event(), handle_outpull_event(), handle_ready_event(), NEW_PACKET_EVENT, OUT_PULL_EVENT, READY_EVENT, and IrisEvent::type.

Here is the call graph for this function:



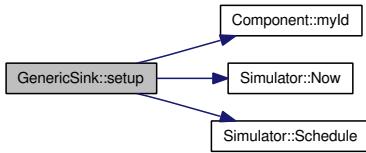
7.50.3.5 void GenericSink::setup (uint v, uint time, uint no_nodes) [virtual]

Implements **Processor** (p. 297).

Definition at line 40 of file genericSink.cc.

References address, IrisEvent::event_data, Processor::interface_connections, max_sim_time, Component::myId(), no_nodes, Simulator::Now(), out_file, out_filename, ready, READY_EVENT, Simulator::Schedule(), IrisEvent::type, VirtualChannelDescription::vc, and vcs.

Here is the call graph for this function:



7.50.3.6 string GenericSink::toString () const [virtual]

Reimplemented from **Processor** (p. 297).

Definition at line 199 of file genericSink.cc.

References address, and out_filename.

7.50.4 Member Data Documentation

7.50.4.1 uint GenericSink::address

Reimplemented from **NetworkComponent** (p. 275).

Definition at line 45 of file genericSink.h.

Referenced by handle_outpull_event(), setup(), and toString().

7.50.4.2 uint GenericSink::last_vc [private]

Definition at line 55 of file genericSink.h.

Referenced by handle_outpull_event().

7.50.4.3 uint GenericSink::max_sim_time [private]

Definition at line 58 of file genericSink.h.

Referenced by setup().

7.50.4.4 uint GenericSink::no_nodes [private]

Definition at line 56 of file genericSink.h.

Referenced by handle_outpull_event(), and setup().

7.50.4.5 ofstream GenericSink::out_file

Definition at line 44 of file genericSink.h.

Referenced by handle_new_packet_event(), handle_outpull_event(), and setup().

7.50.4.6 string GenericSink::out_filename [private]

Definition at line 54 of file genericSink.h.

Referenced by GenericSink(), setup(), and toString().

7.50.4.7 deque<HighLevelPacket*> GenericSink::out_packets [private]

Definition at line 53 of file genericSink.h.

7.50.4.8 unsigned int GenericSink::packets

Definition at line 47 of file genericSink.h.

Referenced by handle_outpull_event().

7.50.4.9 vector<bool> GenericSink::ready

Definition at line 46 of file genericSink.h.

Referenced by handle_outpull_event(), handle_ready_event(), and setup().

7.50.4.10 bool GenericSink::sending

Definition at line 48 of file genericSink.h.

Referenced by handle_outpull_event(), and handle_ready_event().

7.50.4.11 uint GenericSink::vcs [private]

Definition at line 57 of file genericSink.h.

Referenced by setup().

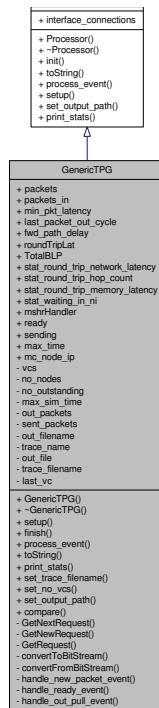
The documentation for this class was generated from the following files:

- **genericSink.h**
- **genericSink.cc**

7.51 GenericTPG Class Reference

```
#include <genericTPG.h>
```

Inheritance diagram for GenericTPG:



Collaboration diagram for GenericTPG:

Public Member Functions

- **GenericTPG ()**
- **~GenericTPG ()**
- **void setup (uint no_nodes, uint vcs, uint max_sim_time)**
- **void finish ()**
- **void process_event (IrisEvent *e)**
- **string toString () const**
- **string print_stats () const**
- **void set_trace_filename (string filename)**
- **void set_no_vcs (uint v)**
- **void set_output_path (string outpath)**
- **bool compare ()**

Public Attributes

- **unsigned int packets**
- **unsigned int packets_in**
- **double min_pkt_latency**
- **double last_packet_out_cycle**
- **unsigned long long int fwd_path_delay**
- **unsigned long long int roundTripLat**
- **unsigned long long int TotalBLP**
- **unsigned long long int stat_round_trip_network_latency**
- **unsigned long long int stat_round_trip_hop_count**
- **unsigned long long int stat_round_trip_memory_latency**
- **unsigned long long int stat_waiting_in_ni**
- **MSHR_H * mshrHandler**
- **vector< bool > ready**
- **bool sending**
- **unsigned long long int max_time**
- **vector< uint > mc_node_ip**

Private Member Functions

- **Request * GetNextRequest ()**
- **bool GetNewRequest (Request *req)**
- **Request * GetRequest ()**
- **void convertToBitStream (Request *req, HighLevelPacket *hlp)**
- **void convertFromBitStream (Request *req, HighLevelPacket *hlp)**
- **void handle_new_packet_event (IrisEvent *e)**
- **void handle_ready_event (IrisEvent *e)**
- **void handle_out_pull_event (IrisEvent *e)**

Private Attributes

- `uint vcs`
- `uint no_nodes`
- `uint no_outstanding`
- `unsigned long long int max_sim_time`
- `deque< HighLevelPacket > out_packets`
- `deque< HighLevelPacket > sent_packets`
- `string out_filename`
- `string trace_name`
- `ofstream out_file`
- `ifstream * trace_filename`
- `unsigned int last_vc`

7.51.1 Detailed Description

Definition at line 25 of file genericTPG.h.

7.51.2 Constructor & Destructor Documentation

7.51.2.1 GenericTPG::GenericTPG ()

Definition at line 12 of file genericTPG.cc.

References Processor::interface_connections, last_vc, NetworkComponent::name, out_filename, and sending.

7.51.2.2 GenericTPG::~GenericTPG ()

Definition at line 22 of file genericTPG.cc.

References MSHR_H::mshr, mshrHandler, out_file, and MSHR_H::trace_filename.

7.51.3 Member Function Documentation

7.51.3.1 bool GenericTPG::compare ()

7.51.3.2 void GenericTPG::convertFromBitStream (Request * req, HighLevelPacket * hlp) [private]

Definition at line 443 of file genericTPG.cc.

References Request::address, CACHE_BLOCK_SIZE, CACHE_PREFETCH, CACHE_READ, CACHE_WRITE, Request::cmdType, Request::data, HighLevelPacket::data, max_phy_link_bits, Request::mcNo, NETWORK_ADDRESS_BITS, Data::size, HighLevelPacket::source, and Data::value.

Referenced by handle_new_packet_event().

Here is the caller graph for this function:



7.51.3.3 void GenericTPG::convertToBitStream (Request * *req*, HighLevelPacket * *hlp*) [private]

Definition at line 406 of file genericTPG.cc.

References Request::address, Request::arrivalTime, CACHE_BLOCK_SIZE, CACHE_WRITEBACK, Request::cmdType, HighLevelPacket::data, HighLevelPacket::data_payload_length, max_phy_link_bits, HighLevelPacket::msg_class, NETWORK_ADDRESS_BITS, NETWORK_COMMAND_BITS, NETWORK_THREADID_BITS, ONE_FLIT_REQ, RESPONSE_PKT, HighLevelPacket::sent_time, and Request::threadId.

Referenced by handle_out_pull_event().

Here is the caller graph for this function:



7.51.3.4 void GenericTPG::finish ()

Definition at line 150 of file genericTPG.cc.

References out_file, and trace_filename.

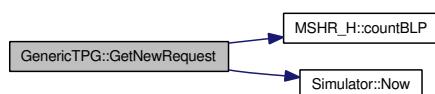
7.51.3.5 bool GenericTPG::GetNewRequest (Request * *req*) [private]

Definition at line 362 of file genericTPG.cc.

References MSHR_H::countBLP(), MSHR_H::lastScheduledIndex, MSHR_H::mshr, mshrHandler, Simulator::Now(), TotalBLP, and MSHR_H::writeQueue.

Referenced by GetRequest().

Here is the call graph for this function:



Here is the caller graph for this function:



7.51.3.6 Request * GenericTPG::GetNextRequest () [private]

Definition at line 324 of file genericTPG.cc.

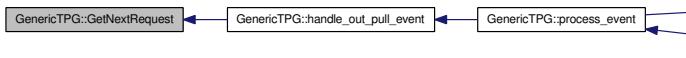
References Request::address, Request::arrivalTime, Request::cmdType, MSHR_H::GlobalAddrMap(), mshrHandler, NetworkComponent::node_ip, Request::threadId, and trace_filename.

Referenced by handle_out_pull_event().

Here is the call graph for this function:



Here is the caller graph for this function:



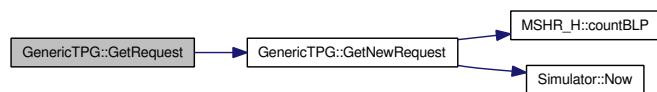
7.51.3.7 Request * GenericTPG::GetRequest () [private]

Definition at line 394 of file genericTPG.cc.

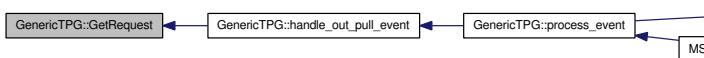
References GetNewRequest().

Referenced by handle_out_pull_event().

Here is the call graph for this function:



Here is the caller graph for this function:



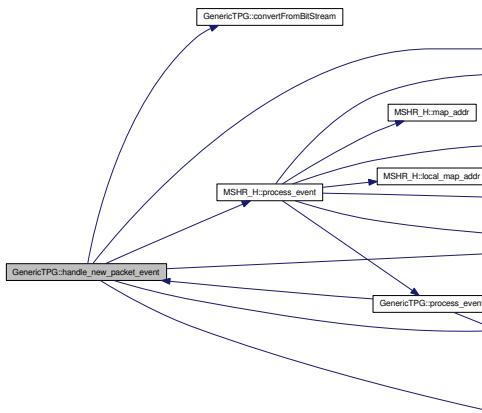
7.51.3.8 void GenericTPG::handle_new_packet_event (IrisEvent * e) [private]

Definition at line 179 of file genericTPG.cc.

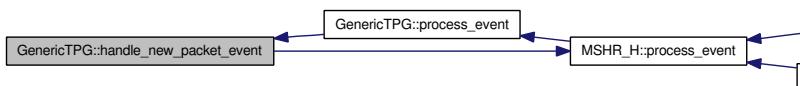
References _DBG, Request::address, HighLevelPacket::avg_network_latency, convertFromBitStream(), IrisEvent::event_data, HighLevelPacket::hop_count, Processor::interface_connections, last_packet_out_cycle, min_pkt_latency, MSHR_DELETE, mshrHandler, Simulator::Now(), out_file, out_filename, packets_in, NetworkComponent::process_event(), MSHR_H::process_event(), READY_EVENT, HighLevelPacket::req_start_time, Simulator::Schedule(), HighLevelPacket::sent_time, Request::startTime, HighLevelPacket::stat_memory_serviced_time, stat_round_trip_hop_count, stat_round_trip_memory_latency, stat_round_trip_network_latency, stat_waiting_in_ni, HighLevelPacket::toString(), IrisEvent::type, IrisEvent::vc, and HighLevelPacket::waiting_in_ni.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.51.3.9 void GenericTPG::handle_out_pull_event (IrisEvent * e) [private]

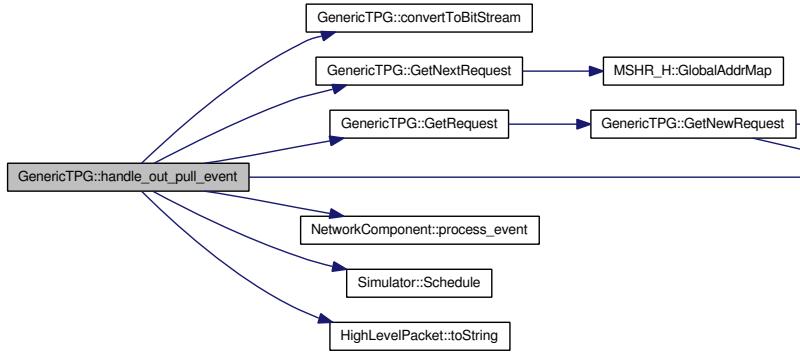
Definition at line 239 of file genericTPG.cc.

References _DBG, HighLevelPacket::addr, Request::address, NetworkComponent::address, Request::arrivalTime, convertToBitStream(), HighLevelPacket::destination, fwd_path_delay, GetNextRequest(), GetRequest(), Processor::interface_connections, MSHR_H::lastScheduledIndex, Request::mcNo, MSHR_H::mshr, mshrHandler, NEW_PACKET_EVENT, no_nodes, NetworkComponent::node_ip, Simulator::Now(), OUT_PULL_EVENT, packets, NetworkComponent::process_event(), ready, HighLevelPacket::req_start_time, Simulator::Schedule(), sending, HighLevelPacket::sent_time, HighLevelPacket::source, Request::startTime, HighLevelPacket::toString(), HighLevelPacket::transaction_id, IrisEvent::type,

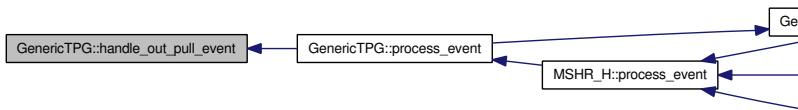
IrisEvent::vc, and HighLevelPacket::virtual_channel.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



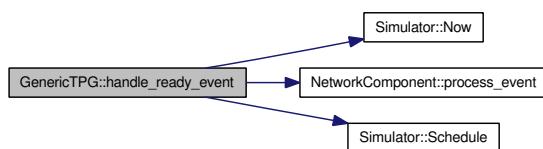
7.51.3.10 void GenericTPG::handle_ready_event (IrisEvent * e) [private]

Definition at line 306 of file genericTPG.cc.

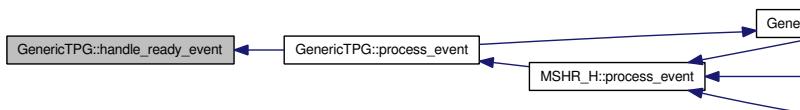
References `max_sim_time`, `Simulator::Now()`, `OUT_PULL_EVENT`, `NetworkComponent::process_event()`, `ready`, `Simulator::Schedule()`, and `sending`.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.51.3.11 string GenericTPG::print_stats () const [virtual]

Implements **Processor** (p. 296).

Definition at line 486 of file genericTPG.cc.

References NetworkComponent::address, fwd_path_delay, last_packet_out_cycle, min_pkt_latency, mshrHandler, NetworkComponent::node_ip, packets, packets_in, roundTripLat, stat_round_trip_hop_count, stat_round_trip_memory_latency, stat_round_trip_network_latency, stat_waiting_in_ni, TotalBLP, and MSHR_H::unsink.

7.51.3.12 void GenericTPG::process_event (IrisEvent * e) [virtual]

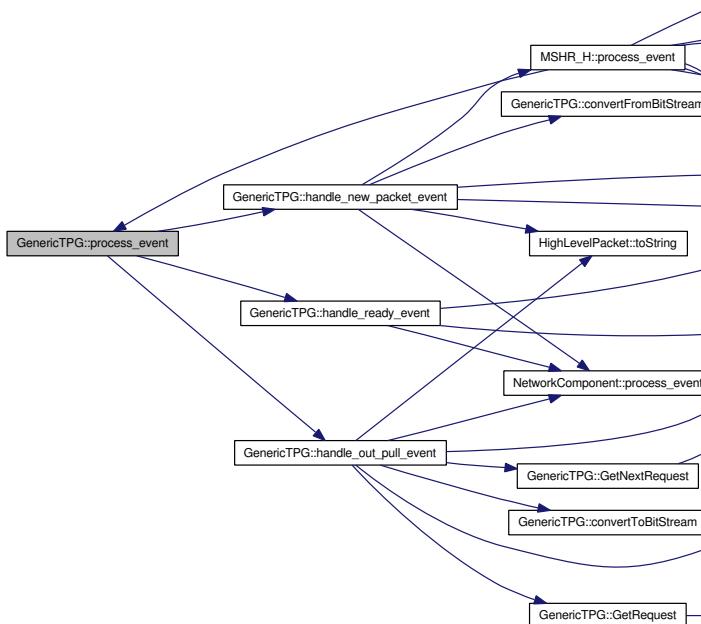
Implements **Processor** (p. 297).

Definition at line 158 of file genericTPG.cc.

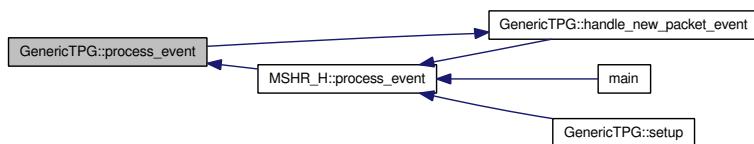
References NetworkComponent::address, handle_new_packet_event(), handle_out_pull_event(), handle_ready_event(), NEW_PACKET_EVENT, OUT_PULL_EVENT, READY_EVENT, and IrisEvent::type.

Referenced by MSHR_H::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.51.3.13 void GenericTPG::set_no_vcs (uint *v*)

Definition at line 39 of file genericTPG.cc.

References vcs.

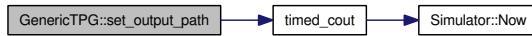
7.51.3.14 void GenericTPG::set_output_path (string *outpath*) [virtual]

Implements **Processor** (p. 297).

Definition at line 134 of file genericTPG.cc.

References NetworkComponent::node_ip, out_file, out_filename, and timed_cout().

Here is the call graph for this function:



7.51.3.15 void GenericTPG::set_trace_filename (string *filename*)

Definition at line 32 of file genericTPG.cc.

References trace_name.

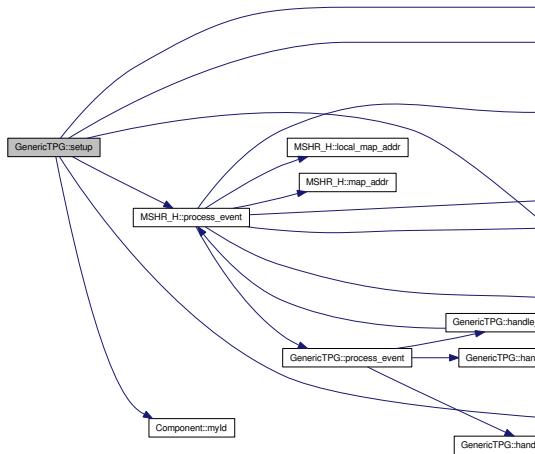
7.51.3.16 void GenericTPG::setup (uint *no_nodes*, uint *vcs*, uint *max_sim_time*) [virtual]

Implements **Processor** (p. 297).

Definition at line 45 of file genericTPG.cc.

References Request::address, NetworkComponent::address, Request::arrivalTime, Request::cmdType, MSHR_H::filename, fwd_path_delay, MSHR_H::GlobalAddrMap(), MSHR_H::id, last_packet_out_cycle, max_sim_time, min_pkt_latency, mshrHandler, Component::myId(), no_nodes, NetworkComponent::node_ip, Simulator::Now(), packets, packets_in, MSHR_H::parent, NetworkComponent::process_event(), MSHR_H::process_event(), ready, READY_EVENT, roundTripLat, Simulator::Schedule(), stat_round_trip_hop_count, stat_round_trip_memory_latency, stat_round_trip_network_latency, stat_waiting_in_ni, Request::threadId, TotalBLP, MSHR_H::trace_filename, trace_filename, trace_name, and vcs.

Here is the call graph for this function:



7.51.3.17 string GenericTPG::toString () const [virtual]

Reimplemented from **Processor** (p. 297).

Definition at line 473 of file genericTPG.cc.

References NetworkComponent::address, NetworkComponent::node_ip, ready, and trace_name.

7.51.4 Member Data Documentation

7.51.4.1 unsigned long long int GenericTPG::fwd_path_delay

Definition at line 57 of file genericTPG.h.

Referenced by handle_out_pull_event(), print_stats(), and setup().

7.51.4.2 double GenericTPG::last_packet_out_cycle

Definition at line 56 of file genericTPG.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.51.4.3 unsigned int GenericTPG::last_vc [private]

Definition at line 39 of file genericTPG.h.

Referenced by GenericTPG().

7.51.4.4 unsigned long long int GenericTPG::max_sim_time [private]

Definition at line 32 of file genericTPG.h.

Referenced by handle_ready_event(), and setup().

7.51.4.5 unsigned long long int GenericTPG::max_time

Definition at line 68 of file genericTPG.h.

7.51.4.6 vector<uint> GenericTPG::mc_node_ip

Definition at line 78 of file genericTPG.h.

7.51.4.7 double GenericTPG::min_pkt_latency

Definition at line 55 of file genericTPG.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.51.4.8 MSHR_H* GenericTPG::mshrHandler

Definition at line 65 of file genericTPG.h.

Referenced by GetNewRequest(), GetNextRequest(), handle_new_packet_event(), handle_out_pull_event(), print_stats(), setup(), and ~GenericTPG().

7.51.4.9 uint GenericTPG::no_nodes [private]

Definition at line 30 of file genericTPG.h.

Referenced by handle_out_pull_event(), and setup().

7.51.4.10 uint GenericTPG::no_outstanding [private]

Definition at line 31 of file genericTPG.h.

7.51.4.11 ofstream GenericTPG::out_file [private]

Definition at line 37 of file genericTPG.h.

Referenced by finish(), handle_new_packet_event(), set_output_path(), and ~GenericTPG().

7.51.4.12 string GenericTPG::out_filename [private]

Definition at line 35 of file genericTPG.h.

Referenced by GenericTPG(), handle_new_packet_event(), and set_output_path().

7.51.4.13 deque< HighLevelPacket > GenericTPG::out_packets [private]

Definition at line 33 of file genericTPG.h.

7.51.4.14 unsigned int GenericTPG::packets

Definition at line 53 of file genericTPG.h.

Referenced by handle_out_pull_event(), print_stats(), and setup().

7.51.4.15 unsigned int GenericTPG::packets_in

Definition at line 54 of file genericTPG.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.51.4.16 vector< bool > GenericTPG::ready

Definition at line 66 of file genericTPG.h.

Referenced by handle_out_pull_event(), handle_ready_event(), setup(), and toString().

7.51.4.17 unsigned long long int GenericTPG::roundTripLat

Definition at line 58 of file genericTPG.h.

Referenced by print_stats(), and setup().

7.51.4.18 bool GenericTPG::sending

Definition at line 67 of file genericTPG.h.

Referenced by GenericTPG(), handle_out_pull_event(), and handle_ready_event().

7.51.4.19 deque< HighLevelPacket > GenericTPG::sent_packets [private]

Definition at line 34 of file genericTPG.h.

7.51.4.20 unsigned long long int GenericTPG::stat_round_trip_hop_count

Definition at line 61 of file genericTPG.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.51.4.21 unsigned long long int GenericTPG::stat_round_trip_memory_latency

Definition at line 62 of file genericTPG.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.51.4.22 unsigned long long int GenericTPG::stat_round_trip_network_latency

Definition at line 60 of file genericTPG.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.51.4.23 unsigned long long int GenericTPG::stat_waiting_in_ni

Definition at line 63 of file genericTPG.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.51.4.24 unsigned long long int GenericTPG::TotalBLP

Definition at line 59 of file genericTPG.h.

Referenced by GetNewRequest(), print_stats(), and setup().

7.51.4.25 ifstream* GenericTPG::trace_filename [private]

Definition at line 38 of file genericTPG.h.

Referenced by finish(), GetNextRequest(), and setup().

7.51.4.26 string GenericTPG::trace_name [private]

Definition at line 36 of file genericTPG.h.

Referenced by set_trace_filename(), setup(), and toString().

7.51.4.27 uint GenericTPG::vcs [private]

Definition at line 29 of file genericTPG.h.

Referenced by set_no_vcs(), and setup().

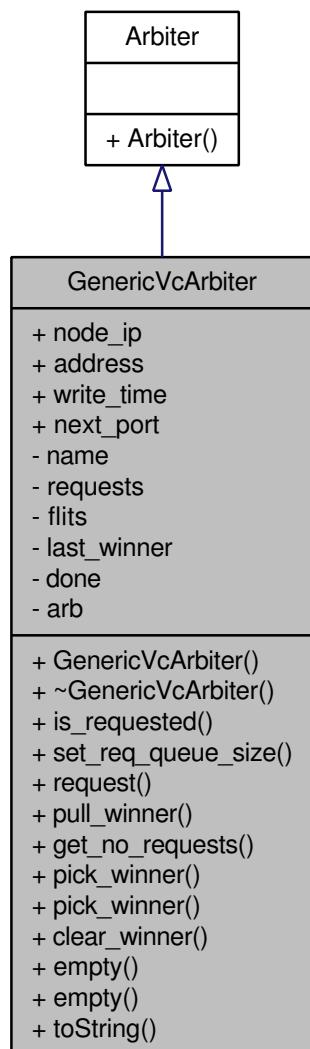
The documentation for this class was generated from the following files:

- **genericTPG.h**
- **genericTPG.cc**
- **genericTPG_temp.cc**

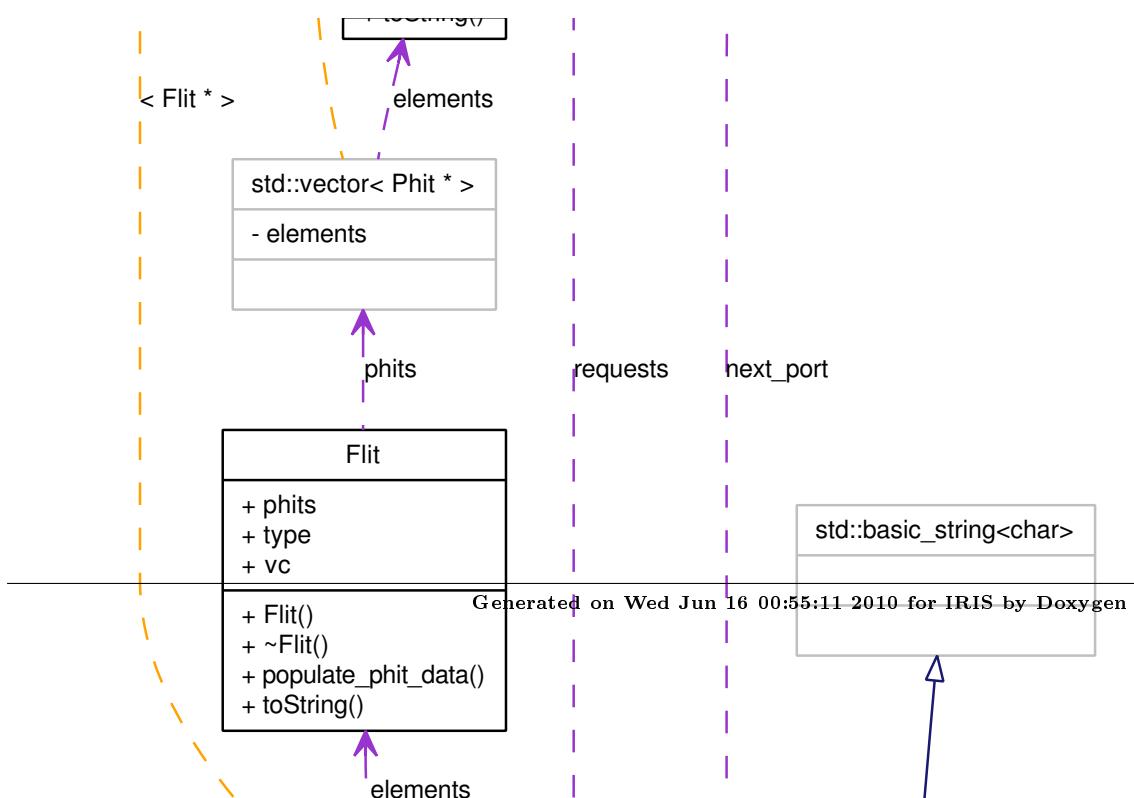
7.52 GenericVcArbiter Class Reference

```
#include <genericVcArbiter.h>
```

Inheritance diagram for GenericVcArbiter:



Collaboration diagram for GenericVcArbiter:



Public Member Functions

- **GenericVcArbiter ()**
- **~GenericVcArbiter ()**
- **bool is_requested (uint ch)**
- **void set_req_queue_size (uint size)**
- **void request (Flit *f, uint index)**
- **Flit * pull_winner ()**
- **uint get_no_requests ()**
- **uint pick_winner ()**
- **uint pick_winner (vector< bool > ready)**
- **void clear_winner ()**
- **bool empty ()**
- **bool empty (vector< bool > ready)**
- **string toString () const**

Public Attributes

- **uint node_ip**
- **uint address**
- **unsigned long long int write_time**
- **vector< uint > next_port**

Private Attributes

- **string name**
- **vector< bool > requests**
- **vector< Flit * > flits**
- **uint last_winner**
- **bool done**
- **bool arb**

7.52.1 Detailed Description

Definition at line 33 of file genericVcArbiter.h.

7.52.2 Constructor & Destructor Documentation

7.52.2.1 GenericVcArbiter::GenericVcArbiter ()

Definition at line 24 of file genericVcArbiter.cc.

References address, done, last_winner, name, and node_ip.

7.52.2.2 GenericVcArbiter::~GenericVcArbiter ()

Definition at line 33 of file genericVcArbiter.cc.

7.52.3 Member Function Documentation

7.52.3.1 void GenericVcArbiter::clear_winner ()

Definition at line 164 of file genericVcArbiter.cc.

References last_winner, and requests.

7.52.3.2 bool GenericVcArbiter::empty (vector< bool > ready)

Definition at line 185 of file genericVcArbiter.cc.

References _DBG, and requests.

7.52.3.3 bool GenericVcArbiter::empty ()

Definition at line 175 of file genericVcArbiter.cc.

References requests.

7.52.3.4 uint GenericVcArbiter::get_no_requests ()

Definition at line 79 of file genericVcArbiter.cc.

References requests.

7.52.3.5 bool GenericVcArbiter::is_requested (uint ch)

Definition at line 65 of file genericVcArbiter.cc.

References _DBG, and requests.

7.52.3.6 uint GenericVcArbiter::pick_winner (vector< bool > ready)

Definition at line 128 of file genericVcArbiter.cc.

References _DBG, done, last_winner, and requests.

7.52.3.7 uint GenericVcArbiter::pick_winner ()

Definition at line 97 of file genericVcArbiter.cc.

References _DBG, done, last_winner, and requests.

Referenced by pull_winner().

Here is the caller graph for this function:



7.52.3.8 Flit * GenericVcArbiter::pull_winner ()

Definition at line 47 of file genericVcArbiter.cc.

References _DBG, done, flits, last_winner, pick_winner(), and requests.

Here is the call graph for this function:

**7.52.3.9 void GenericVcArbiter::request (Flit * f, uint index)**

Definition at line 39 of file genericVcArbiter.cc.

References flits, and requests.

7.52.3.10 void GenericVcArbiter::set_req_queue_size (uint size)

Definition at line 85 of file genericVcArbiter.cc.

References flits, next_port, and requests.

7.52.3.11 string GenericVcArbiter::toString () const

Definition at line 202 of file genericVcArbiter.cc.

References last_winner, and requests.

7.52.4 Member Data Documentation

7.52.4.1 uint GenericVcArbiter::address

Definition at line 37 of file genericVcArbiter.h.

Referenced by GenericVcArbiter().

7.52.4.2 bool GenericVcArbiter::arb [private]

Definition at line 65 of file genericVcArbiter.h.

7.52.4.3 bool GenericVcArbiter::done [private]

Definition at line 64 of file genericVcArbiter.h.

Referenced by GenericVcArbiter(), pick_winner(), and pull_winner().

7.52.4.4 vector<Flit*> GenericVcArbiter::flits [private]

Definition at line 62 of file genericVcArbiter.h.

Referenced by pull_winner(), request(), and set_req_queue_size().

7.52.4.5 uint GenericVcArbiter::last_winner [private]

Definition at line 63 of file genericVcArbiter.h.

Referenced by clear_winner(), GenericVcArbiter(), pick_winner(), pull_winner(), and toString().

7.52.4.6 string GenericVcArbiter::name [private]

Definition at line 60 of file genericVcArbiter.h.

Referenced by GenericVcArbiter().

7.52.4.7 vector<uint> GenericVcArbiter::next_port

Definition at line 55 of file genericVcArbiter.h.

Referenced by set_req_queue_size().

7.52.4.8 uint GenericVcArbiter::node_ip

Definition at line 36 of file genericVcArbiter.h.

Referenced by GenericVcArbiter().

7.52.4.9 vector<bool> GenericVcArbiter::requests [private]

Definition at line 61 of file genericVcArbiter.h.

Referenced by clear_winner(), empty(), get_no_requests(), is_requested(), pick_winner(), pull_winner(), request(), set_req_queue_size(), and toString().

7.52.4.10 unsigned long long int GenericVcArbiter::write_time

Definition at line 50 of file genericVcArbiter.h.

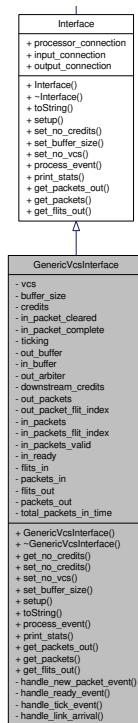
The documentation for this class was generated from the following files:

- **genericVcArbiter.h**
- **genericVcArbiter.cc**

7.53 GenericVcsInterface Class Reference

```
#include <genericVcsInterface.h>
```

Inheritance diagram for GenericVcsInterface:



Collaboration diagram for GenericVcsInterface:



Public Member Functions

- `GenericVcsInterface ()`
- `~GenericVcsInterface ()`
- `uint get_no_credits () const`
- `void set_no_credits (int credits)`
- `void set_no_vcs (uint v)`
- `void set_buffer_size (uint b)`
- `void setup (uint v, uint cr)`
- `string toString () const`
- `void process_event (IrisEvent *e)`
- `string print_stats ()`
- `ullint get_packets_out ()`
- `ullint get_packets ()`
- `ullint get_flits_out ()`

Private Member Functions

- `void handle_new_packet_event (IrisEvent *e)`
- `void handle_ready_event (IrisEvent *e)`
- `void handle_tick_event (IrisEvent *e)`
- `void handle_link_arrival (IrisEvent *e)`

Private Attributes

- `uint vcs`
- `uint buffer_size`
- `int credits`
- `bool in_packet_cleared`
- `vector< bool > in_packet_complete`
- `bool ticking`
- `GenericBuffer out_buffer`
- `GenericBuffer in_buffer`
- `GenericArbiter out_arbiter`
- `vector< int > downstream_credits`
- `vector< LowLevelPacket > out_packets`
- `vector< uint > out_packet_flit_index`
- `vector< LowLevelPacket > in_packets`
- `vector< uint > in_packets_flit_index`
- `vector< bool > in_packets_valid`
- `vector< bool > in_ready`
- `ullint flits_in`
- `ullint packets_in`
- `ullint flits_out`
- `ullint packets_out`
- `ullint total_packets_in_time`

7.53.1 Detailed Description

Definition at line 45 of file genericVcsInterface.h.

7.53.2 Constructor & Destructor Documentation

7.53.2.1 GenericVcsInterface::GenericVcsInterface ()

Definition at line 26 of file genericVcsInterface.cc.

References NetworkComponent::name, and ticking.

7.53.2.2 GenericVcsInterface::~GenericVcsInterface ()

Definition at line 32 of file genericVcsInterface.cc.

References in_packets, and out_packets.

7.53.3 Member Function Documentation

7.53.3.1 ullint GenericVcsInterface::get_flits_out () [virtual]

Implements **Interface** (p. 231).

Definition at line 505 of file genericVcsInterface.cc.

References flits_out.

7.53.3.2 uint GenericVcsInterface::get_no_credits () const

Definition at line 119 of file genericVcsInterface.cc.

References credits.

7.53.3.3 ullint GenericVcsInterface::get_packets () [virtual]

Implements **Interface** (p. 231).

Definition at line 517 of file genericVcsInterface.cc.

References packets_in, and packets_out.

7.53.3.4 ullint GenericVcsInterface::get_packets_out () [virtual]

Implements **Interface** (p. 231).

Definition at line 511 of file genericVcsInterface.cc.

References packets_out.

7.53.3.5 void GenericVcsInterface::handle_link_arrival (IrisEvent * e) [private]

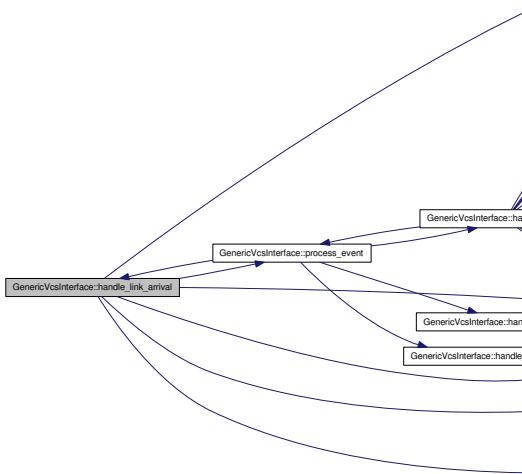
link arrival can be for a flit or credit. FLIT: Add incoming flit to buffer and send a credit back. Credit for the tail flit is not sent here. This is held till we know the ejection port is free. (This is modelled here for the additional **MC** (p. 247) backpressure modelling and need not be this way). CREDIT: Update state for downstream credits. Generate a TICK event at the end of it. This will restart the interface if it is blocked on a credit(outgoing path) or initiate a new packet(incoming path).

Definition at line 200 of file genericVcsInterface.cc.

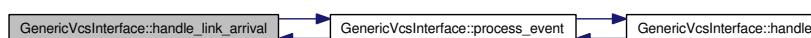
References NetworkComponent::address, GenericBuffer::change_push_channel(), CREDIT_ID, do_two_stage_router, downstream_credits, IrisEvent::event_data, FLIT_ID, flits_in, HEAD, in_buffer, Interface::input_connection, LINK_ARRIVAL_EVENT, Simulator::Now(), ONE_FLIT_REQ, packets_in, process_event(), NetworkComponent::process_event(), LinkArrivalData::ptr, GenericBuffer::push(), Simulator::Schedule(), TAIL, TICK_EVENT, ticking, total_packets_in_time, IrisEvent::type, Flit::type, LinkArrivalData::type, LinkArrivalData::valid, IrisEvent::vc, and LinkArrivalData::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



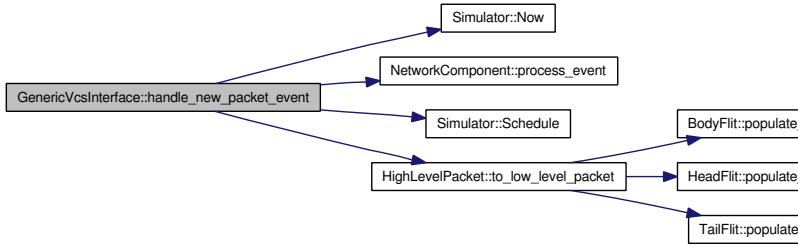
7.53.3.6 void GenericVcsInterface::handle_new_packet_event (IrisEvent * e)
[private]

Definition at line 279 of file genericVcsInterface.cc.

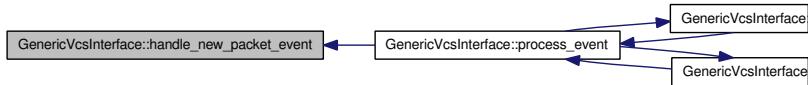
References `_DBG`, `IrisEvent::event_data`, `Simulator::Now()`, `out_packet_flit_index`, `out_packets`, `NetworkComponent::process_event()`, `Simulator::Schedule()`, `TICK_EVENT`, `ticking`, `HighLevelPacket::to_low_level_packet()`, `IrisEvent::vc`, and `HighLevelPacket::virtual_channel`.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.53.3.7 void GenericVcsInterface::handle_ready_event (IrisEvent * e) [private]

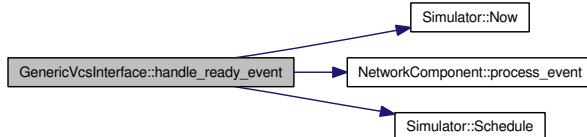
These are credits from the processor and hence do not update flow control state. Flow control state is updated at the link arrival event for a credit.

Definition at line 175 of file genericVcsInterface.cc.

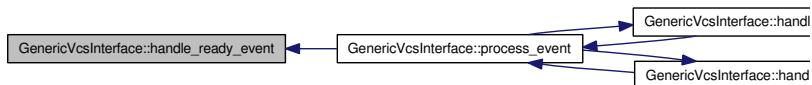
References _DBG, in_ready, Simulator::Now(), NetworkComponent::process_event(), Simulator::Schedule(), TICK_EVENT, ticking, and IrisEvent::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.53.3.8 void GenericVcsInterface::handle_tick_event (IrisEvent * e) [private]

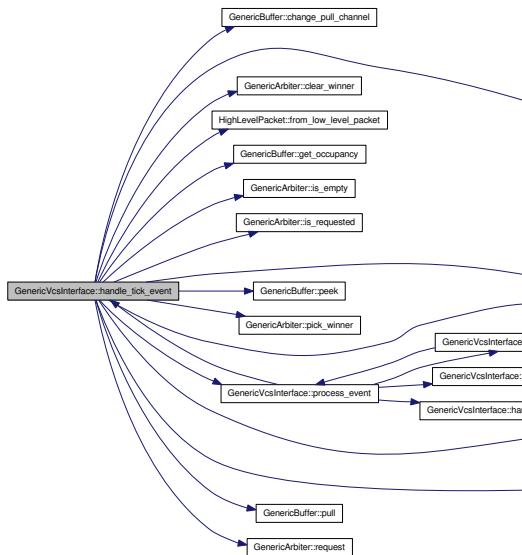
Definition at line 308 of file genericVcsInterface.cc.

References _DBG, NetworkComponent::address, HighLevelPacket::avg_network_latency, GenericBuffer::change_pull_channel(), GenericBuffer::change_push_channel(),

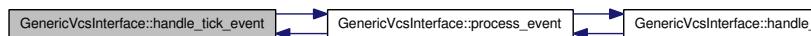
GenericArbiter::clear_winner(), CREDIT_ID, credits, do_two_stage_router, downstream_credits, IrisEvent::event_data, FLIT_ID, flits_out, HighLevelPacket::from_low_level_packet(), GenericBuffer::get_occupancy(), HEAD, HighLevelPacket::hop_count, in_buffer, in_packet_complete, in_packets, in_packets_flist_index, in_ready, Interface::input_connection, GenericArbiter::is_empty(), GenericArbiter::is_requested(), LINK_ARRIVAL_EVENT, NEW_PACKET_EVENT, NetworkComponent::node_ip, Simulator::Now(), ONE_FLIT_REQ, out_arbiter, out_buffer, out_packet_flist_index, out_packets, Interface::output_connection, packets_out, GenericBuffer::peek(), GenericArbiter::pick_winner(), process_event(), NetworkComponent::process_event(), Interface::processor_connection, LinkArrivalData::ptr, GenericBuffer::pull(), GenericBuffer::push(), READY_EVENT, HighLevelPacket::recv_time, HighLevelPacket::req_start_time, GenericArbiter::request(), Simulator::Schedule(), IrisEvent::src_id, TAIL, TICK_EVENT, ticking, IrisEvent::type, Flit::type, LinkArrivalData::type, IrisEvent::vc, LinkArrivalData::vc, vcs, and HighLevelPacket::waiting_in_ni.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.53.3.9 string GenericVcsInterface::print_stats () [virtual]

Implements **Interface** (p. 231).

Definition at line 489 of file genericVcsInterface.cc.

References flits_in, flits_out, NetworkComponent::node_ip, packets_in, packets_out, and total_packets_in_time.

7.53.3.10 void GenericVcsInterface::process_event (IrisEvent * e) [virtual]

READY_EVENT: A ready is the credit at a packet level with the injection/ejection point

LINK_ARRIVAL_EVENT: This is a flit/credit arrival from the network. (Incoming path)

NEW_PACKET_EVENT: This is a high level packet from the processor connected to this node

TICK_EVENT: Update internal state of the interface. This involves two distinct paths (Incoming and Outgoing). Push flits into the output buffer and send them out after all flits for a packet are moved to the output buffer(Outgoing path). Push flits from the input buffer to a low level packet. Convert the low level packet to a high level packet when complete and then send it to the injection processor connected to this node (Incoming path).

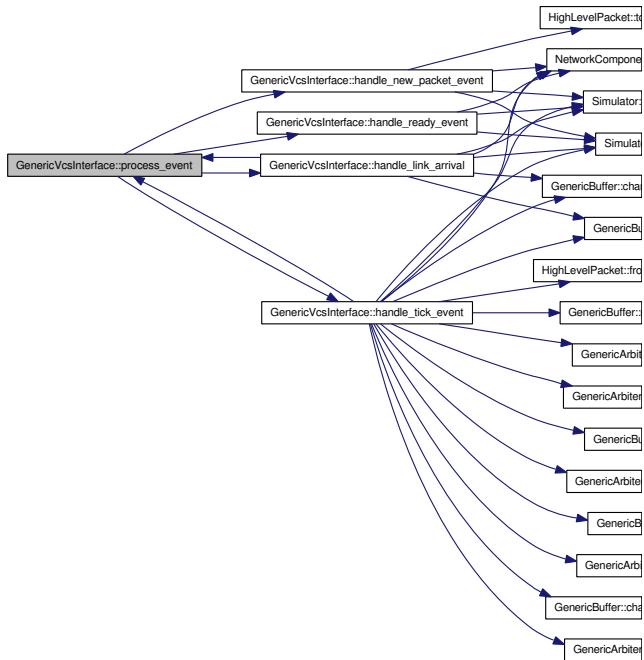
Implements **Interface** (p. 231).

Definition at line 135 of file genericVcsInterface.cc.

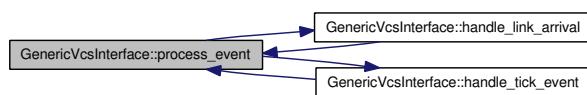
References handle_link_arrival(), handle_new_packet_event(), handle_ready_event(), handle_tick_event(), LINK_ARRIVAL_EVENT, NEW_PACKET_EVENT, READY_EVENT, TICK_EVENT, and IrisEvent::type.

Referenced by handle_link_arrival(), and handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.53.3.11 void GenericVcsInterface::set_buffer_size (uint b) [virtual]

Implements **Interface** (p. 231).

Definition at line 130 of file genericVcsInterface.cc.

7.53.3.12 void GenericVcsInterface::set_no_credits (int credits) [virtual]

Implements **Interface** (p. 231).

Definition at line 114 of file genericVcsInterface.cc.

7.53.3.13 void GenericVcsInterface::set_no_vcs (uint v) [virtual]

Implements **Interface** (p. 231).

Definition at line 125 of file genericVcsInterface.cc.

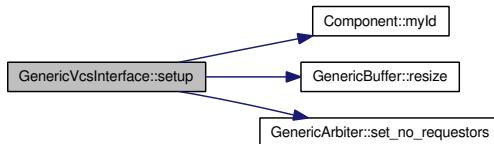
7.53.3.14 void GenericVcsInterface::setup (uint v, uint cr) [virtual]

Implements **Interface** (p. 231).

Definition at line 39 of file genericVcsInterface.cc.

References NetworkComponent::address, buffer_size, credits, downstream_credits, flits_in, flits_out, in_buffer, in_packet_complete, in_packets, in_packets_flit_index, in_packets_valid, in_ready, Component::myId(), out_arbiter, out_buffer, out_packet_flit_index, out_packets, packets_in, packets_out, GenericBuffer::resize(), GenericArbiter::set_no_requestors(), ticking, total_packets_in_time, and vcs.

Here is the call graph for this function:

**7.53.3.15 string GenericVcsInterface::toString () const [virtual]**

Reimplemented from **Interface** (p. 231).

Definition at line 101 of file genericVcsInterface.cc.

References NetworkComponent::address, in_buffer, NetworkComponent::node_ip, out_buffer, out_packets, and GenericBuffer::toString().

Here is the call graph for this function:



7.53.4 Member Data Documentation

7.53.4.1 `uint GenericVcsInterface::buffer_size [private]`

Definition at line 68 of file genericVcsInterface.h.

Referenced by `setup()`.

7.53.4.2 `int GenericVcsInterface::credits [private]`

Definition at line 69 of file genericVcsInterface.h.

Referenced by `get_no_credits()`, `handle_tick_event()`, and `setup()`.

7.53.4.3 `vector< int > GenericVcsInterface::downstream_credits [private]`

Definition at line 77 of file genericVcsInterface.h.

Referenced by `handle_link_arrival()`, `handle_tick_event()`, and `setup()`.

7.53.4.4 `ullint GenericVcsInterface::flits_in [private]`

Definition at line 97 of file genericVcsInterface.h.

Referenced by `handle_link_arrival()`, `print_stats()`, and `setup()`.

7.53.4.5 `ullint GenericVcsInterface::flits_out [private]`

Definition at line 99 of file genericVcsInterface.h.

Referenced by `get_flits_out()`, `handle_tick_event()`, `print_stats()`, and `setup()`.

7.53.4.6 `GenericBuffer GenericVcsInterface::in_buffer [private]`

Definition at line 75 of file genericVcsInterface.h.

Referenced by `handle_link_arrival()`, `handle_tick_event()`, `setup()`, and `toString()`.

7.53.4.7 `bool GenericVcsInterface::in_packet_cleared [private]`

Definition at line 70 of file genericVcsInterface.h.

7.53.4.8 `vector<bool> GenericVcsInterface::in_packet_complete [private]`

Definition at line 71 of file genericVcsInterface.h.

Referenced by `handle_tick_event()`, and `setup()`.

7.53.4.9 `vector< LowLevelPacket> GenericVcsInterface::in_packets [private]`

Definition at line 84 of file genericVcsInterface.h.

Referenced by `handle_tick_event()`, `setup()`, and `~GenericVcsInterface()`.

7.53.4.10 vector< uint > GenericVcsInterface::in_packets_flit_index [private]

Definition at line 85 of file genericVcsInterface.h.

Referenced by handle_tick_event(), and setup().

7.53.4.11 vector< bool> GenericVcsInterface::in_packets_valid [private]

Definition at line 86 of file genericVcsInterface.h.

Referenced by setup().

7.53.4.12 vector< bool > GenericVcsInterface::in_ready [private]

Definition at line 88 of file genericVcsInterface.h.

Referenced by handle_ready_event(), handle_tick_event(), and setup().

7.53.4.13 GenericArbiter GenericVcsInterface::out_arbiter [private]

Definition at line 76 of file genericVcsInterface.h.

Referenced by handle_tick_event(), and setup().

7.53.4.14 GenericBuffer GenericVcsInterface::out_buffer [private]

Definition at line 74 of file genericVcsInterface.h.

Referenced by handle_tick_event(), setup(), and toString().

7.53.4.15 vector< uint > GenericVcsInterface::out_packet_flit_index [private]

Definition at line 81 of file genericVcsInterface.h.

Referenced by handle_new_packet_event(), handle_tick_event(), and setup().

7.53.4.16 vector< LowLevelPacket> GenericVcsInterface::out_packets [private]

Definition at line 80 of file genericVcsInterface.h.

Referenced by handle_new_packet_event(), handle_tick_event(), setup(), toString(), and ~GenericVcsInterface().

7.53.4.17 ullint GenericVcsInterface::packets_in [private]

Definition at line 98 of file genericVcsInterface.h.

Referenced by get_packets(), handle_link_arrival(), print_stats(), and setup().

7.53.4.18 ullint GenericVcsInterface::packets_out [private]

Definition at line 100 of file genericVcsInterface.h.

Referenced by `get_packets()`, `get_packets_out()`, `handle_tick_event()`, `print_stats()`, and `setup()`.

7.53.4.19 bool GenericVcsInterface::ticking [private]

Definition at line 73 of file genericVcsInterface.h.

Referenced by `GenericVcsInterface()`, `handle_link_arrival()`, `handle_new_packet_event()`, `handle_ready_event()`, `handle_tick_event()`, and `setup()`.

7.53.4.20 ullint GenericVcsInterface::total_packets_in_time [private]

Definition at line 101 of file genericVcsInterface.h.

Referenced by `handle_link_arrival()`, `print_stats()`, and `setup()`.

7.53.4.21 uint GenericVcsInterface::vcs [private]

Definition at line 67 of file genericVcsInterface.h.

Referenced by `handle_tick_event()`, and `setup()`.

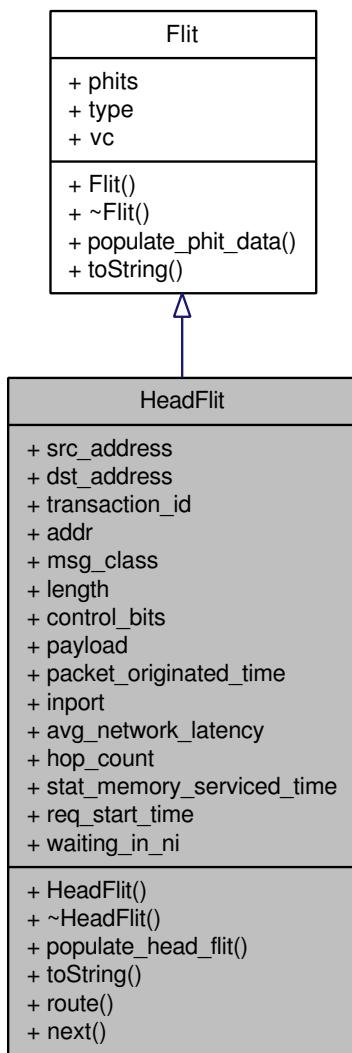
The documentation for this class was generated from the following files:

- `genericVcsInterface.h`
- `genericVcsInterface.cc`

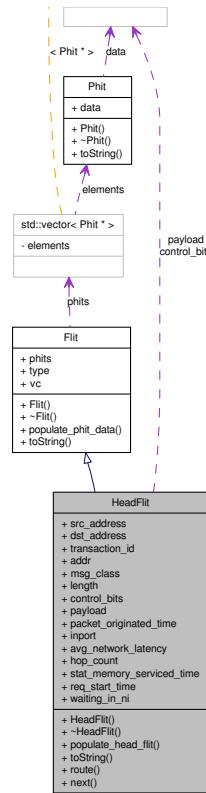
7.54 HeadFlit Class Reference

```
#include <flit.h>
```

Inheritance diagram for HeadFlit:



Collaboration diagram for HeadFlit:



Public Member Functions

- **HeadFlit ()**
- **~HeadFlit ()**
- void **populate_head_flit ()**
- string **toString () const**
- void **route ()**
- pair< **uint, uint > next ()**

Public Attributes

- **uint src_address**
- **uint dst_address**
- **uint transaction_id**

- `unsigned long long int addr`
- `message_class msg_class`
- `uint length`
- `vector< bool > control_bits`
- `vector< bool > payload`
- `simTime packet_originated_time`
- `uint import`
- `double avg_network_latency`
- `unsigned int hop_count`
- `unsigned long long int stat_memory_serviced_time`
- `unsigned long long int req_start_time`
- `unsigned long long int waiting_in_ni`

7.54.1 Detailed Description

Definition at line 85 of file flit.h.

7.54.2 Constructor & Destructor Documentation

7.54.2.1 HeadFlit::HeadFlit ()

Definition at line 93 of file flit.cc.

References dst_address, HEAD, src_address, transaction_id, and Flit::type.

7.54.2.2 HeadFlit::~HeadFlit ()

Definition at line 101 of file flit.cc.

References control_bits, payload, and Flit::phits.

7.54.3 Member Function Documentation

7.54.3.1 pair<uint, uint> HeadFlit::next ()

7.54.3.2 void HeadFlit::populate_head_flit ()

Definition at line 140 of file flit.cc.

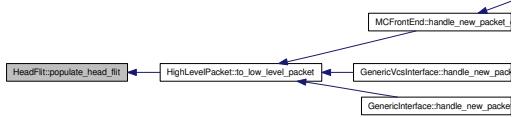
References control_bits, dst_address, max_network_node_bits, max_transaction_id_bits, payload, Flit::populate_phit_data(), src_address, and transaction_id.

Referenced by HighLevelPacket::to_low_level_packet().

Here is the call graph for this function:



Here is the caller graph for this function:



7.54.3.3 void HeadFlit::route ()

7.54.3.4 string HeadFlit::toString () const

Reimplemented from `Flit` (p. 110).

Definition at line 112 of file `flit.cc`.

References `avg_network_latency`, `control_bits`, `dst_address`, `hop_count`, `length`, `msg_class`, `payload`, `Flit::phits`, `src_address`, `transaction_id`, and `Flit::vc`.

7.54.4 Member Data Documentation

7.54.4.1 unsigned long long int HeadFlit::addr

Definition at line 93 of file `flit.h`.

Referenced by `LowLevelPacket::add()`, and `HighLevelPacket::to_low_level_packet()`.

7.54.4.2 double HeadFlit::avg_network_latency

Definition at line 107 of file `flit.h`.

Referenced by `LowLevelPacket::add()`, `GenericRouterAdaptive::do_switch_traversal()`, `GenericRouterVct::do_switch_traversal()`, `HighLevelPacket::to_low_level_packet()`, and `toString()`.

7.54.4.3 vector<bool> HeadFlit::control_bits

Definition at line 96 of file `flit.h`.

Referenced by `LowLevelPacket::add()`, `populate_head_flt()`, `HighLevelPacket::to_low_level_packet()`, `toString()`, and `~HeadFlit()`.

7.54.4.4 uint HeadFlit::dst_address

Definition at line 91 of file `flit.h`.

Referenced by LowLevelPacket::add(), HeadFlit(), populate_head_flit(), GenericRC::push(), GenericRC::route_negative_first(), GenericRC::route_north_last(), GenericRC::route_north_last_non_minimal(), GenericRC::route_west_first(), HighLevelPacket::to_low_level_packet(), and toString().

7.54.4.5 unsigned int HeadFlit::hop_count

Definition at line 108 of file flit.h.

Referenced by LowLevelPacket::add(), GenericRouterAdaptive::do_switch_traversal(), GenericRouterVct::do_switch_traversal(), HighLevelPacket::to_low_level_packet(), and toString().

7.54.4.6 uint HeadFlit::import

Definition at line 104 of file flit.h.

Referenced by GenericRouterAdaptive::handle_link_arrival_event(), and GenericRC::route_north_last_non_minimal().

7.54.4.7 uint HeadFlit::length

Definition at line 95 of file flit.h.

Referenced by LowLevelPacket::add(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), HighLevelPacket::to_low_level_packet(), and toString().

7.54.4.8 message_class HeadFlit::msg_class

Definition at line 94 of file flit.h.

Referenced by LowLevelPacket::add(), GenericRouterAdaptive::handle_link_arrival_event(), HighLevelPacket::to_low_level_packet(), and toString().

7.54.4.9 simTime HeadFlit::packet_originated_time

Definition at line 103 of file flit.h.

Referenced by LowLevelPacket::add(), and HighLevelPacket::to_low_level_packet().

7.54.4.10 vector<bool> HeadFlit::payload

Definition at line 97 of file flit.h.

Referenced by LowLevelPacket::add(), populate_head_flit(), HighLevelPacket::to_low_level_packet(), toString(), and ~HeadFlit().

7.54.4.11 unsigned long long int HeadFlit::req_start_time

Definition at line 110 of file flit.h.

Referenced by LowLevelPacket::add(), and HighLevelPacket::to_low_level_packet().

7.54.4.12 uint HeadFlit::src_address

Definition at line 90 of file flit.h.

Referenced by LowLevelPacket::add(), HeadFlit(), populate_head_flit(), HighLevelPacket::to_low_level_packet(), and toString().

7.54.4.13 unsigned long long int HeadFlit::stat_memory_serviced_time

Definition at line 109 of file flit.h.

Referenced by LowLevelPacket::add(), and HighLevelPacket::to_low_level_packet().

7.54.4.14 uint HeadFlit::transaction_id

Definition at line 92 of file flit.h.

Referenced by LowLevelPacket::add(), HeadFlit(), populate_head_flit(), HighLevelPacket::to_low_level_packet(), and toString().

7.54.4.15 unsigned long long int HeadFlit::waiting_in_ni

Definition at line 111 of file flit.h.

Referenced by LowLevelPacket::add(), and HighLevelPacket::to_low_level_packet().

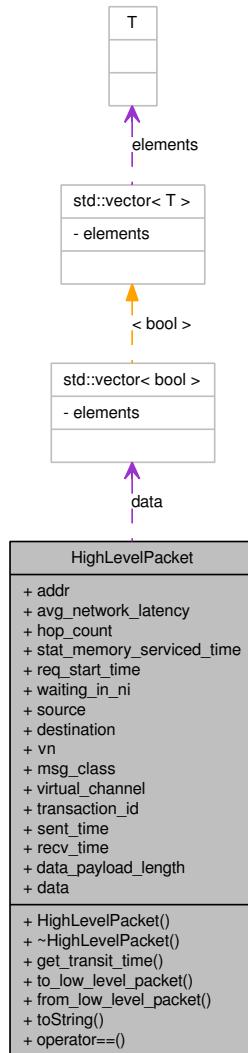
The documentation for this class was generated from the following files:

- **flit.h**
- **flit.cc**

7.55 HighLevelPacket Class Reference

```
#include <highLevelPacket.h>
```

Collaboration diagram for HighLevelPacket:



Public Member Functions

- `HighLevelPacket ()`
- `~HighLevelPacket ()`
- `simTime get_transit_time ()`
- `void to_low_level_packet (LowLevelPacket *llp)`
- `void from_low_level_packet (LowLevelPacket *llp)`
- `string toString () const`
- `bool operator==(const HighLevelPacket *p)`

Public Attributes

- unsigned long long int **addr**
- double **avg_network_latency**
- unsigned int **hop_count**
- unsigned int **stat_memory_serviced_time**
- unsigned long long int **req_start_time**
- unsigned long long int **waiting_in_ni**
- uint **source**
- uint **destination**
- **virtual_network vn**
- **message_class msg_class**
- uint **virtual_channel**
- uint **transaction_id**
- simTime **sent_time**
- simTime **recv_time**
- unsigned int **data_payload_length**
- vector< bool > **data**

7.55.1 Detailed Description

Definition at line 49 of file highLevelPacket.h.

7.55.2 Constructor & Destructor Documentation

7.55.2.1 HighLevelPacket::HighLevelPacket ()

Definition at line 31 of file highLevelPacket.cc.

References avg_network_latency, data_payload_length, destination, hop_count, INVALID_PKT, msg_class, Simulator::Now(), recv_time, req_start_time, sent_time, source, stat_memory_serviced_time, transaction_id, virtual_channel, vn, VN0, and waiting_in_ni.

Here is the call graph for this function:



7.55.2.2 HighLevelPacket::~HighLevelPacket ()

Definition at line 49 of file highLevelPacket.cc.

References data.

7.55.3 Member Function Documentation

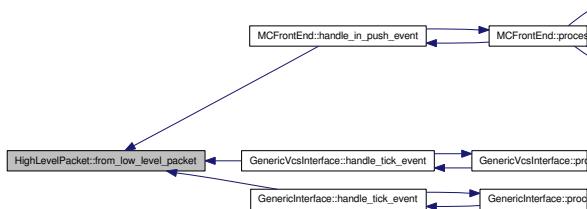
7.55.3.1 void HighLevelPacket::from_low_level_packet (LowLevelPacket * llp)

Definition at line 184 of file highLevelPacket.cc.

References LowLevelPacket::addr, addr, LowLevelPacket::avg_network_latency, avg_network_latency, LowLevelPacket::control_bits, data, data_payload_length, LowLevelPacket::destination, destination, LowLevelPacket::flits, LowLevelPacket::hop_count, hop_count, LowLevelPacket::length, LowLevelPacket::msg_class, msg_class, LowLevelPacket::payload, LowLevelPacket::req_start_time, req_start_time, LowLevelPacket::sent_time, sent_time, LowLevelPacket::source, source, LowLevelPacket::stat_memory_serviced_time, stat_memory_serviced_time, LowLevelPacket::transaction_id, transaction_id, LowLevelPacket::virtual_channel, virtual_channel, vn, VN0, VN1, VN2, LowLevelPacket::waiting_in_ni, and waiting_in_ni.

Referenced by MCFrontEnd::handle_in_push_event(), GenericVcsInterface::handle_tick_event(), and GenericInterface::handle_tick_event().

Here is the caller graph for this function:



7.55.3.2 simTime HighLevelPacket::get_transit_time ()

Definition at line 72 of file highLevelPacket.cc.

References recv_time, and sent_time.

7.55.3.3 bool HighLevelPacket::operator== (const HighLevelPacket * p)

Definition at line 77 of file highLevelPacket.cc.

References source, and transaction_id.

7.55.3.4 void HighLevelPacket::to_low_level_packet (LowLevelPacket * llp)

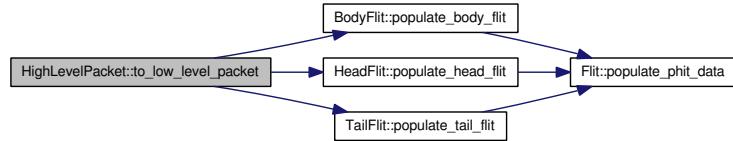
Definition at line 89 of file highLevelPacket.cc.

References HeadFlit::addr, addr, LowLevelPacket::addr, HeadFlit::avg_network_latency, avg_network_latency, LowLevelPacket::avg_network_latency, BodyFlit::bf_data, LowLevelPacket::control_bits, HeadFlit::control_bits, CONTROL_VECTOR_SIZE, data, data_payload_length, destination, LowLevelPacket::destination, HeadFlit::dst_address, LowLevelPacket::flits, HeadFlit::hop_count, hop_count, LowLevelPacket::hop_count, HeadFlit::length, LowLevelPacket::length, max_phy_link_bits, HeadFlit::msg_class, msg_class, LowLevelPacket::msg_class, ONE_FLIT_REQ, TailFlit::packet_originated_time, HeadFlit::packet_originated_time, HeadFlit::payload, BodyFlit::populate_body_flit(), HeadFlit::populate_head_flit(), TailFlit::populate_tail_flit(), HeadFlit::req_start_time, req_start_time, LowLevelPacket::req_start_time, REQUEST_PKT, RESPONSE_PKT, sent_time, LowLevelPacket::sent_time, source, LowLevelPacket::source, HeadFlit::src_address, HeadFlit::stat_memory_serviced_time, stat_memory_serviced_time,

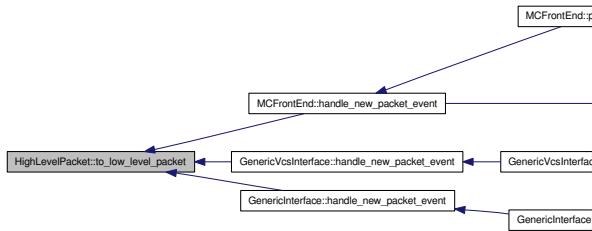
LowLevelPacket::stat_memory_serviced_time, HeadFlit::transaction_id, transaction_id, LowLevelPacket::transaction_id, Flit::vc, virtual_channel, LowLevelPacket::virtual_channel, vn, waiting_in_ni, and HeadFlit::waiting_in_ni.

Referenced by MCFrontEnd::handle_new_packet_event(), GenericVcsInterface::handle_new_packet_event(), and GenericInterface::handle_new_packet_event().

Here is the call graph for this function:



Here is the caller graph for this function:



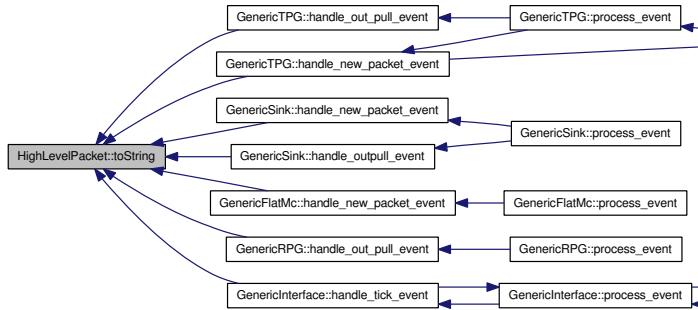
7.55.3.5 string HighLevelPacket::toString () const

Definition at line 54 of file highLevelPacket.cc.

References addr, avg_network_latency, data, data_payload_length, destination, hop_count, msg_class, sent_time, source, transaction_id, and virtual_channel.

Referenced by GenericTPG::handle_new_packet_event(), GenericSink::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericSink::handle_outpull_event(), and GenericInterface::handle_tick_event().

Here is the caller graph for this function:



7.55.4 Member Data Documentation

7.55.4.1 unsigned long long int HighLevelPacket::addr

Definition at line 70 of file `highLevelPacket.h`.

Referenced by `from_low_level_packet()`, `NI::handle_out_pull_event()`, `GenericTPG::handle_out_pull_event()`, `to_low_level_packet()`, and `toString()`.

7.55.4.2 double HighLevelPacket::avg_network_latency

Definition at line 73 of file `highLevelPacket.h`.

Referenced by `from_low_level_packet()`, `NI::handle_new_packet_event()`, `GenericTPG::handle_new_packet_event()`, `NI::handle_out_pull_event()`, `GenericVcsInterface::handle_tick_event()`, `GenericInterface::handle_tick_event()`, `HighLevelPacket()`, `to_low_level_packet()`, and `toString()`.

7.55.4.3 vector<bool> HighLevelPacket::data

Definition at line 88 of file `highLevelPacket.h`.

Referenced by `NI::convertFromBitStream()`, `MCFrontEnd::convertFromBitStream()`, `GenericTPG::convertFromBitStream()`, `GenericFlatMc::convertFromBitStream()`, `NI::convertToBitStream()`, `GenericTPG::convertToBitStream()`, `from_low_level_packet()`, `GenericRPG::handle_out_pull_event()`, `GenericFlatMc::handle_out_pull_event()`, `to_low_level_packet()`, `toString()`, and `~HighLevelPacket()`.

7.55.4.4 unsigned int HighLevelPacket::data_payload_length

Definition at line 87 of file `highLevelPacket.h`.

Referenced by `NI::convertToBitStream()`, `GenericTPG::convertToBitStream()`, `from_low_level_packet()`, `GenericRPG::handle_out_pull_event()`, `GenericFlatMc::handle_out_pull_event()`, `HighLevelPacket()`, `to_low_level_packet()`, and `toString()`.

7.55.4.5 uint HighLevelPacket::destination

Definition at line 80 of file `highLevelPacket.h`.

Referenced by from_low_level_packet(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericSink::handle_outpull_event(), HighLevelPacket(), to_low_level_packet(), and toString().

7.55.4.6 unsigned int HighLevelPacket::hop_count

Definition at line 74 of file highLevelPacket.h.

Referenced by from_low_level_packet(), NI::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), NI::handle_out_pull_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), HighLevelPacket(), to_low_level_packet(), and toString().

7.55.4.7 message_class HighLevelPacket::msg_class

Definition at line 82 of file highLevelPacket.h.

Referenced by GenericTPG::convertToBitStream(), from_low_level_packet(), NI::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), HighLevelPacket(), to_low_level_packet(), and toString().

7.55.4.8 simTime HighLevelPacket::recv_time

Definition at line 86 of file highLevelPacket.h.

Referenced by get_transit_time(), NI::handle_new_packet_event(), NI::handle_out_pull_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), and HighLevelPacket().

7.55.4.9 unsigned long long int HighLevelPacket::req_start_time

Definition at line 76 of file highLevelPacket.h.

Referenced by from_low_level_packet(), NI::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), HighLevelPacket(), and to_low_level_packet().

7.55.4.10 simTime HighLevelPacket::sent_time

Definition at line 85 of file highLevelPacket.h.

Referenced by GenericTPG::convertToBitStream(), from_low_level_packet(), get_transit_time(), GenericTPG::handle_new_packet_event(), GenericSink::handle_new_packet_event(), GenericRPG::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericSink::handle_outpull_event(), HighLevelPacket(), to_low_level_packet(), and toString().

7.55.4.11 uint HighLevelPacket::source

Definition at line 79 of file highLevelPacket.h.

Referenced by GenericTPG::convertFromBitStream(), from_low_level_packet(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericSink::handle_outpull_event(), HighLevelPacket(), operator==(), to_low_level_packet(), and toString().

7.55.4.12 unsigned int HighLevelPacket::stat_memory_serviced_time

Definition at line 75 of file highLevelPacket.h.

Referenced by from_low_level_packet(), GenericTPG::handle_new_packet_event(), NI::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), HighLevelPacket(), and to_low_level_packet().

7.55.4.13 uint HighLevelPacket::transaction_id

Definition at line 84 of file highLevelPacket.h.

Referenced by from_low_level_packet(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericSink::handle_outpull_event(), HighLevelPacket(), operator==(), to_low_level_packet(), and toString().

7.55.4.14 uint HighLevelPacket::virtual_channel

Definition at line 83 of file highLevelPacket.h.

Referenced by from_low_level_packet(), MCFrontEnd::handle_new_packet_event(), GenericVcsInterface::handle_new_packet_event(), GenericSink::handle_new_packet_event(), GenericRPG::handle_new_packet_event(), GenericInterface::handle_new_packet_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericSink::handle_outpull_event(), HighLevelPacket(), to_low_level_packet(), and toString().

7.55.4.15 virtual_network HighLevelPacket::vn

Definition at line 81 of file highLevelPacket.h.

Referenced by from_low_level_packet(), HighLevelPacket(), and to_low_level_packet().

7.55.4.16 unsigned long long int HighLevelPacket::waiting_in_ni

Definition at line 77 of file highLevelPacket.h.

Referenced by from_low_level_packet(), GenericTPG::handle_new_packet_event(), NI::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), HighLevelPacket(), and to_low_level_packet().

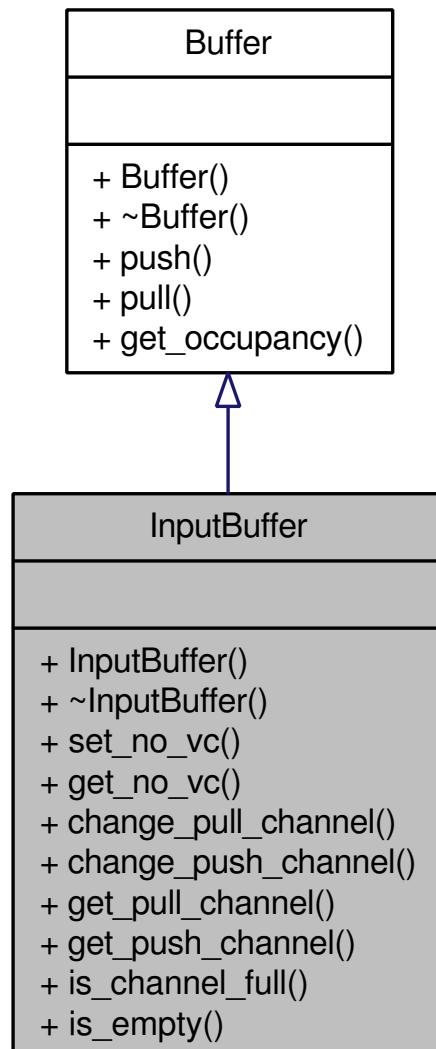
The documentation for this class was generated from the following files:

- **highLevelPacket.h**
- **highLevelPacket.cc**

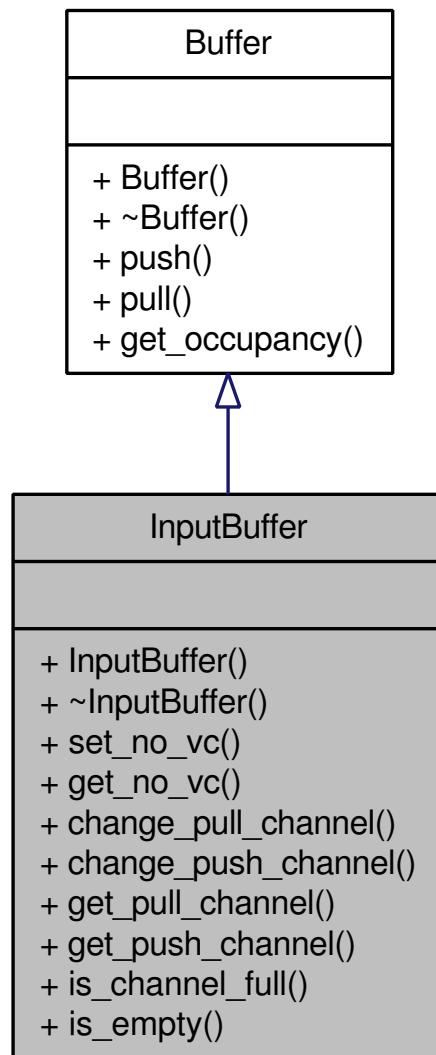
7.56 InputBuffer Class Reference

```
#include <inputBuffer.h>
```

Inheritance diagram for InputBuffer:



Collaboration diagram for InputBuffer:



Public Member Functions

- **InputBuffer ()**
- virtual **~InputBuffer ()**
- virtual void **set_no_vc (uint vc)**
- virtual **uint get_no_vc () const**
- virtual void **change_pull_channel (uint channel)**
- virtual void **change_push_channel (uint channel)**
- virtual **uint get_pull_channel () const**
- virtual **uint get_push_channel () const**
- virtual bool **is_channel_full (uint channel) const**
- virtual bool **is_empty (uint channel) const**

7.56.1 Detailed Description

Definition at line 32 of file inputBuffer.h.

7.56.2 Constructor & Destructor Documentation

7.56.2.1 InputBuffer::InputBuffer () [inline]

Definition at line 35 of file inputBuffer.h.

7.56.2.2 virtual InputBuffer::~InputBuffer () [inline, virtual]

Definition at line 36 of file inputBuffer.h.

7.56.3 Member Function Documentation

7.56.3.1 virtual void InputBuffer::change_pull_channel (uint *channel*) [virtual]

7.56.3.2 virtual void InputBuffer::change_push_channel (uint *channel*) [virtual]

7.56.3.3 virtual uint InputBuffer::get_no_vc () const [virtual]

7.56.3.4 virtual uint InputBuffer::get_pull_channel () const [virtual]

7.56.3.5 virtual uint InputBuffer::get_push_channel () const [virtual]

7.56.3.6 virtual bool InputBuffer::is_channel_full (uint *channel*) const [virtual]

7.56.3.7 virtual bool InputBuffer::is_empty (uint *channel*) const [virtual]

7.56.3.8 virtual void InputBuffer::set_no_vc (uint *vc*) [virtual]

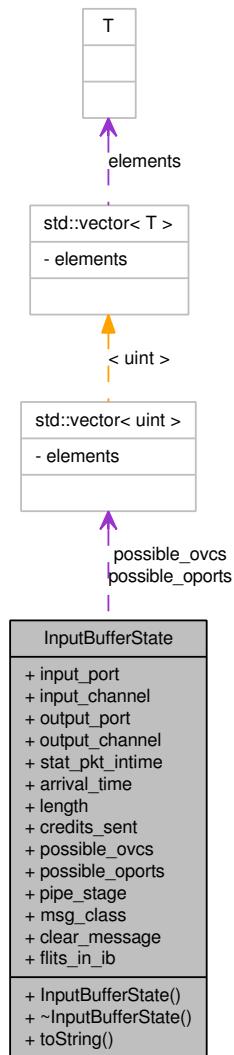
The documentation for this class was generated from the following file:

- **inputBuffer.h**

7.57 InputBufferState Class Reference

```
#include <genericData.h>
```

Collaboration diagram for InputBufferState:



Public Member Functions

- **InputBufferState ()**
- **~InputBufferState ()**
- string **toString () const**

Public Attributes

- **uint input_port**
- **uint input_channel**
- **uint output_port**

- **uint output_channel**
- **double stat_pkt_intime**
- **double arrival_time**
- **int length**
- **int credits_sent**
- **vector< uint > possible_ovcs**
- **vector< uint > possible_oports**
- **RouterPipeStage pipe_stage**
- **message_class msg_class**
- **bool clear_message**
- **uint flits_in_ib**

7.57.1 Detailed Description

Definition at line 29 of file genericData.h.

7.57.2 Constructor & Destructor Documentation

7.57.2.1 InputBufferState::InputBufferState ()

Definition at line 80 of file genericData.cc.

References INVALID, and pipe_stage.

7.57.2.2 InputBufferState::~InputBufferState () [inline]

Definition at line 33 of file genericData.h.

7.57.3 Member Function Documentation

7.57.3.1 string InputBufferState::toString () const

Definition at line 38 of file genericData.cc.

References EMPTY, FULL, IB, input_channel, input_port, INVALID, output_channel, output_port, pipe_stage, REQ_OUTVC_ARB, ROUTED, ST, SW_ALLOCATED, and SWA_REQUESTED.

7.57.4 Member Data Documentation

7.57.4.1 double InputBufferState::arrival_time

Definition at line 39 of file genericData.h.

7.57.4.2 bool InputBufferState::clear_message

Definition at line 46 of file genericData.h.

7.57.4.3 int InputBufferState::credits_sent

Definition at line 41 of file genericData.h.

7.57.4.4 uint InputBufferState::flts_in_ib

Definition at line 47 of file genericData.h.

7.57.4.5 uint InputBufferState::input_channel

Definition at line 35 of file genericData.h.

Referenced by `toString()`.

7.57.4.6 uint InputBufferState::input_port

Definition at line 34 of file genericData.h.

Referenced by `toString()`.

7.57.4.7 int InputBufferState::length

Definition at line 40 of file genericData.h.

7.57.4.8 message_class InputBufferState::msg_class

Definition at line 45 of file genericData.h.

7.57.4.9 uint InputBufferState::output_channel

Definition at line 37 of file genericData.h.

Referenced by `toString()`.

7.57.4.10 uint InputBufferState::output_port

Definition at line 36 of file genericData.h.

Referenced by `toString()`.

7.57.4.11 RouterPipeStage InputBufferState::pipe_stage

Definition at line 44 of file genericData.h.

Referenced by `InputBufferState()`, and `toString()`.

7.57.4.12 vector< uint > InputBufferState::possible_oports

Definition at line 43 of file genericData.h.

7.57.4.13 vector< uint > InputBufferState::possible_ovcs

Definition at line 42 of file genericData.h.

7.57.4.14 double InputBufferState::stat_pkt_intime

Definition at line 38 of file genericData.h.

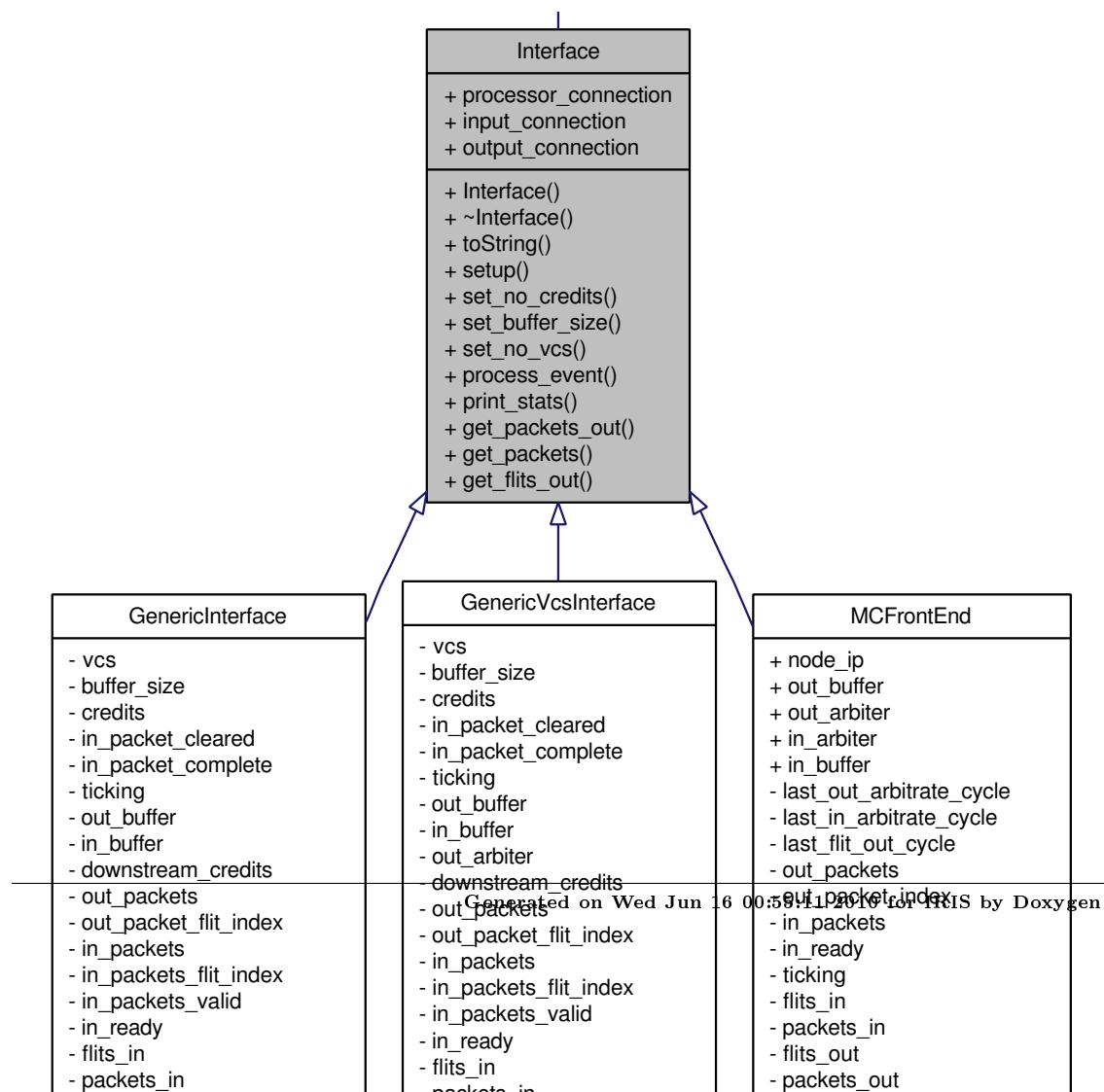
The documentation for this class was generated from the following files:

- **genericData.h**
- **genericData.cc**

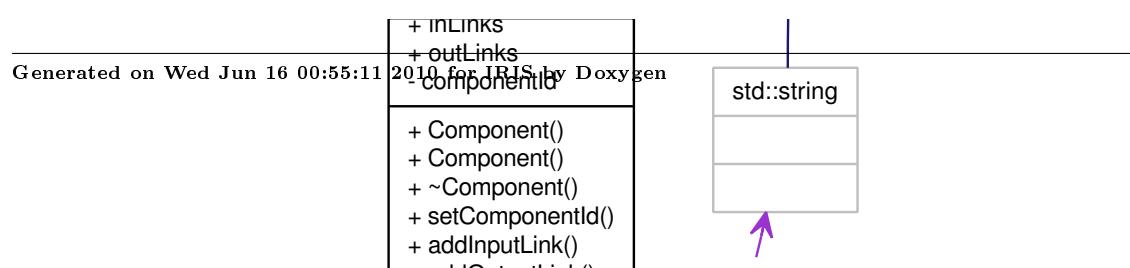
7.58 Interface Class Reference

```
#include <interface.h>
```

Inheritance diagram for Interface:



Collaboration diagram for Interface:



Public Member Functions

- **Interface ()**
- virtual ~**Interface ()**
- virtual string **toString () const**
- virtual void **setup (uint a, uint b)=0**
- virtual void **set_no_credits (int cr)=0**
- virtual void **set_buffer_size (uint cr)=0**
- virtual void **set_no_vcs (uint cr)=0**
- virtual void **process_event (IrisEvent *e)=0**
- virtual string **print_stats ()=0**
- virtual ullint **get_packets_out ()=0**
- virtual ullint **get_packets ()=0**
- virtual ullint **get_flits_out ()=0**

Public Attributes

- NetworkComponent * **processor_connection**
- IrisLink * **input_connection**
- IrisLink * **output_connection**

7.58.1 Detailed Description

Definition at line 33 of file interface.h.

7.58.2 Constructor & Destructor Documentation

7.58.2.1 Interface::Interface ()

Definition at line 31 of file interface.cc.

References NetworkComponent::address, NetworkComponent::interface, and NetworkComponent::type.

7.58.2.2 Interface::~Interface () [virtual]

Definition at line 37 of file interface.cc.

7.58.3 Member Function Documentation

7.58.3.1 virtual ullint Interface::get_flits_out () [pure virtual]

Implemented in **GenericInterface** (p. 134), and **GenericVcsInterface** (p. 207).

7.58.3.2 virtual ullint Interface::get_packets () [pure virtual]

Implemented in **GenericInterface** (p. 134), and **GenericVcsInterface** (p. 207).

7.58.3.3 virtual ullint Interface::get_packets_out () [pure virtual]

Implemented in **GenericInterface** (p. 134), and **GenericVcsInterface** (p. 208).

7.58.3.4 virtual string Interface::print_stats () [pure virtual]

Implemented in **MCFrontEnd** (p. 255), **GenericInterface** (p. 136), and **GenericVcsInterface** (p. 209).

7.58.3.5 virtual void Interface::process_event (IrisEvent * e) [pure virtual]

Implements **NetworkComponent** (p. 274).

Implemented in **MCFrontEnd** (p. 255), **GenericInterface** (p. 136), and **GenericVcsInterface** (p. 210).

7.58.3.6 virtual void Interface::set_buffer_size (uint cr) [pure virtual]

Implemented in **GenericInterface** (p. 137), and **GenericVcsInterface** (p. 210).

7.58.3.7 virtual void Interface::set_no_credits (int cr) [pure virtual]

Implemented in **GenericInterface** (p. 137), and **GenericVcsInterface** (p. 210).

7.58.3.8 virtual void Interface::set_no_vcs (uint cr) [pure virtual]

Implemented in **GenericInterface** (p. 137), and **GenericVcsInterface** (p. 210).

7.58.3.9 virtual void Interface::setup (uint a, uint b) [pure virtual]

Implemented in **GenericInterface** (p. 137), and **GenericVcsInterface** (p. 211).

7.58.3.10 string Interface::toString () const [virtual]

Reimplemented from **NetworkComponent** (p. 275).

Reimplemented in **MCFrontEnd** (p. 256), **GenericInterface** (p. 138), and **GenericVcsInterface** (p. 211).

Definition at line 42 of file interface.cc.

References `NetworkComponent::address`, `input_connection`, `output_connection`, `processor_connection`, and `NetworkComponent::toString()`.

Here is the call graph for this function:



7.58.4 Member Data Documentation

7.58.4.1 IrisLink* Interface::input_connection

Definition at line 37 of file interface.h.

Referenced by MCFrontEnd::handle_in_arbitrate_event(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), and toString().

7.58.4.2 IrisLink* Interface::output_connection

Definition at line 38 of file interface.h.

Referenced by MCFrontEnd::handle_flist_out_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), and toString().

7.58.4.3 NetworkComponent* Interface::processor_connection

Definition at line 36 of file interface.h.

Referenced by MCFrontEnd::handle_out_arbitrate_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), MCFrontEnd::setup(), and toString().

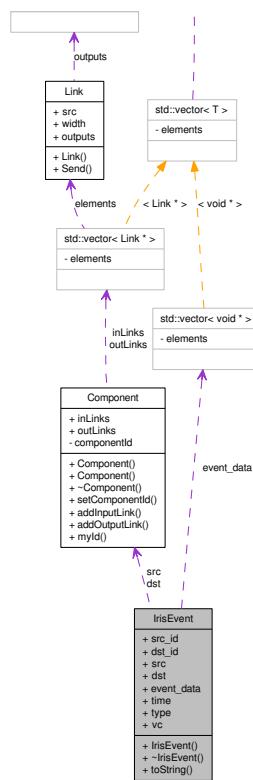
The documentation for this class was generated from the following files:

- **interface.h**
- **interface.cc**

7.59 IrisEvent Class Reference

```
#include <irisEvent.h>
```

Collaboration diagram for IrisEvent:



Public Member Functions

- **IrisEvent ()**
- **~IrisEvent ()**
- string **toString (void)**

Public Attributes

- uint **src_id**
- uint **dst_id**
- Component * **src**

- **Component * dst**
- **vector< void * > event _ data**
- **simTime time**
- **uint type**
- **uint vc**

7.59.1 Detailed Description

Definition at line 38 of file irisEvent.h.

7.59.2 Constructor & Destructor Documentation

7.59.2.1 IrisEvent::IrisEvent ()

Definition at line 34 of file irisEvent.cc.

References src_id, and vc.

7.59.2.2 IrisEvent::~IrisEvent ()

Definition at line 40 of file irisEvent.cc.

7.59.3 Member Function Documentation

7.59.3.1 string IrisEvent::toString (void)

Definition at line 24 of file irisEvent.cc.

References Simulator::Now(), and vc.

Here is the call graph for this function:



7.59.4 Member Data Documentation

7.59.4.1 Component* IrisEvent::dst

Definition at line 46 of file irisEvent.h.

Referenced by NI::handle_new_packet_event(), NI::handle_out_pull_event(), main(), ResponseHandler::process_event(), RequestHandler::process_event(), RefreshMgr::process_event(), MSHR_H::process_event(), CmdIssuer::process_event(), and BankHandler::process_event().

7.59.4.2 uint IrisEvent::dst_id

Definition at line 44 of file irisEvent.h.

Referenced by GenericLink::handle_link_arrival_event().

7.59.4.3 `vector<void *> IrisEvent::event_data`

Definition at line 47 of file irisEvent.h.

Referenced by MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericLink::handle_link_arrival_event(), NI::handle_new_packet_event(), MCFrontEnd::handle_new_packet_event(), GenericVcsInterface::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), GenericSink::handle_new_packet_event(), GenericRPG::handle_new_packet_event(), GenericInterface::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), MCFrontEnd::handle_ready_event(), GenericSink::handle_ready_event(), GenericRPG::handle_ready_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), main(), ResponseHandler::process_event(), RequestHandler::process_event(), MSHR_H::process_event(), DRAMChannel::process_event(), DataBusHandler::process_event(), CmdIssuer::process_event(), CmdBusHandler::process_event(), BankHandler::process_event(), and GenericSink::setup().

7.59.4.4 `Component* IrisEvent::src`

Definition at line 45 of file irisEvent.h.

Referenced by NI::handle_new_packet_event(), main(), RequestHandler::process_event(), RefreshMgr::process_event(), MSHR_H::process_event(), CmdIssuer::process_event(), and BankHandler::process_event().

7.59.4.5 `uint IrisEvent::src_id`

Definition at line 43 of file irisEvent.h.

Referenced by RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericLink::handle_link_arrival_event(), NI::handle_new_packet_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), and IrisEvent().

7.59.4.6 `simTime IrisEvent::time`

Definition at line 48 of file irisEvent.h.

7.59.4.7 `uint IrisEvent::type`

Definition at line 49 of file irisEvent.h.

Referenced by MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_in_push_event(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), NI::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), main(), RouterFourStageVcs::process_event(), ResponseHandler::process_event(), RequestHandler::process_event(), NI::process_event(),

MSHR_H::process_event(), MCFrontEnd::process_event(), GenericVcsInterface::process_event(), GenericTPG::process_event(), GenericSink::process_event(), GenericRPG::process_event(), GenericRouterVct::process_event(), GenericRouterNoVcs::process_event(), GenericRouterAdaptive::process_event(), GenericLink::process_event(), GenericInterface::process_event(), GenericFlatMc::process_event(), DRAMChannel::process_event(), DataBusHandler::process_event(), CmdIssuer::process_event(), BankHandler::process_event(), and GenericSink::setup().

7.59.4.8 uint IrisEvent::vc

Definition at line 50 of file irisEvent.h.

Referenced by MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_in_push_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericVcsInterface::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), GenericInterface::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), MCFrontEnd::handle_ready_event(), GenericVcsInterface::handle_ready_event(), GenericSink::handle_ready_event(), GenericRPG::handle_ready_event(), GenericInterface::handle_ready_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterVct::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericRouterAdaptive::handle_tick_event(), GenericInterface::handle_tick_event(), IrisEvent(), and toString().

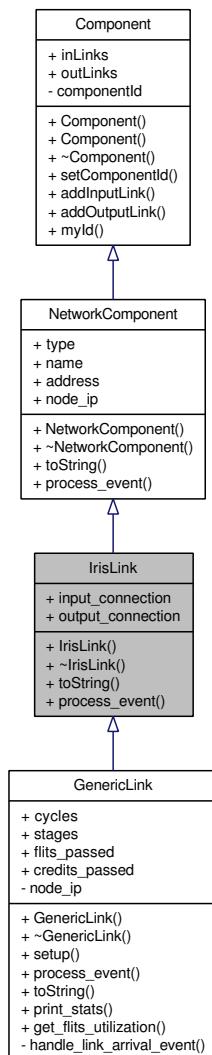
The documentation for this class was generated from the following files:

- **irisEvent.h**
- **irisEvent.cc**

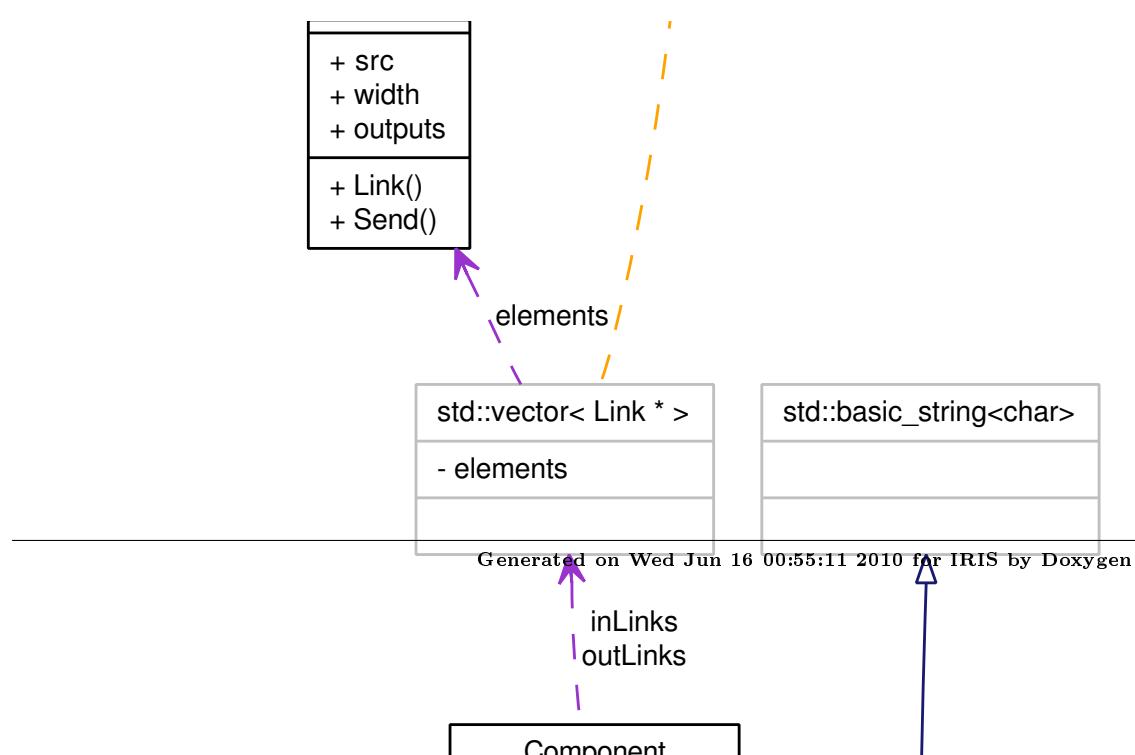
7.60 IrisLink Class Reference

```
#include <irisLink.h>
```

Inheritance diagram for IrisLink:



Collaboration diagram for IrisLink:



Public Member Functions

- `IrisLink ()`
- `~IrisLink ()`
- `string toString () const`
- `virtual void process_event (IrisEvent *e)=0`

Public Attributes

- `NetworkComponent * input_connection`
- `NetworkComponent * output_connection`

7.60.1 Detailed Description

Definition at line 33 of file irisLink.h.

7.60.2 Constructor & Destructor Documentation

7.60.2.1 IrisLink::IrisLink () [inline]

Definition at line 36 of file irisLink.h.

7.60.2.2 IrisLink::~IrisLink () [inline]

Definition at line 37 of file irisLink.h.

7.60.3 Member Function Documentation

7.60.3.1 virtual void IrisLink::process_event (IrisEvent * e) [pure virtual]

Implements `NetworkComponent` (p. 274).

Implemented in `GenericLink` (p. 142).

7.60.3.2 string IrisLink::toString () const [virtual]

Reimplemented from `NetworkComponent` (p. 275).

Reimplemented in `GenericLink` (p. 143).

Definition at line 32 of file irisLink.cc.

References `input_connection`, and `output_connection`.

7.60.4 Member Data Documentation

7.60.4.1 NetworkComponent* IrisLink::input_connection

Definition at line 38 of file irisLink.h.

Referenced by `GenericLink::handle_link_arrival_event()`, `toString()`, and `GenericLink::toString()`.

7.60.4.2 NetworkComponent* IrisLink::output_connection

Definition at line 39 of file irisLink.h.

Referenced by GenericLink::handle_link_arrival_event(), toString(), and GenericLink::toString().

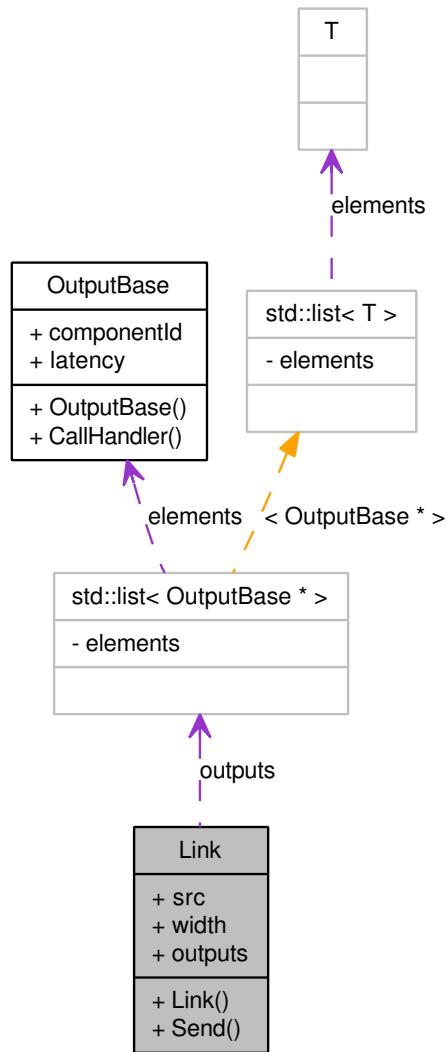
The documentation for this class was generated from the following files:

- [irisLink.h](#)
- [irisLink.cc](#)

7.61 Link Class Reference

```
#include <link.h>
```

Collaboration diagram for Link:



Public Member Functions

- **Link** (int srcComponentId, int linkWidth)
- void **Send** (uint64_t data, int srcComponentId)

Public Attributes

- int **src**
- int **width**
- list< **OutputBase** * > **outputs**

7.61.1 Detailed Description

Definition at line 35 of file link.h.

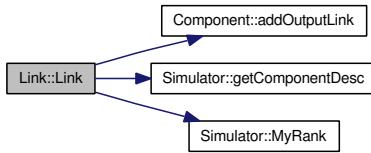
7.61.2 Constructor & Destructor Documentation

7.61.2.1 Link::Link (int *srcComponentId*, int *linkWidth*)

Definition at line 6 of file link.cc.

References Component::addOutputLink(), Simulator::getComponentDesc(), ComponentDescription::lpId, Simulator::MyRank(), ComponentDescription::ptr, src, and width.

Here is the call graph for this function:



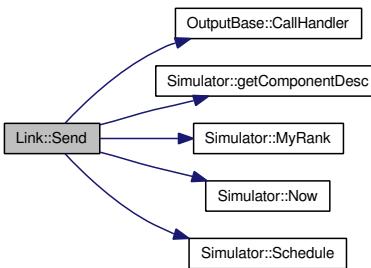
7.61.3 Member Function Documentation

7.61.3.1 void Link::Send (uint64_t *data*, int *srcComponentId*)

Definition at line 19 of file link.cc.

References OutputBase::CallHandler(), Simulator::getComponentDesc(), ComponentDescription::lpId, Simulator::MyRank(), Simulator::Now(), outputs, and Simulator::Schedule().

Here is the call graph for this function:



7.61.4 Member Data Documentation

7.61.4.1 list<OutputBase*> Link::outputs

Definition at line 42 of file link.h.

Referenced by addOutput(), and Send().

7.61.4.2 int Link::src

Definition at line 39 of file link.h.

Referenced by Link().

7.61.4.3 int Link::width

Definition at line 40 of file link.h.

Referenced by Link().

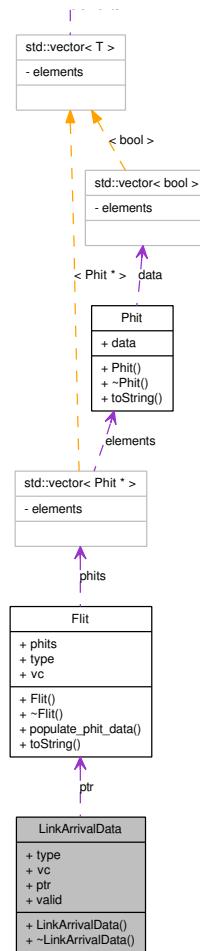
The documentation for this class was generated from the following files:

- **link.h**
- **link.cc**

7.62 LinkArrivalData Class Reference

```
#include <genericData.h>
```

Collaboration diagram for LinkArrivalData:



Public Member Functions

- `LinkArrivalData ()`
- `~LinkArrivalData ()`

Public Attributes

- `uint type`
- `uint vc`
- `Flit * ptr`
- `bool valid`

7.62.1 Detailed Description

Definition at line 58 of file genericData.h.

7.62.2 Constructor & Destructor Documentation

7.62.2.1 LinkArrivalData::LinkArrivalData ()

Definition at line 24 of file genericData.cc.

References valid.

7.62.2.2 LinkArrivalData::~LinkArrivalData ()

Definition at line 29 of file genericData.cc.

7.62.3 Member Data Documentation

7.62.3.1 Flit* LinkArrivalData::ptr

Definition at line 65 of file genericData.h.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), and GenericInterface::handle_tick_event().

7.62.3.2 uint LinkArrivalData::type

Definition at line 63 of file genericData.h.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericLink::handle_link_arrival_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericInterface::handle_tick_event(), GenericRouterVct::send_credit_back(), GenericRouterNoVcs::send_credit_back(), and GenericRouterAdaptive::send_credit_back().

7.62.3.3 bool LinkArrivalData::valid

Definition at line 66 of file genericData.h.

Referenced by MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(),

GenericInterface::handle_link_arrival(), and LinkArrivalData().

7.62.3.4 uint LinkArrivalData::vc

Definition at line 64 of file genericData.h.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericLink::handle_link_arrival_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericInterface::handle_tick_event(), GenericRouterVct::send_credit_back(), GenericRouterNoVcs::send_credit_back(), and GenericRouterAdaptive::send_credit_back().

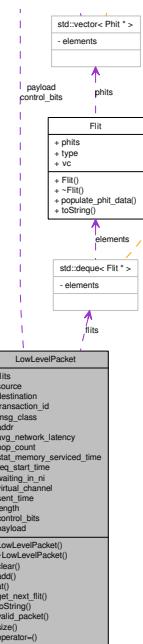
The documentation for this class was generated from the following files:

- **genericData.h**
- **genericData.cc**

7.63 LowLevelPacket Class Reference

```
#include <lowLevelPacket.h>
```

Collaboration diagram for LowLevelPacket:



Public Member Functions

- **LowLevelPacket ()**
- **~LowLevelPacket ()**
- **void clear ()**
- **void add (Flit *ptr)**
- **Flit * at (uint index)**
- **Flit * get_next_flit ()**
- **string toString () const**
- **bool valid_packet ()**
- **uint size ()**
- **void operator= (const LowLevelPacket *p)**

Public Attributes

- `deque< Flit * > flits`
- `uint source`
- `uint destination`
- `uint transaction_id`
- `message_class msg_class`
- `unsigned long long int addr`
- `double avg_network_latency`
- `unsigned int hop_count`
- `unsigned int stat_memory_serviced_time`
- `unsigned long long int req_start_time`
- `unsigned long long int waiting_in_ni`
- `short int virtual_channel`
- `unsigned long int sent_time`
- `unsigned int length`
- `vector< bool > control_bits`
- `vector< bool > payload`

7.63.1 Detailed Description

Definition at line 44 of file lowLevelPacket.h.

7.63.2 Constructor & Destructor Documentation

7.63.2.1 LowLevelPacket::LowLevelPacket ()

Definition at line 27 of file lowLevelPacket.cc.

References `destination`, `length`, `sent_time`, `source`, `transaction_id`, and `virtual_channel`.

7.63.2.2 LowLevelPacket::~LowLevelPacket ()

Definition at line 38 of file lowLevelPacket.cc.

References `control_bits`, `flits`, and `payload`.

7.63.3 Member Function Documentation

7.63.3.1 void LowLevelPacket::add (Flit * *ptr*)

Definition at line 102 of file lowLevelPacket.cc.

References `HeadFlit::addr`, `addr`, `TailFlit::avg_network_latency`, `HeadFlit::avg_network_latency`, `avg_network_latency`, `BodyFlit::bf_data`, `BODY`, `control_bits`, `HeadFlit::control_bits`, `destination`, `HeadFlit::dst_address`, `HEAD`, `HeadFlit::hop_count`, `hop_count`, `HeadFlit::length`, `length`, `HeadFlit::msg_class`, `msg_class`, `TailFlit::packet_originated_time`, `HeadFlit::packet_originated_time`, `payload`, `HeadFlit::payload`, `HeadFlit::req_start_time`, `req_start_time`, `sent_time`, `source`, `HeadFlit::src_address`, `HeadFlit::stat_memory_serviced_time`, `stat_memory_serviced_time`, `TAIL`, `HeadFlit::transaction_id`, `transaction_id`, `Flit::type`, `Flit::vc`, `virtual_channel`, `HeadFlit::waiting_in_ni`, and `waiting_in_ni`.

7.63.3.2 Flit * LowLevelPacket::at (uint *index*)

Definition at line 158 of file lowLevelPacket.cc.

References flits.

7.63.3.3 void LowLevelPacket::clear ()

Definition at line 175 of file lowLevelPacket.cc.

References flits.

7.63.3.4 Flit * LowLevelPacket::get_next_flit ()

Definition at line 166 of file lowLevelPacket.cc.

References flits.

7.63.3.5 void LowLevelPacket::operator= (const LowLevelPacket * *p*)

Definition at line 45 of file lowLevelPacket.cc.

References control_bits, destination, length, msg_class, payload, sent_time, source, transaction_id, and virtual_channel.

7.63.3.6 uint LowLevelPacket::size ()

Definition at line 182 of file lowLevelPacket.cc.

References flits.

7.63.3.7 string LowLevelPacket::toString (void) const

Definition at line 62 of file lowLevelPacket.cc.

References addr, avg_network_latency, BODY, destination, flits, HEAD, hop_count, length, msg_class, sent_time, source, TAIL, and virtual_channel.

7.63.3.8 bool LowLevelPacket::valid_packet ()

Definition at line 92 of file lowLevelPacket.cc.

References flits, and length.

7.63.4 Member Data Documentation**7.63.4.1 unsigned long long int LowLevelPacket::addr**

Definition at line 57 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.2 double LowLevelPacket::avg_network_latency

Definition at line 60 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.3 vector<bool> LowLevelPacket::control_bits

Definition at line 69 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), operator=(), HighLevelPacket::to_low_level_packet(), and ~LowLevelPacket().

7.63.4.4 uint LowLevelPacket::destination

Definition at line 53 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), LowLevelPacket(), operator=(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.5 deque<Flit*> LowLevelPacket::flits

Definition at line 51 of file lowLevelPacket.h.

Referenced by at(), clear(), HighLevelPacket::from_low_level_packet(), get_next_flit(), size(), HighLevelPacket::to_low_level_packet(), toString(), valid_packet(), and ~LowLevelPacket().

7.63.4.6 unsigned int LowLevelPacket::hop_count

Definition at line 61 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.7 unsigned int LowLevelPacket::length

Definition at line 68 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), LowLevelPacket(), operator=(), HighLevelPacket::to_low_level_packet(), toString(), and valid_packet().

7.63.4.8 message_class LowLevelPacket::msg_class

Definition at line 55 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), operator=(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.9 vector<bool> LowLevelPacket::payload

Definition at line 70 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), operator=(), and ~LowLevelPacket().

7.63.4.10 unsigned long long int LowLevelPacket::req_start_time

Definition at line 63 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), and HighLevelPacket::to_low_level_packet().

7.63.4.11 unsigned long int LowLevelPacket::sent_time

Definition at line 67 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), LowLevelPacket(), operator=(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.12 uint LowLevelPacket::source

Definition at line 52 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), LowLevelPacket(), operator=(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.13 unsigned int LowLevelPacket::stat_memory_serviced_time

Definition at line 62 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), and HighLevelPacket::to_low_level_packet().

7.63.4.14 uint LowLevelPacket::transaction_id

Definition at line 54 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), LowLevelPacket(), operator=(), and HighLevelPacket::to_low_level_packet().

7.63.4.15 short int LowLevelPacket::virtual_channel

Definition at line 66 of file lowLevelPacket.h.

Referenced by add(), HighLevelPacket::from_low_level_packet(), LowLevelPacket(), operator=(), HighLevelPacket::to_low_level_packet(), and toString().

7.63.4.16 unsigned long long int LowLevelPacket::waiting_in_ni

Definition at line 64 of file lowLevelPacket.h.

Referenced by add(), and HighLevelPacket::from_low_level_packet().

The documentation for this class was generated from the following files:

- `lowLevelPacket.h`
- `lowLevelPacket.cc`

7.64 MC Class Reference

```
#include <MC.h>
```

Collaboration diagram for MC:

Public Member Functions

- void **Init** ()
- void **StartRefresh** ()
- ~**MC** ()

Public Attributes

- NI * ni
- RequestHandler * reqH
- RefreshMgr * refMgr
- Bus * bus
- DRAM * dram
- ResponseHandler * responseH
- Statistic * stats
- Component * parent
- bool * doneOnce [NO_OF_THREADS]
- UInt id

7.64.1 Detailed Description

Definition at line 35 of file MC.h.

7.64.2 Constructor & Destructor Documentation

7.64.2.1 MC::~MC () [inline]

Definition at line 60 of file MC.h.

References bus, dram, refMgr, reqH, and responseH.

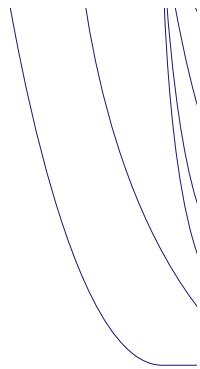
7.64.3 Member Function Documentation

7.64.3.1 void MC::Init ()

Definition at line 20 of file MC.cc.

References bus, ResponseHandler::child, RequestHandler::child, Bus::child1, Bus::child2, dram, Statistic::InitStats(), Statistic::mc, RefreshMgr::mc, ResponseHandler::mc, DRAM::mc, Bus::mc, RequestHandler::mc, ni, ResponseHandler::parent, DRAM::parent, Bus::parent, RequestHandler::parent, refMgr, reqH, RefreshMgr::reqPtr, ResponseHandler::reqPtr, responseH, RequestHandler::resPtr, DRAM::SetLinks(), Bus::SetLinks(), RequestHandler::SetLinks(), StartRefresh(), and stats.

Here is the call graph for this function:



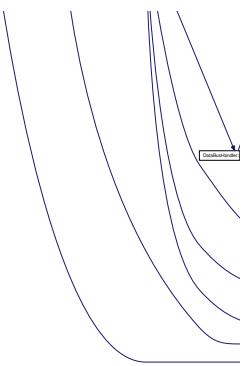
7.64.3.2 void MC::StartRefresh ()

Definition at line 49 of file MC.cc.

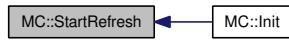
References RefreshMgr::nextRefresh, Simulator::Now(), RefreshMgr::process_event(), refMgr, REFRESH_INC, and Simulator::Schedule().

Referenced by Init().

Here is the call graph for this function:



Here is the caller graph for this function:



7.64.4 Member Data Documentation

7.64.4.1 Bus* MC::bus

Definition at line 41 of file MC.h.

Referenced by Init(), and ~MC().

7.64.4.2 bool* MC::doneOnce[NO_OF_THREADS]

Definition at line 46 of file MC.h.

7.64.4.3 DRAM* MC::dram

Definition at line 42 of file MC.h.

Referenced by Init(), and ~MC().

7.64.4.4 UInt MC::id

Definition at line 49 of file MC.h.

7.64.4.5 NI* MC::ni

Definition at line 38 of file MC.h.

Referenced by Init(), and main().

7.64.4.6 Component* MC::parent

Definition at line 45 of file MC.h.

7.64.4.7 RefreshMgr* MC::refMgr

Definition at line 40 of file MC.h.

Referenced by Init(), StartRefresh(), and ~MC().

7.64.4.8 RequestHandler* MC::reqH

Definition at line 39 of file MC.h.

Referenced by Init(), main(), and ~MC().

7.64.4.9 ResponseHandler* MC::responseH

Definition at line 43 of file MC.h.

Referenced by Init(), and ~MC().

7.64.4.10 Statistic* MC::stats

Definition at line 44 of file MC.h.

Referenced by Init(), and main().

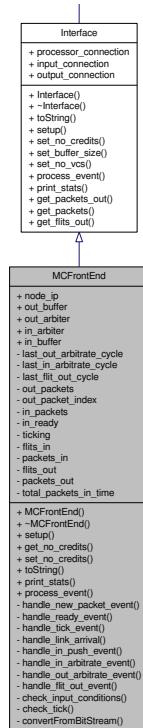
The documentation for this class was generated from the following files:

- **MC.h**
- **MC.cc**

7.65 MCFrontEnd Class Reference

```
#include <mcFrontEnd.h>
```

Inheritance diagram for MCFrontEnd:



Collaboration diagram for MCFrontEnd:

Public Member Functions

- `MCFrontEnd ()`
- `~MCFrontEnd ()`
- `void setup ()`
- `uint get_no_credits () const`
- `void set_no_credits (uint credits)`
- `string toString () const`
- `string print_stats ()`
- `void process_event (IrisEvent *e)`

Public Attributes

- `uint node_ip`
- `GenericOutputBuffer out_buffer`
- `GenericArbiter out_arbiter`
- `GenericArbiter in_arbiter`
- `GenericOutputBuffer in_buffer`

Private Member Functions

- `void handle_new_packet_event (IrisEvent *e)`
- `void handle_ready_event (IrisEvent *e)`
- `void handle_tick_event (IrisEvent *e)`
- `void handle_link_arrival (IrisEvent *e)`
- `void handle_in_push_event (IrisEvent *e)`
- `void handle_in_arbitrate_event (IrisEvent *e)`
- `void handle_out_arbitrate_event (IrisEvent *e)`
- `void handle_flit_out_event (IrisEvent *e)`
- `uint check_input_conditions ()`
- `uint check_tick ()`
- `void convertFromBitStream (Request *req, HighLevelPacket *hlp)`

Private Attributes

- `unsigned long long int last_out_arbitrate_cycle`
- `unsigned long long int last_in_arbitrate_cycle`
- `unsigned long long int last_flit_out_cycle`
- `vector< LowLevelPacket > out_packets`
- `vector< uint > out_packet_index`
- `vector< LowLevelPacket > in_packets`
- `vector< bool > in_ready`
- `bool ticking`
- `uint flits_in`
- `uint packets_in`
- `uint flits_out`
- `uint packets_out`
- `uint total_packets_in_time`

7.65.1 Detailed Description

Definition at line 38 of file mcFrontEnd.h.

7.65.2 Constructor & Destructor Documentation

7.65.2.1 MCFrontEnd::MCFrontEnd ()

Definition at line 25 of file mcFrontEnd.cc.

References NetworkComponent::name.

7.65.2.2 MCFrontEnd::~MCFrontEnd ()

Definition at line 30 of file mcFrontEnd.cc.

7.65.3 Member Function Documentation

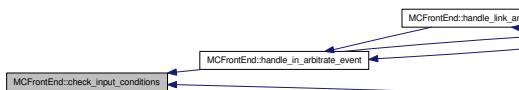
7.65.3.1 uint MCFrontEnd::check_input_conditions () [private]

Definition at line 525 of file mcFrontEnd.cc.

References in_arbiter, IN_ARBITRATE_EVENT, in_buffer, in_packets, IN_PUSH_EVENT, and in_ready.

Referenced by handle_in_arbitrate_event(), and handle_in_push_event().

Here is the caller graph for this function:



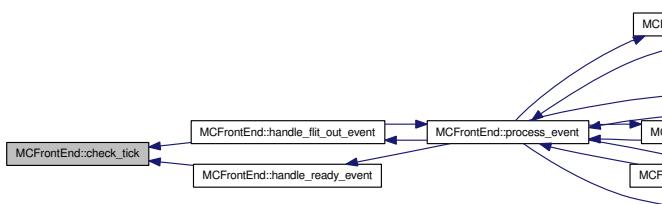
7.65.3.2 uint MCFrontEnd::check_tick () [private]

Definition at line 515 of file mcFrontEnd.cc.

References out_arbiter, OUT_ARBITRATE_EVENT, out_buffer, and out_packets.

Referenced by handle_flit_out_event(), and handle_ready_event().

Here is the caller graph for this function:



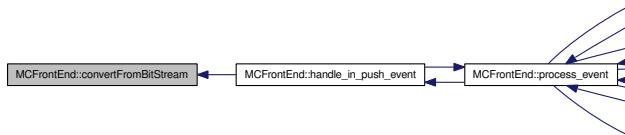
7.65.3.3 void MCFrontEnd::convertFromBitStream (Request * *req*, HighLevelPacket * *hlp*) [private]

Definition at line 355 of file mcFrontEnd.cc.

References Request::address, CACHE_WRITEBACK, Request::cmdType, Request::data, HighLevelPacket::data, NETWORK_ADDRESS_BITS, NETWORK_COMMAND_BITS, NETWORK_THREADID_BITS, Data::size, Request::threadId, Data::value, and WRITEBACK_SIZE.

Referenced by handle_in_push_event().

Here is the caller graph for this function:



7.65.3.4 uint MCFrontEnd::get_no_credits () const

Definition at line 88 of file mcFrontEnd.cc.

References out_buffer.

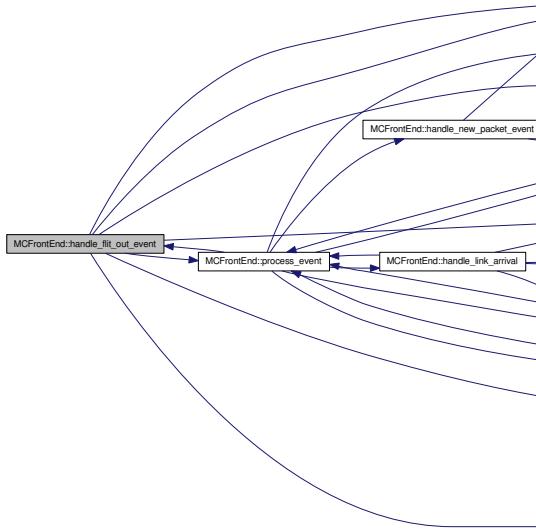
7.65.3.5 void MCFrontEnd::handle_flit_out_event (IrisEvent * *e*) [private]

Definition at line 313 of file mcFrontEnd.cc.

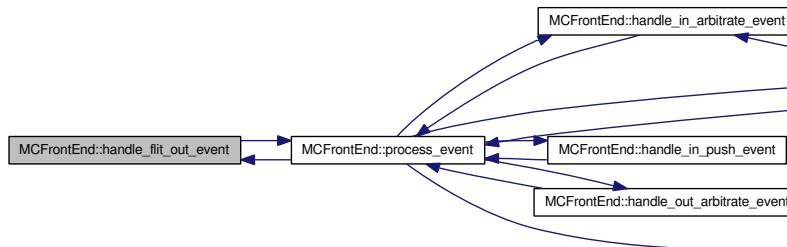
References NetworkComponent::address, check_tick(), FLIT_ID, flits_out, HEAD, last_out_arbitrate_cycle, LINK_ARRIVAL_EVENT, node_ip, Simulator::Now(), out_arbiter, Interface::output_connection, packets_out, GenericArbiter::pick_winner(), process_event(), NetworkComponent::process_event(), LinkArrivalData::ptr, Simulator::Schedule(), timed_cout(), IrisEvent::type, Flit::type, LinkArrivalData::type, LinkArrivalData::valid, IrisEvent::vc, and LinkArrivalData::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



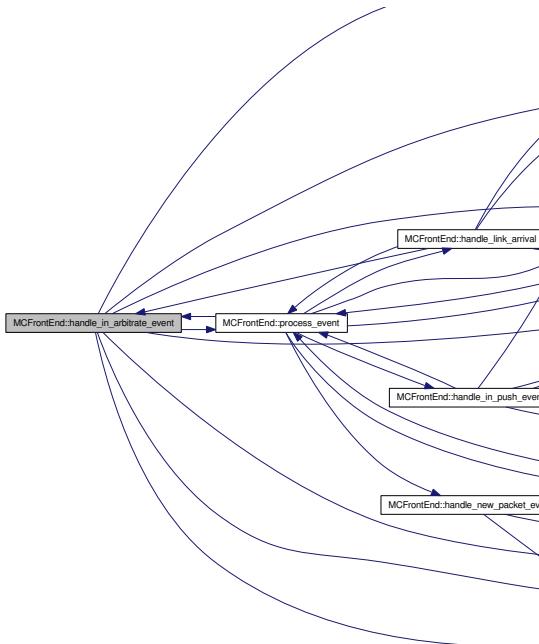
7.65.3.6 void MCFrontEnd::handle_in_arbitrate_event (IrisEvent * e) [private]

Definition at line 435 of file mcFrontEnd.cc.

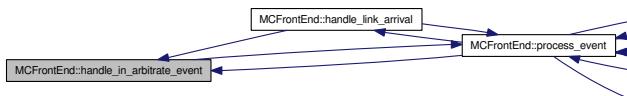
References NetworkComponent::address, check_input_conditions(), CREDIT_ID, HEAD, in_arbiter, in_buffer, in_packets, Interface::input_connection, last_in_arbitrate_cycle, LINK_ARRIVAL_EVENT, node_ip, Simulator::Now(), GenericArbiter::pick_winner(), process_event(), NetworkComponent::process_event(), GenericArbiter::request(), Simulator::Schedule(), timed_cout(), IrisEvent::type, Flit::type, LinkArrivalData::type, LinkArrivalData::valid, IrisEvent::vc, and LinkArrivalData::vc.

Referenced by handle_link_arrival(), and process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



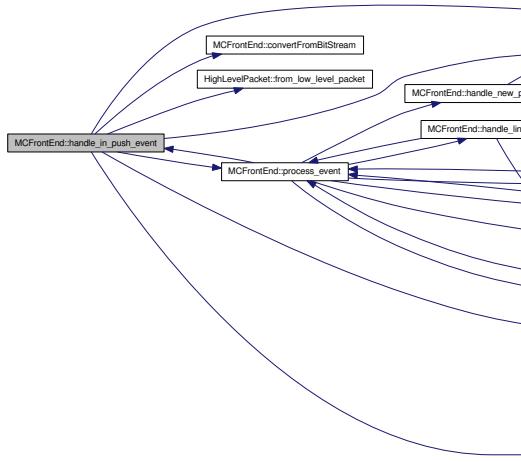
7.65.3.7 void MCFrontEnd::handle_in_push_event (IrisEvent * e) [private]

Definition at line 385 of file mcFrontEnd.cc.

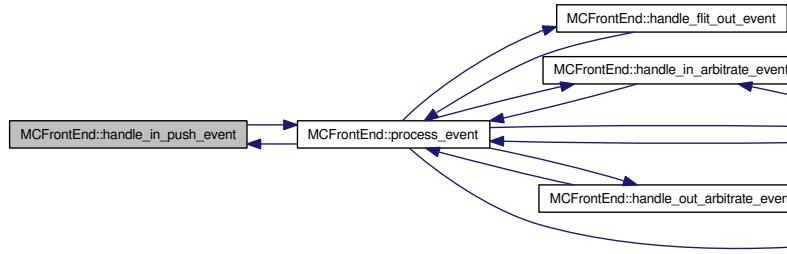
References `NetworkComponent::address`, `check_input_conditions()`, `convertFromBitStream()`, `HighLevelPacket::from_low_level_packet()`, `in_packets`, `in_ready`, `node_ip`, `Simulator::Now()`, `process_event()`, `Simulator::Schedule()`, `timed_cout()`, `IrisEvent::type`, and `IrisEvent::vc`.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



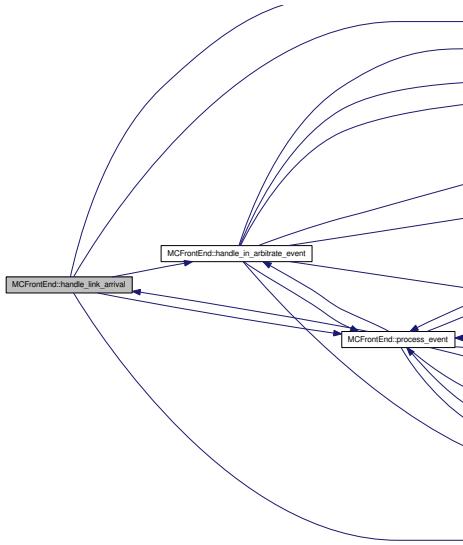
7.65.3.8 void MCFrontEnd::handle_link_arrival (IrisEvent * e) [private]

Definition at line 183 of file mcFrontEnd.cc.

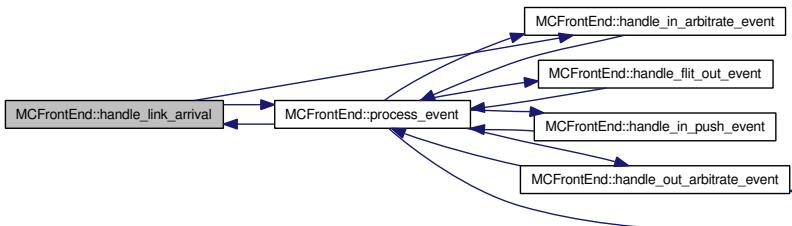
References NetworkComponent::address, CREDIT_ID, IrisEvent::event_data, FLIT_ID, flits_in, handle_in_arbitrate_event(), in_buffer, node_ip, Simulator::Now(), OUT_ARBITRATE_EVENT, out_buffer, packets_in, process_event(), LinkArrivalData::ptr, Simulator::Schedule(), TAIL, timed_cout(), Flit::type, LinkArrivalData::type, LinkArrivalData::valid, IrisEvent::vc, and LinkArrivalData::vc.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



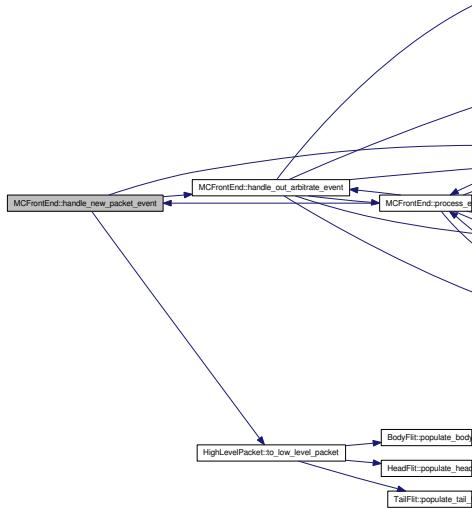
7.65.3.9 void MCFrontEnd::handle_new_packet_event (IrisEvent * e) [private]

Definition at line 234 of file mcFrontEnd.cc.

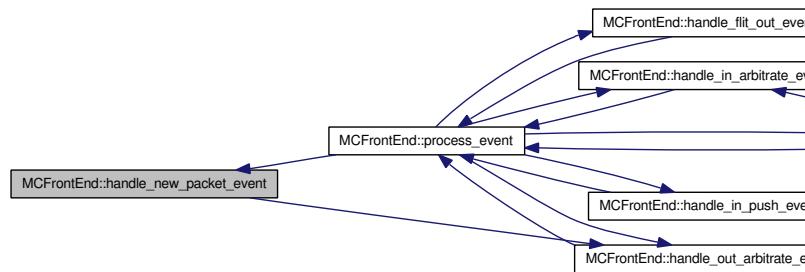
References NetworkComponent::address, IrisEvent::event_data, handle_out_arbitrate_event(), node_ip, out_packet_index, out_packets, timed_cout(), HighLevelPacket::to_low_level_packet(), and HighLevelPacket::virtual_channel.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



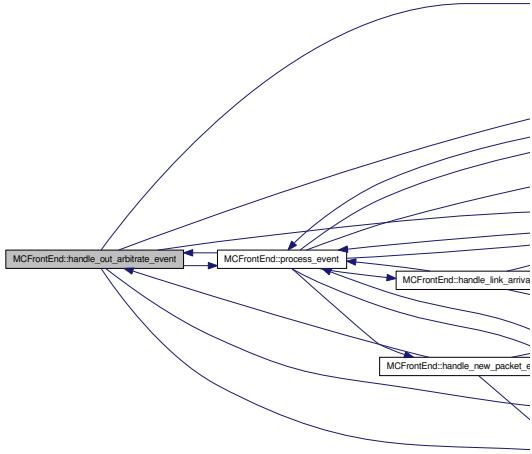
7.65.3.10 void MCFrontEnd::handle_out_arbitrate_event (IrisEvent * e)
[private]

Definition at line 252 of file mcFrontEnd.cc.

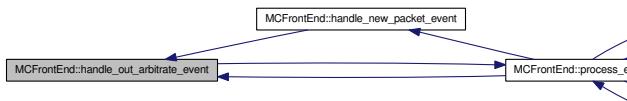
References NetworkComponent::address, FLIT_OUT_EVENT, last_flit_out_cycle, node_ip, Simulator::Now(), out_arbiter, out_buffer, out_packet_index, out_packets, process_event(), NetworkComponent::process_event(), Interface::processor_connection, READY_EVENT, GenericArbiter::request(), Simulator::Schedule(), timed_cout(), IrisEvent::type, Flit::type, IrisEvent::vc, and VirtualChannelDescription::vc.

Referenced by handle_new_packet_event(), and process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



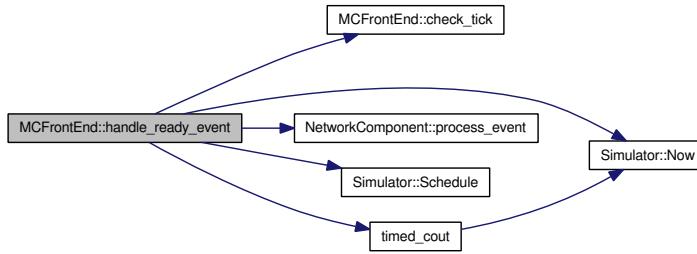
7.65.3.11 void MCFrontEnd::handle_ready_event (IrisEvent * e) [private]

Definition at line 157 of file mcFrontEnd.cc.

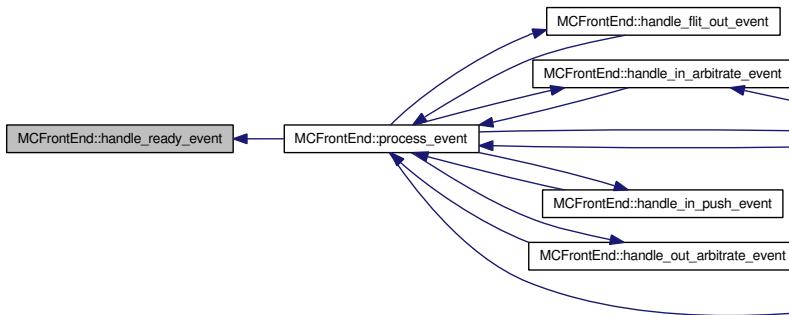
References NetworkComponent::address, check_tick(), IrisEvent::event_data, in_ready, node_ip, Simulator::Now(), NetworkComponent::process_event(), Simulator::Schedule(), timed_cout(), and IrisEvent::vc.

Referenced by `process_event()`.

Here is the call graph for this function:



Here is the caller graph for this function:



7.65.3.12 void MCFrontEnd::handle_tick_event (IrisEvent * e) [private]

7.65.3.13 string MCFrontEnd::print_stats () [virtual]

Implements **Interface** (p. 231).

Definition at line 542 of file mcFrontEnd.cc.

References flits_in, flits_out, packets_in, packets_out, and total_packets_in_time.

7.65.3.14 void MCFrontEnd::process_event (IrisEvent * e) [virtual]

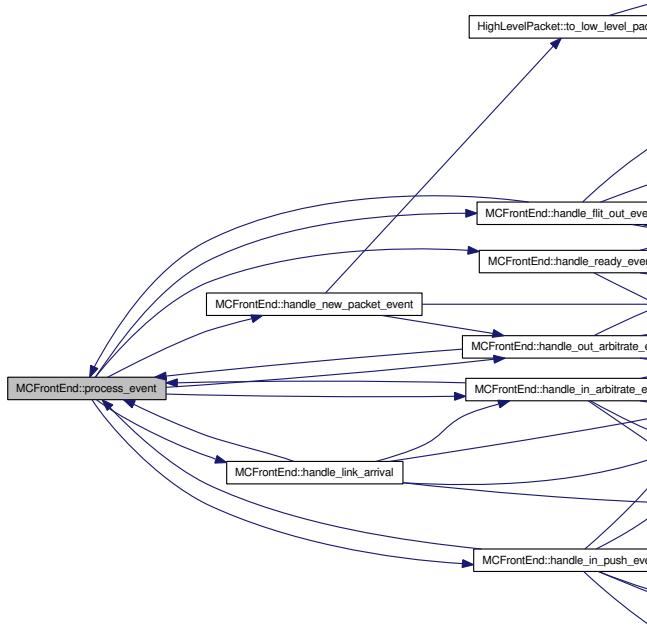
Implements **Interface** (p. 231).

Definition at line 117 of file mcFrontEnd.cc.

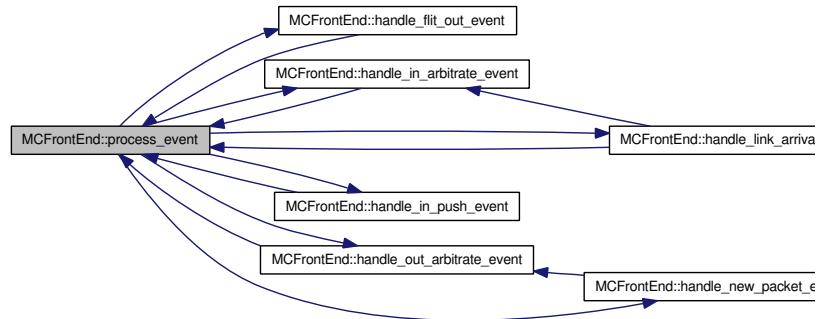
References FLIT_OUT_EVENT, handle_flt_out_event(), handle_in_arbitrate_event(), handle_in_push_event(), handle_link_arrival(), handle_new_packet_event(), handle_out_arbitrate_event(), handle_ready_event(), IN_ARBITRATE_EVENT, IN_PUSH_EVENT, LINK_ARRIVAL_EVENT, NEW_PACKET_EVENT, OUT_ARBITRATE_EVENT, READY_EVENT, and IrisEvent::type.

Referenced by handle_flt_out_event(), handle_in_arbitrate_event(), handle_in_push_event(), handle_link_arrival(), and handle_out_arbitrate_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.65.3.15 void MCFrontEnd::set_no_credits (uint credits)

Definition at line 94 of file mcFrontEnd.cc.

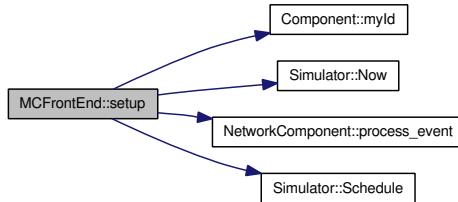
References `out_buffer`.

7.65.3.16 void MCFrontEnd::setup ()

Definition at line 35 of file mcFrontEnd.cc.

References NetworkComponent::address, flits_in, flits_out, in_arbiter, in_buffer, in_packets, in_ready, last_in_arbitrate_cycle, last_out_arbitrate_cycle, Component::myId(), Simulator::Now(), out_arbiter, out_buffer, out_packet_index, out_packets, packets_in, packets_out, NetworkComponent::process_event(), Interface::processor_connection, READY_EVENT, Simulator::Schedule(), total_packets_in_time, and VirtualChannelDescription::vc.

Here is the call graph for this function:



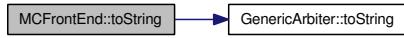
7.65.3.17 string MCFrontEnd::toString () const [virtual]

Reimplemented from **Interface** (p. 231).

Definition at line 101 of file mcFrontEnd.cc.

References NetworkComponent::address, in_arbiter, in_buffer, node_ip, out_arbiter, out_buffer, out_packets, and GenericArbiter::toString().

Here is the call graph for this function:



7.65.4 Member Data Documentation

7.65.4.1 uint MCFrontEnd::flits_in [private]

Definition at line 83 of file mcFrontEnd.h.

Referenced by handle_link_arrival(), print_stats(), and setup().

7.65.4.2 uint MCFrontEnd::flits_out [private]

Definition at line 85 of file mcFrontEnd.h.

Referenced by handle_flit_out_event(), print_stats(), and setup().

7.65.4.3 GenericArbiter MCFrontEnd::in_arbiter

Definition at line 53 of file mcFrontEnd.h.

Referenced by check_input_conditions(), handle_in_arbitrate_event(), setup(), and toString().

7.65.4.4 GenericOutputBuffer MCFrontEnd::in_buffer

Definition at line 54 of file mcFrontEnd.h.

Referenced by check_input_conditions(), handle_in_arbitrate_event(), handle_link_arrival(), setup(), and toString().

7.65.4.5 vector< LowLevelPacket> MCFrontEnd::in_packets [private]

Definition at line 65 of file mcFrontEnd.h.

Referenced by check_input_conditions(), handle_in_arbitrate_event(), handle_in_push_event(), and setup().

7.65.4.6 vector< bool > MCFrontEnd::in_ready [private]

Definition at line 66 of file mcFrontEnd.h.

Referenced by check_input_conditions(), handle_in_push_event(), handle_ready_event(), and setup().

7.65.4.7 unsigned long long int MCFrontEnd::last_flit_out_cycle [private]

Definition at line 59 of file mcFrontEnd.h.

Referenced by handle_out_arbitrate_event().

7.65.4.8 unsigned long long int MCFrontEnd::last_in_arbitrate_cycle [private]

Definition at line 58 of file mcFrontEnd.h.

Referenced by handle_in_arbitrate_event(), and setup().

7.65.4.9 unsigned long long int MCFrontEnd::last_out_arbitrate_cycle [private]

Definition at line 57 of file mcFrontEnd.h.

Referenced by handle_flit_out_event(), and setup().

7.65.4.10 uint MCFrontEnd::node_ip

Reimplemented from **NetworkComponent** (p. 276).

Definition at line 50 of file mcFrontEnd.h.

Referenced by handle_flit_out_event(), handle_in_arbitrate_event(), handle_in_push_event(), handle_link_arrival(), handle_new_packet_event(), handle_out_arbitrate_event(), handle_ready_event(), and toString().

7.65.4.11 GenericArbiter MCFrontEnd::out_arbiter

Definition at line 52 of file mcFrontEnd.h.

Referenced by check_tick(), handle_flit_out_event(), handle_out_arbitrate_event(), setup(), and toString().

7.65.4.12 GenericOutputBuffer MCFrontEnd::out_buffer

Definition at line 51 of file mcFrontEnd.h.

Referenced by check_tick(), get_no_credits(), handle_link_arrival(), handle_out_arbitrate_event(), set_no_credits(), setup(), and toString().

7.65.4.13 vector< uint > MCFrontEnd::out_packet_index [private]

Definition at line 62 of file mcFrontEnd.h.

Referenced by handle_new_packet_event(), handle_out_arbitrate_event(), and setup().

7.65.4.14 vector< LowLevelPacket> MCFrontEnd::out_packets [private]

Definition at line 61 of file mcFrontEnd.h.

Referenced by check_tick(), handle_new_packet_event(), handle_out_arbitrate_event(), setup(), and toString().

7.65.4.15 uint MCFrontEnd::packets_in [private]

Definition at line 84 of file mcFrontEnd.h.

Referenced by handle_link_arrival(), print_stats(), and setup().

7.65.4.16 uint MCFrontEnd::packets_out [private]

Definition at line 86 of file mcFrontEnd.h.

Referenced by handle_flit_out_event(), print_stats(), and setup().

7.65.4.17 bool MCFrontEnd::ticking [private]

Definition at line 78 of file mcFrontEnd.h.

7.65.4.18 uint MCFrontEnd::total_packets_in_time [private]

Definition at line 87 of file mcFrontEnd.h.

Referenced by print_stats(), and setup().

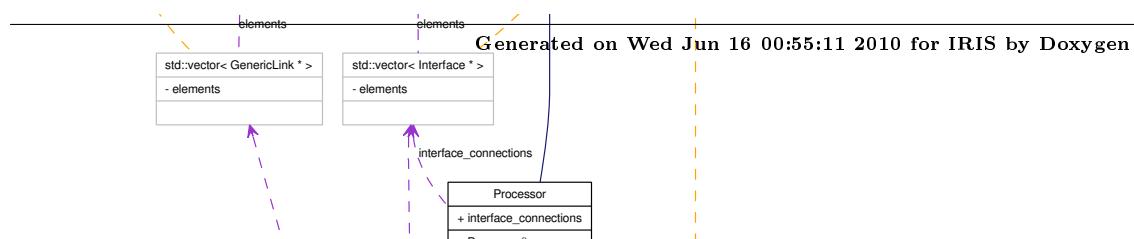
The documentation for this class was generated from the following files:

- **mcFrontEnd.h**
- **mcFrontEnd.cc**

7.66 Mesh Class Reference

```
#include <mesh.h>
```

Collaboration diagram for Mesh:



Public Member Functions

- `Mesh ()`
- `~Mesh ()`
- `void init (uint ports, uint vcs, uint credits, uint buffer_size, uint no_nodes, uint grid_size, uint links)`
- `void setup (void)`
- `void connect_interface_processor (void)`
- `void connect_interface_routers (void)`
- `void connect_routers (void)`
- `string print_stats (void)`
- `void set_max_phy_link_bits (uint a)`

Public Attributes

- `vector< Router * > routers`
- `vector< Interface * > interfaces`
- `vector< Processor * > processors`
- `vector< GenericLink * > link_a`
- `vector< GenericLink * > link_b`
- `unsigned long long int max_sim_time`

Private Attributes

- `uint ports`
- `uint vcs`
- `uint credits`
- `uint buffer_size`
- `uint no_nodes`
- `uint links`
- `uint grid_size`

7.66.1 Detailed Description

Definition at line 64 of file mesh.h.

7.66.2 Constructor & Destructor Documentation

7.66.2.1 Mesh::Mesh ()

Definition at line 23 of file mesh.cc.

7.66.2.2 Mesh::~Mesh ()

Definition at line 28 of file mesh.cc.

References interfaces, link_a, link_b, links, no_nodes, processors, and routers.

7.66.3 Member Function Documentation

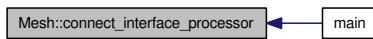
7.66.3.1 void Mesh::connect_interface_processor (void)

Definition at line 80 of file mesh.cc.

References interfaces, link_a, link_b, no_nodes, and processors.

Referenced by main().

Here is the caller graph for this function:



7.66.3.2 void Mesh::connect_interface_routers (void)

Definition at line 96 of file mesh.cc.

References interfaces, link_a, link_b, no_nodes, and routers.

Referenced by main().

Here is the caller graph for this function:



7.66.3.3 void Mesh::connect_routers (void)

Definition at line 112 of file mesh.cc.

References grid_size, link_a, link_b, links, no_nodes, and routers.

Referenced by main().

Here is the caller graph for this function:



7.66.3.4 void Mesh::init (uint ports, uint vcs, uint credits, uint buffer_size, uint no_nodes, uint grid_size, uint links)

Definition at line 46 of file mesh.cc.

References buffer_size, credits, grid_size, links, no_nodes, ports, and vcs.

Referenced by main().

Here is the caller graph for this function:



7.66.3.5 string Mesh::print_stats (void)

Definition at line 58 of file mesh.cc.

References interfaces, no_nodes, processors, and routers.

Referenced by main().

Here is the caller graph for this function:

**7.66.3.6 void Mesh::set_max_phy_link_bits (uint a)****7.66.3.7 void Mesh::setup (void)**

Definition at line 254 of file mesh.cc.

References buffer_size, credits, interfaces, link_a, link_b, links, max_sim_time, no_nodes, ports, processors, routers, and vcs.

Referenced by main().

Here is the caller graph for this function:

**7.66.4 Member Data Documentation****7.66.4.1 uint Mesh::buffer_size [private]**

Definition at line 90 of file mesh.h.

Referenced by init(), and setup().

7.66.4.2 uint Mesh::credits [private]

Definition at line 89 of file mesh.h.

Referenced by init(), and setup().

7.66.4.3 uint Mesh::grid_size [private]

Definition at line 93 of file mesh.h.

Referenced by connect_routers(), and init().

7.66.4.4 vector<Interface*> Mesh::interfaces

Definition at line 70 of file mesh.h.

Referenced by connect_interface_processor(), connect_interface_routers(), main(), print_stats(), setup(), and ~Mesh().

7.66.4.5 vector<GenericLink*> Mesh::link_a

Definition at line 72 of file mesh.h.

Referenced by connect_interface_processor(), connect_interface_routers(), connect_routers(), main(), setup(), and ~Mesh().

7.66.4.6 vector<GenericLink*> Mesh::link_b

Definition at line 73 of file mesh.h.

Referenced by connect_interface_processor(), connect_interface_routers(), connect_routers(), main(), setup(), and ~Mesh().

7.66.4.7 uint Mesh::links [private]

Definition at line 92 of file mesh.h.

Referenced by connect_routers(), init(), setup(), and ~Mesh().

7.66.4.8 unsigned long long int Mesh::max_sim_time

Definition at line 81 of file mesh.h.

Referenced by main(), and setup().

7.66.4.9 uint Mesh::no_nodes [private]

Definition at line 91 of file mesh.h.

Referenced by connect_interface_processor(), connect_interface_routers(), connect_routers(), init(), print_stats(), setup(), and ~Mesh().

7.66.4.10 uint Mesh::ports [private]

Definition at line 87 of file mesh.h.

Referenced by init(), and setup().

7.66.4.11 vector<Processor*> Mesh::processors

Definition at line 71 of file mesh.h.

Referenced by connect_interface_processor(), main(), print_stats(), setup(), and ~Mesh().

7.66.4.12 vector<Router*> Mesh::routers

Definition at line 69 of file mesh.h.

Referenced by connect_interface_routers(), connect_routers(), main(), print_stats(), setup(), and ~Mesh().

7.66.4.13 uint Mesh::vcs [private]

Definition at line 88 of file mesh.h.

Referenced by init(), and setup().

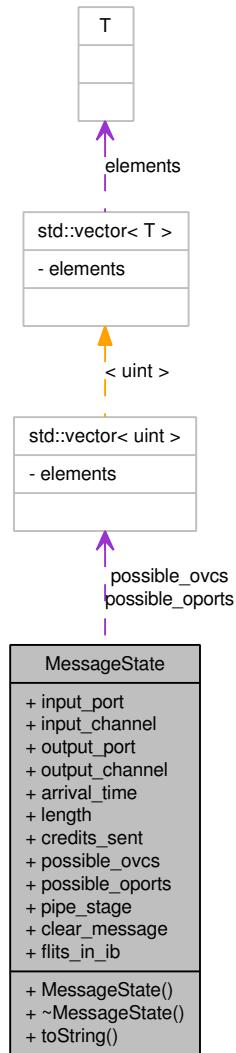
The documentation for this class was generated from the following files:

- **mesh.h**
- **mesh.cc**

7.67 MessageState Class Reference

```
#include <genericRouterNoVcs.h>
```

Collaboration diagram for MessageState:



Public Member Functions

- `MessageState ()`
- `~MessageState ()`
- string `toString () const`

Public Attributes

- `uint input_port`
- `uint input_channel`
- `uint output_port`

- `uint output_channel`
- `double arrival_time`
- `int length`
- `int credits_sent`
- `vector< uint > possible_ovcs`
- `vector< uint > possible_oports`
- `GenericRouterNoVcsPipeStage pipe_stage`
- `bool clear_message`
- `uint flits_in_ib`

7.67.1 Detailed Description

Definition at line 34 of file backup/genericRouterNoVcs.h.

7.67.2 Constructor & Destructor Documentation

7.67.2.1 MessageState::MessageState ()

Definition at line 42 of file backup/genericRouterNoVcs.cc.

References INVALID, and pipe_stage.

7.67.2.2 MessageState::~MessageState () [inline]

Definition at line 38 of file backup/genericRouterNoVcs.h.

7.67.3 Member Function Documentation

7.67.3.1 string MessageState::toString () const

Definition at line 539 of file backup/genericRouterNoVcs.cc.

References EMPTY, FULL, IB, input_channel, input_port, INVALID, output_channel, output_port, pipe_stage, REQ_OUTVC_ARB, ROUTED, ST, SW_ALLOCATED, and SWA_REQUESTED.

7.67.4 Member Data Documentation

7.67.4.1 double MessageState::arrival_time

Definition at line 43 of file backup/genericRouterNoVcs.h.

7.67.4.2 bool MessageState::clear_message

Definition at line 49 of file backup/genericRouterNoVcs.h.

7.67.4.3 int MessageState::credits_sent

Definition at line 45 of file backup/genericRouterNoVcs.h.

7.67.4.4 uint MessageState::flits_in_ib

Definition at line 50 of file backup/genericRouterNoVcs.h.

7.67.4.5 uint MessageState::input_channel

Definition at line 40 of file backup/genericRouterNoVcs.h.

Referenced by `toString()`.

7.67.4.6 uint MessageState::input_port

Definition at line 39 of file backup/genericRouterNoVcs.h.

Referenced by `toString()`.

7.67.4.7 int MessageState::length

Definition at line 44 of file backup/genericRouterNoVcs.h.

7.67.4.8 uint MessageState::output_channel

Definition at line 42 of file backup/genericRouterNoVcs.h.

Referenced by `toString()`.

7.67.4.9 uint MessageState::output_port

Definition at line 41 of file backup/genericRouterNoVcs.h.

Referenced by `toString()`.

7.67.4.10 GenericRouterNoVcsPipeStage MessageState::pipe_stage

Definition at line 48 of file backup/genericRouterNoVcs.h.

Referenced by `MessageState()`, and `toString()`.

7.67.4.11 vector< uint > MessageState::possible_oports

Definition at line 47 of file backup/genericRouterNoVcs.h.

7.67.4.12 vector< uint > MessageState::possible_ovcs

Definition at line 46 of file backup/genericRouterNoVcs.h.

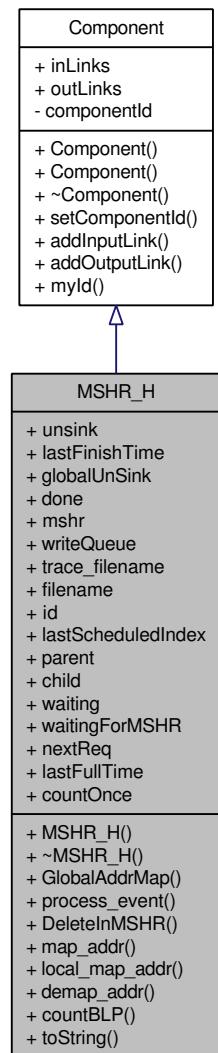
The documentation for this class was generated from the following files:

- [backup/genericRouterNoVcs.h](#)
- [backup/genericRouterNoVcs.cc](#)

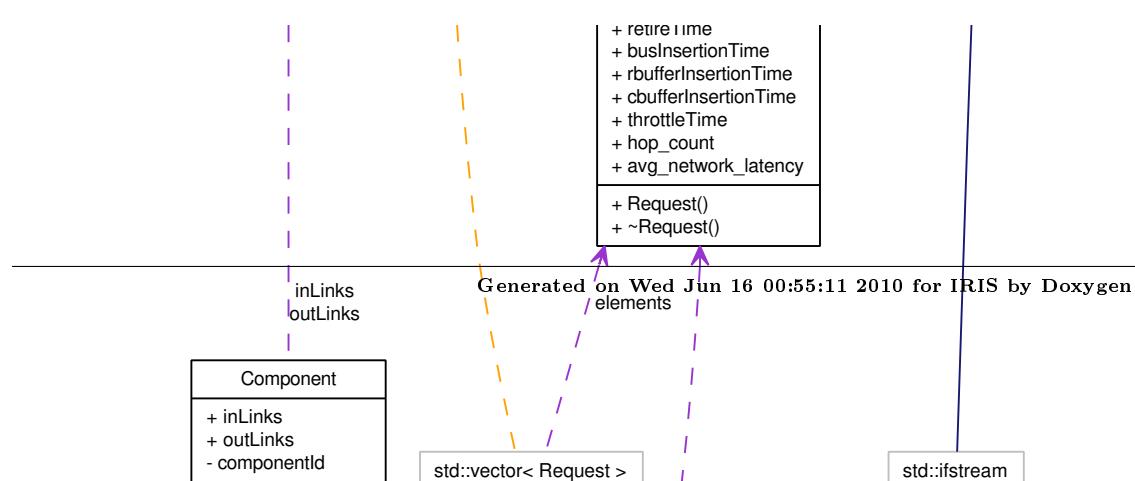
7.68 MSHR_H Class Reference

```
#include <mshr.h>
```

Inheritance diagram for MSHR_H:



Collaboration diagram for MSHR_H:



Public Member Functions

- **MSHR_H()**
- **~MSHR_H()**
- **Addr_t GlobalAddrMap (Addr_t addr, UInt threadId)**
- **void process_event (IrisEvent *e)**
- **void DeleteInMSHR (Request *req)**
- **short int map_addr (unsigned long long int *addr)**
- **void local_map_addr (Request *req)**
- **void demap_addr (Addr_t oldAddress, Addr_t newAddress)**
- **unsigned int countBLP (Request req)**
- **std::string toString ()**

Public Attributes

- **Time unsink**
- **Time lastFinishTime**
- **Time globalUnSink**
- **bool done**
- **vector< Request > mshr**
- **vector< Request > writeQueue**
- **ifstream trace_filename**
- **char * filename**
- **unsigned int id**
- **unsigned int lastScheduledIndex**
- **Component * parent**
- **Component * child**
- **bool waiting**
- **Request waitingForMSHR**
- **Request nextReq**
- **Time lastFullTime**
- **bool countOnce [8][NO_OF_BANKS]**

7.68.1 Detailed Description

Definition at line 50 of file mshr.h.

7.68.2 Constructor & Destructor Documentation

7.68.2.1 MSHR_H::MSHR_H()

Definition at line 34 of file mshr.cc.

References done, globalUnSink, lastFinishTime, lastFullTime, lastScheduledIndex, mshr, unsink, waiting, and writeQueue.

7.68.2.2 MSHR_H::~MSHR_H()

Definition at line 54 of file mshr.cc.

References mshr, and writeQueue.

7.68.3 Member Function Documentation

7.68.3.1 unsigned int MSHR_H::countBLP (Request *req*)

Definition at line 265 of file mshr.cc.

References countOnce, lastScheduledIndex, mshr, no_mcs, and NO_OF_BANKS.

Referenced by GenericTPG::GetNewRequest().

Here is the caller graph for this function:



7.68.3.2 void MSHR_H::DeleteInMSHR (Request * *req*)

Definition at line 240 of file mshr.cc.

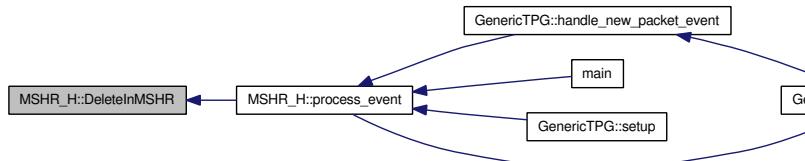
References Request::address, lastScheduledIndex, Request::mcNo, mshr, Simulator::Now(), parent, and Request::startTime.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.68.3.3 void MSHR_H::demap_addr (Addr_t *oldAddress*, Addr_t *newAddress*)

Definition at line 288 of file mshr.cc.

References mshr, and Simulator::Now().

Here is the call graph for this function:



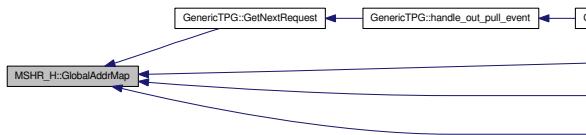
7.68.3.4 Addr_t MSHR_H::GlobalAddrMap (Addr_t *addr*, UInt *threadId*)

Definition at line 225 of file mshr.cc.

References NO_OF_THREADS, and THREAD_BITS_POSITION.

Referenced by GenericTPG::GetNextRequest(), main(), process_event(), and GenericTPG::setup().

Here is the caller graph for this function:



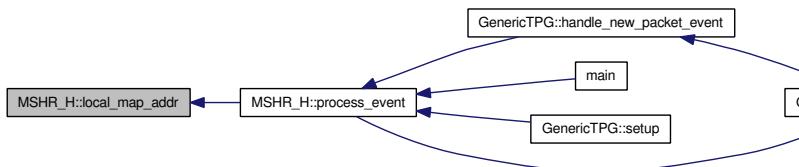
7.68.3.5 void MSHR_H::local_map_addr (Request * *req*)

Definition at line 330 of file mshr.cc.

References Request::address, Request::bankNo, BLOCKS_PER_ROW, CACHE_BLOCK_SIZE, Request::channelNo, COLUMN_SIZE, Request::columnNo, DRAM_SIZE, Request::lowerBits, NO_OF_BANKS, NO_OF_CHANNELS, NO_OF_COLUMNS, NO_OF_RANKS, NO_OF_ROWS, Request::rankNo, ROW_SIZE, Request::rowNo, and TAG_BITS.

Referenced by process_event().

Here is the caller graph for this function:



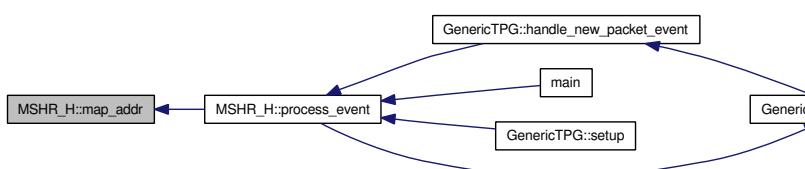
7.68.3.6 short int MSHR_H::map_addr (unsigned long long * *addr*)

Definition at line 307 of file mshr.cc.

References MC_ADDR_BITS, no_mcs, and NO_OF_THREADS.

Referenced by process_event().

Here is the caller graph for this function:



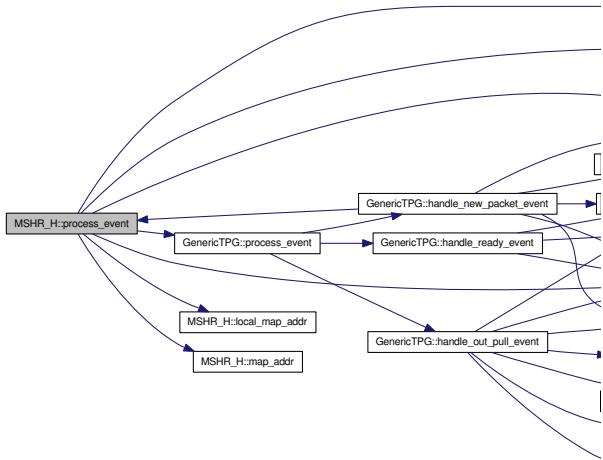
7.68.3.7 void MSHR_H::process_event (IrisEvent * e)

Definition at line 67 of file mshr.cc.

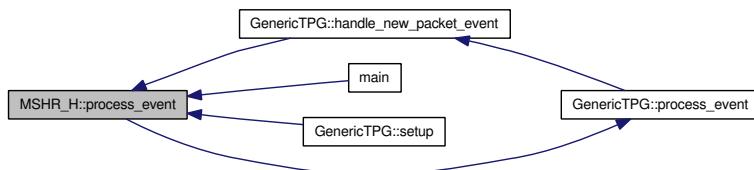
References Request::address, Request::arrivalTime, CACHE_WRITEBACK, Request::cmdType, DeleteInMSHR(), done, IrisEvent::dst, IrisEvent::event_data, filename, GlobalAddrMap(), globalUnSink, id, lastFinishTime, lastFullTime, local_map_addr(), map_addr(), max_sim_time, Request::mcNo, mshr, MSHR_DELETE, MSHR_SIZE, nextReq, Simulator::Now(), OUT_PULL_EVENT, parent, GenericTPG::process_event(), Simulator::Schedule(), IrisEvent::src, Request::startTime, Request::threadId, trace_filename, IrisEvent::type, unsink, waiting, waitingForMSHR, and writeQueue.

Referenced by GenericTPG::handle_new_packet_event(), main(), and GenericTPG::setup().

Here is the call graph for this function:



Here is the caller graph for this function:



7.68.3.8 std::string MSHR_H::toString ()

7.68.4 Member Data Documentation

7.68.4.1 Component* MSHR_H::child

Definition at line 66 of file mshr.h.

7.68.4.2 bool MSHR_H::countOnce[8][NO_OF_BANKS]

Definition at line 79 of file mshr.h.

Referenced by countBLP().

7.68.4.3 bool MSHR_H::done

Definition at line 58 of file mshr.h.

Referenced by main(), MSHR_H(), and process_event().

7.68.4.4 char* MSHR_H::filename

Definition at line 62 of file mshr.h.

Referenced by main(), process_event(), and GenericTPG::setup().

7.68.4.5 Time MSHR_H::globalUnSink

Definition at line 57 of file mshr.h.

Referenced by main(), MSHR_H(), and process_event().

7.68.4.6 unsigned int MSHR_H::id

Definition at line 63 of file mshr.h.

Referenced by main(), process_event(), and GenericTPG::setup().

7.68.4.7 Time MSHR_H::lastFinishTime

Definition at line 56 of file mshr.h.

Referenced by MSHR_H(), and process_event().

7.68.4.8 Time MSHR_H::lastFullTime

Definition at line 77 of file mshr.h.

Referenced by MSHR_H(), and process_event().

7.68.4.9 unsigned int MSHR_H::lastScheduledIndex

Definition at line 64 of file mshr.h.

Referenced by countBLP(), DeleteInMSHR(), GenericTPG::GetNewRequest(), GetNextRequest(), GenericTPG::handle_out_pull_event(), and MSHR_H().

7.68.4.10 vector<Request> MSHR_H::mshr

Definition at line 59 of file mshr.h.

Referenced by countBLP(), DeleteInMSHR(), demap_addr(), GenericTPG::GetNewRequest(), GetNextRequest(), GenericTPG::handle_out_pull_event(), MSHR_H(), process_event(), GenericTPG::~GenericTPG(), and ~MSHR_H().

7.68.4.11 Request MSHR_H::nextReq

Definition at line 76 of file mshr.h.

Referenced by process_event().

7.68.4.12 Component* MSHR_H::parent

Definition at line 65 of file mshr.h.

Referenced by DeleteInMSHR(), process_event(), and GenericTPG::setup().

7.68.4.13 ifstream MSHR_H::trace_filename

Definition at line 61 of file mshr.h.

Referenced by main(), process_event(), GenericTPG::setup(), and GenericTPG::~GenericTPG().

7.68.4.14 Time MSHR_H::unsink

Definition at line 55 of file mshr.h.

Referenced by main(), MSHR_H(), GenericTPG::print_stats(), and process_event().

7.68.4.15 bool MSHR_H::waiting

Definition at line 74 of file mshr.h.

Referenced by MSHR_H(), and process_event().

7.68.4.16 Request MSHR_H::waitForMSHR

Definition at line 75 of file mshr.h.

Referenced by process_event().

7.68.4.17 vector<Request> MSHR_H::writeQueue

Definition at line 60 of file mshr.h.

Referenced by GenericTPG::GetNewRequest(), GetNextRequest(), MSHR_H(), process_event(), and ~MSHR_H().

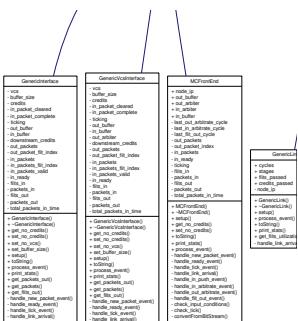
The documentation for this class was generated from the following files:

- **mshr.h**
- **mshr.cc**

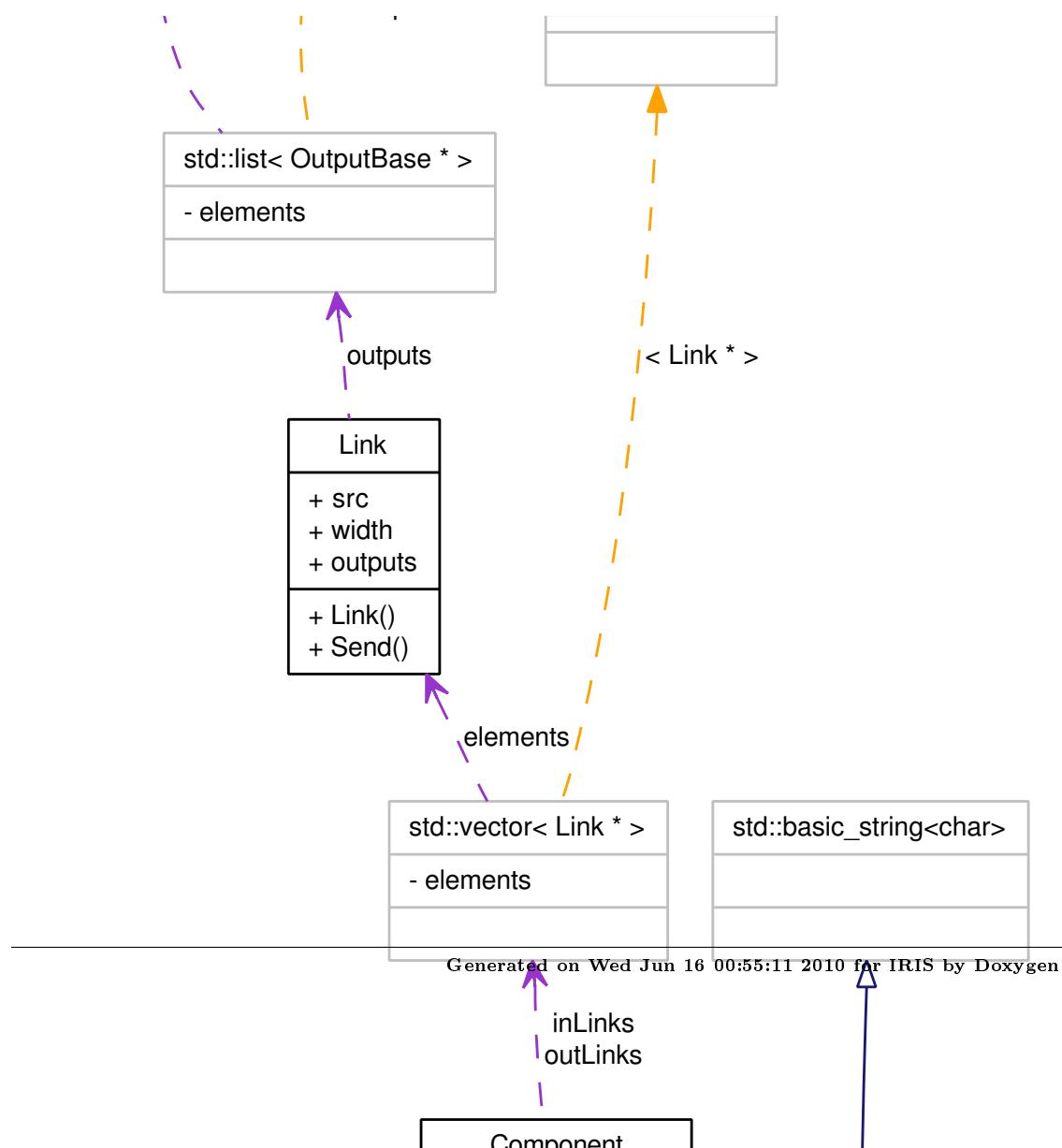
7.69 NetworkComponent Class Reference

```
#include <networkComponent.h>
```

Inheritance diagram for NetworkComponent:



Collaboration diagram for NetworkComponent:



Public Types

- enum types { processor, interface, link, router }

Public Member Functions

- NetworkComponent ()
- virtual ~NetworkComponent ()
- virtual string toString () const
- virtual void process_event (IrisEvent *e)=0

Public Attributes

- types type
- string name
- uniqueId address
- uint node_ip

7.69.1 Detailed Description

Definition at line 34 of file networkComponent.h.

7.69.2 Member Enumeration Documentation

7.69.2.1 enum NetworkComponent::types

Enumerator:

processor
interface
link
router

Definition at line 37 of file networkComponent.h.

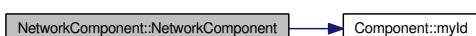
7.69.3 Constructor & Destructor Documentation

7.69.3.1 NetworkComponent::NetworkComponent ()

Definition at line 29 of file networkComponent.cc.

References address, and Component::myId().

Here is the call graph for this function:



7.69.3.2 NetworkComponent::~NetworkComponent () [virtual]

Definition at line 38 of file networkComponent.cc.

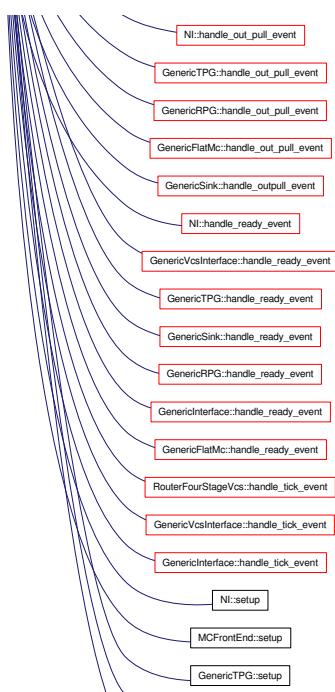
7.69.4 Member Function Documentation

7.69.4.1 virtual void NetworkComponent::process_event (IrisEvent * e)
 [pure virtual]

Implemented in **GenericRouterNoVcs** (p. 167), **GenericRPG** (p. 182), **MCFrontEnd** (p. 255), **RouterFourStageVcs** (p. 333), **GenericFlatMc** (p. 129), **GenericInterface** (p. 136), **GenericLink** (p. 142), **GenericRouterAdaptive** (p. 158), **GenericRouterNoVcs** (p. 167), **GenericRouterVct** (p. 174), **GenericRPG** (p. 182), **GenericSink** (p. 189), **GenericTPG** (p. 197), **GenericVcsInterface** (p. 210), **Interface** (p. 231), **IrisLink** (p. 236), **Processor** (p. 297), and **NI** (p. 280).

Referenced by `GenericRouterVct::do_switch_traversal()`, `GenericRouterNoVcs::do_switch_traversal()`, `GenericRouterAdaptive::do_switch_traversal()`, `MCFrontEnd::handle_flit_out_event()`, `MCFrontEnd::handle_in_arbitrate_event()`, `GenericVcsInterface::handle_link_arrival()`, `GenericInterface::handle_link_arrival()`, `RouterFourStageVcs::handle_link_arrival_event()`, `GenericRouterVct::handle_link_arrival_event()`, `GenericRouterNoVcs::handle_link_arrival_event()`, `GenericRouterAdaptive::handle_link_arrival_event()`, `GenericLink::handle_link_arrival_event()`, `NI::handle_new_packet_event()`, `GenericVcsInterface::handle_new_packet_event()`, `GenericTPG::handle_new_packet_event()`, `GenericSink::handle_new_packet_event()`, `GenericRPG::handle_new_packet_event()`, `GenericInterface::handle_new_packet_event()`, `GenericFlatMc::handle_new_packet_event()`, `MCFrontEnd::handle_out_arbitrate_event()`, `NI::handle_out_pull_event()`, `GenericTPG::handle_out_pull_event()`, `GenericRPG::handle_out_pull_event()`, `GenericFlatMc::handle_out_pull_event()`, `GenericSink::handle_outpull_event()`, `NI::handle_ready_event()`, `MCFrontEnd::handle_ready_event()`, `GenericVcsInterface::handle_ready_event()`, `GenericTPG::handle_ready_event()`, `GenericSink::handle_ready_event()`, `GenericRPG::handle_ready_event()`, `GenericInterface::handle_ready_event()`, `GenericFlatMc::handle_ready_event()`, `RouterFourStageVcs::handle_tick_event()`, `GenericVcsInterface::handle_tick_event()`, `GenericRouterVct::handle_tick_event()`, `GenericRouterNoVcs::handle_tick_event()`, `GenericRouterAdaptive::handle_tick_event()`, `GenericInterface::handle_tick_event()`, `GenericRouterVct::send_credit_back()`, `GenericRouterNoVcs::send_credit_back()`, `GenericRouterAdaptive::send_credit_back()`, `NI::setup()`, `MCFrontEnd::setup()`, `GenericTPG::setup()`, `GenericRPG::setup()`, and `GenericFlatMc::setup()`.

Here is the caller graph for this function:



7.69.4.2 string NetworkComponent::toString () const [virtual]

Reimplemented in **GenericRouterNoVcs** (p. 168), **GenericRPG** (p. 183), **MCFrontEnd** (p. 256), **RouterFourStageVcs** (p. 334), **GenericFlatMc** (p. 129), **GenericInterface** (p. 138), **GenericLink** (p. 143), **GenericRouterAdaptive** (p. 160), **GenericRouterNoVcs** (p. 168), **GenericRouterVct** (p. 175), **GenericRPG** (p. 183), **GenericSink** (p. 189), **GenericTPG** (p. 198), **GenericVcsInterface** (p. 211), **Interface** (p. 231), **IrisLink** (p. 237), **Processor** (p. 297), **Router** (p. 329), and **NI** (p. 281).

Definition at line 43 of file networkComponent.cc.

References address, and type.

Referenced by **Interface::toString()**.

Here is the caller graph for this function:



7.69.5 Member Data Documentation

7.69.5.1 uniqueId NetworkComponent::address

Reimplemented in **GenericRPG** (p. 183), and **GenericSink** (p. 189).

Definition at line 40 of file networkComponent.h.

Referenced by **GenericRouterVct::do_switch_traversal()**, **GenericRouterNoVcs::do_switch_traversal()**, **GenericRouterAdaptive::do_switch_traversal()**, **MCFrontEnd::handle_flit_out_event()**, **MCFrontEnd::handle_in_arbitrate_event()**, **MCFrontEnd::handle_in_push_event()**, **MCFrontEnd::handle_link_arrival()**, **GenericVcsInterface::handle_link_arrival()**, **GenericInterface::handle_link_arrival()**, **GenericLink::handle_link_arrival_event()**, **MCFrontEnd::handle_new_packet_event()**, **MCFrontEnd::handle_out_arbitrate_event()**, **NI::handle_out_pull_event()**, **GenericTPG::handle_out_pull_event()**, **GenericFlatMc::handle_out_pull_event()**, **MCFrontEnd::handle_ready_event()**, **RouterFourStageVcs::handle_tick_event()**, **GenericVcsInterface::handle_tick_event()**, **GenericRouterNoVcs::handle_tick_event()**, **GenericInterface::handle_tick_event()**, **RouterFourStageVcs::init()**, **GenericRouterVct::init()**, **GenericRouterNoVcs::init()**, **GenericRouterAdaptive::init()**, **Interface::Interface()**, **NetworkComponent()**, **NI::print_stats()**, **GenericTPG::print_stats()**, **GenericLink::print_stats()**, **GenericTPG::process_event()**, **GenericFlatMc::process_event()**, **GenericRouterVct::send_credit_back()**, **GenericRouterNoVcs::send_credit_back()**, **GenericRouterAdaptive::send_credit_back()**, **GenericFlatMc::set_output_path()**, **NI::setup()**, **MCFrontEnd::setup()**, **GenericVcsInterface::setup()**, **GenericTPG::setup()**, **GenericLink::setup()**, **GenericInterface::setup()**, **GenericFlatMc::setup()**, **RouterFourStageVcs::toString()**, **NI::toString()**, **toString()**, **MCFrontEnd::toString()**, **Interface::toString()**, **GenericVcsInterface::toString()**, **GenericTPG::toString()**, **GenericRouterVct::toString()**, **GenericRouterNoVcs::toString()**, **GenericRouterAdaptive::toString()**, **GenericLink::toString()**, **GenericInterface::toString()**, and **GenericFlatMc::toString()**.

7.69.5.2 string NetworkComponent::name

Definition at line 39 of file networkComponent.h.

Referenced by **GenericFlatMc::GenericFlatMc()**, **GenericInterface::GenericInterface()**, **GenericRouterAdaptive::GenericRouterAdaptive()**, **GenericRouterNoVcs::GenericRouterNoVcs()**, **GenericRouterVct::GenericRouterVct()**, **Generi-**

cRPG::GenericRPG(), GenericSink::GenericSink(), GenericTPG::GenericTPG(), GenericVcsInterface::GenericVcsInterface(), MCFrontEnd::MCFrontEnd(), NI::NI(), RouterFourStageVcs::RouterFourStageVcs(), and GenericLink::setup().

7.69.5.3 uint NetworkComponent::node_ip

Reimplemented in **MCFrontEnd** (p. 257), **GenericLink** (p. 143), and **NI** (p. 282).

Definition at line 41 of file networkComponent.h.

Referenced by GenericTPG::GetNextRequest(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericSink::handle_outpull_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), RouterFourStageVcs::init(), GenericRouterVct::init(), GenericRouterNoVcs::init(), GenericRouterAdaptive::init(), RouterFourStageVcs::print_stats(), GenericVcsInterface::print_stats(), GenericTPG::print_stats(), GenericRouterVct::print_stats(), GenericRouterNoVcs::print_stats(), GenericRouterAdaptive::print_stats(), GenericInterface::print_stats(), GenericFlatMc::print_stats(), GenericTPG::set_output_path(), GenericTPG::setup(), RouterFourStageVcs::toString(), GenericVcsInterface::toString(), GenericTPG::toString(), GenericRouterVct::toString(), GenericRouterNoVcs::toString(), GenericRouterAdaptive::toString(), GenericInterface::toString(), and GenericFlatMc::toString().

7.69.5.4 types NetworkComponent::type

Definition at line 38 of file networkComponent.h.

Referenced by Interface::Interface(), Processor::Processor(), Router::Router(), and toString().

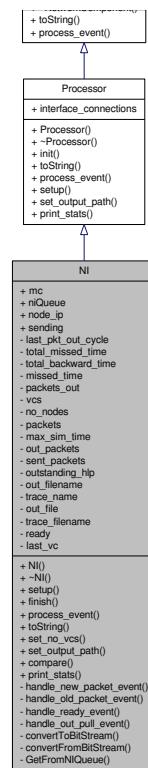
The documentation for this class was generated from the following files:

- **networkComponent.h**
- **networkComponent.cc**

7.70 NI Class Reference

```
#include <NI.h>
```

Inheritance diagram for NI:



Collaboration diagram for NI:

Public Member Functions

- `NI()`
- `~NI()`
- `void setup (uint no_nodes, uint vcs, uint max_sim_time)`
- `void finish ()`
- `void process_event (IrisEvent *e)`
- `string toString () const`
- `void set_no_vcs (uint v)`
- `void set_output_path (string v)`
- `bool compare ()`
- `string print_stats () const`

Public Attributes

- `Component * mc`
- `vector< Request > niQueue`
- `uint node_ip`
- `bool sending`

Private Member Functions

- `void handle_new_packet_event (IrisEvent *e)`
- `void handle_old_packet_event (IrisEvent *e)`
- `void handle_ready_event (IrisEvent *e)`
- `void handle_out_pull_event (IrisEvent *e)`
- `void convertToBitStream (Request *req, HighLevelPacket *hlp)`
- `void convertFromBitStream (Request *req, HighLevelPacket *hlp)`
- `bool GetFromNIQueue (Request *req)`

Private Attributes

- `double last_pkt_out_cycle`
- `unsigned long long int total_missed_time`
- `unsigned long long int total_backward_time`
- `uint missed_time`
- `unsigned long long int packets_out`
- `uint vcs`
- `uint no_nodes`
- `unsigned int packets`
- `unsigned long long int max_sim_time`
- `deque< HighLevelPacket > out_packets`
- `deque< HighLevelPacket > sent_packets`
- `HighLevelPacket * outstanding_hlp`
- `string out_filename`
- `string trace_name`
- `ofstream out_file`
- `fstream trace_filename`
- `vector< bool > ready`
- `unsigned int last_vc`

7.70.1 Detailed Description

Definition at line 43 of file NI.h.

7.70.2 Constructor & Destructor Documentation

7.70.2.1 NI::NI ()

Definition at line 27 of file NI.cc.

References Processor::interface_connections, mc, and NetworkComponent::name.

7.70.2.2 NI::~NI ()

Definition at line 37 of file NI.cc.

7.70.3 Member Function Documentation

7.70.3.1 bool NI::compare ()

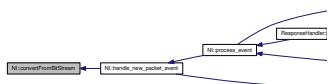
7.70.3.2 void NI::convertFromBitStream (Request * *req*, HighLevelPacket * *hlp*) [private]

Definition at line 385 of file NI.cc.

References Request::address, Request::cmdType, HighLevelPacket::data, NETWORK_ADDRESS_BITS, NETWORK_COMMAND_BITS, NETWORK_THREADID_BITS, and Request::threadId.

Referenced by handle_new_packet_event().

Here is the caller graph for this function:



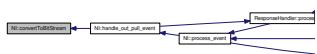
7.70.3.3 void NI::convertToBitStream (Request * *req*, HighLevelPacket * *hlp*) [private]

Definition at line 337 of file NI.cc.

References Request::address, CACHE_BLOCK_SIZE, CACHE_PREFETCH, CACHE_READ, CACHE_WRITE, Request::cmdType, HighLevelPacket::data, HighLevelPacket::data_payload_length, max_phy_link_bits, and NETWORK_ADDRESS_BITS.

Referenced by handle_out_pull_event().

Here is the caller graph for this function:



7.70.3.4 void NI::finish ()

Definition at line 102 of file NI.cc.

References out_file.

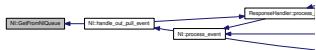
7.70.3.5 bool NI::GetFromNIQueue (Request * *req*) [private]

Definition at line 323 of file NI.cc.

References niQueue.

Referenced by handle_out_pull_event().

Here is the caller graph for this function:



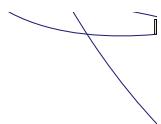
7.70.3.6 void NI::handle_new_packet_event (IrisEvent * *e*) [private]

Definition at line 130 of file NI.cc.

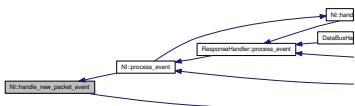
References _DBG, _DBG_NOARG, Request::address, Request::arrivalTime, HighLevelPacket::avg_network_latency, Request::avg_network_latency, CACHE_WRITEBACK, Request::cmdType, convertFromBitStream(), IrisEvent::dst, IrisEvent::event_data, HighLevelPacket::hop_count, Request::hop_count, Processor::interface_connections, last_pkt_out_cycle, mc, Simulator::Now(), outstanding_hlp, packets, NetworkComponent::process_event(), RequestHandler::process_event(), READY_EVENT, HighLevelPacket::recv_time, HighLevelPacket::req_start_time, Simulator::Schedule(), IrisEvent::src, IrisEvent::src_id, START, Request::startTime, Request::threadId, total_missed_time, and IrisEvent::type.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.70.3.7 void NI::handle_old_packet_event (IrisEvent * e) [private]

7.70.3.8 void NI::handle_out_pull_event (IrisEvent * e) [private]

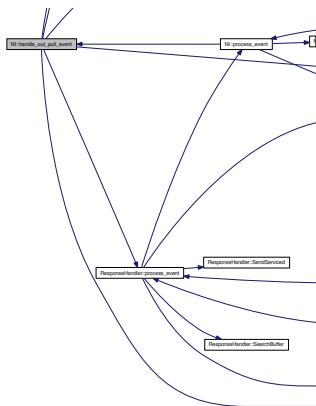
Definition at line 186 of file NI.cc.

References _DBG, HighLevelPacket::addr, Request::address, NetworkComponent::address, Request::arrivalTime, Request::avg_network_latency, HighLevelPacket::avg_network_latency, convertToBitStream(), HighLevelPacket::destination, IrisEvent::dst, GetFromNIQueue(), Request::hop_count, HighLevelPacket::hop_count, Processor::interface_connections, mc, HighLevelPacket::msg_class, NEW_PACKET_EVENT, niQueue, Simulator::Now(), packets_out, ResponseHandler::process_event(), NetworkComponent::process_event(), ready, HighLevelPacket::recv_time,

HighLevelPacket::req_start_time, RESPONSE_PKT, Request::retireTime, Simulator::Schedule(), SEND_TO_NI, sending, HighLevelPacket::sent_time, HighLevelPacket::source, Request::startTime, HighLevelPacket::stat_memory_serviced_time, Request::threadId, total_backward_time, HighLevelPacket::transaction_id, IrisEvent::type, HighLevelPacket::virtual_channel, and HighLevelPacket::waiting_in_ni.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



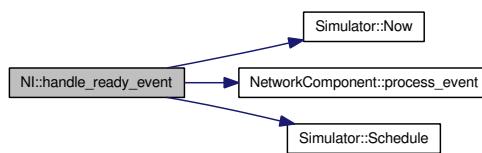
7.70.3.9 void NI::handle_ready_event (IrisEvent * e) [private]

Definition at line 253 of file NI.cc.

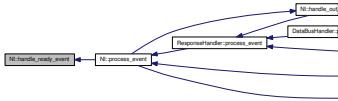
References max_sim_time, Simulator::Now(), OUT_PULL_EVENT, NetworkComponent::process_event(), ready, Simulator::Schedule(), and sending.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.70.3.10 string NI::print_stats (void) const [virtual]

Implements **Processor** (p. 296).

Definition at line 414 of file NI.cc.

References NetworkComponent::address, last_pkt_out_cycle, mc, node_ip, packets, packets_out, total_backward_time, and total_missed_time.

7.70.3.11 void NI::process_event (IrisEvent * e) [virtual]

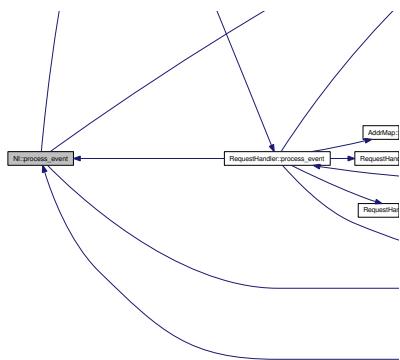
Implements **Processor** (p. 297).

Definition at line 109 of file NI.cc.

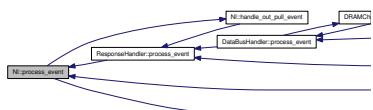
References handle_new_packet_event(), handle_out_pull_event(), handle_ready_event(), NEW_PACKET_EVENT, OUT_PULL_EVENT, READY_EVENT, and IrisEvent::type.

Referenced by ResponseHandler::process_event(), and RequestHandler::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.70.3.12 void NI::set_no_vcs (uint v)

Definition at line 42 of file NI.cc.

References vcs.

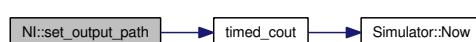
7.70.3.13 void NI::set_output_path(string *v*) [virtual]

Implements **Processor** (p. 297).

Definition at line 86 of file NI.cc.

References node ip, out file, out filename, and timed cout().

Here is the call graph for this function:



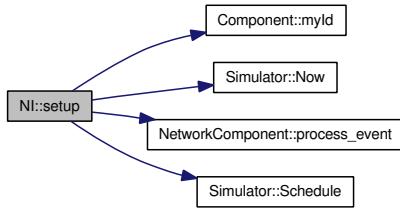
7.70.3.14 void NI::setup (uint no_nodes, uint vcs, uint max_sim_time) [virtual]

Implements **Processor** (p. 297).

Definition at line 48 of file NI.cc.

References NetworkComponent::address, max_sim_time, mc, missed_time, Component::myId(), no_nodes, node_ip, Simulator::Now(), packets, packets_out, NetworkComponent::process_event(), ready, READY_EVENT, Simulator::Schedule(), total_backward_time, total_missed_time, VirtualChannelDescription::vc, and vcs.

Here is the call graph for this function:



7.70.3.15 string NI::toString (void) const [virtual]

Reimplemented from **Processor** (p. 297).

Definition at line 313 of file NI.cc.

References NetworkComponent::address, and out_filename.

7.70.4 Member Data Documentation

7.70.4.1 double NI::last_pkt_out_cycle [private]

Definition at line 64 of file NI.h.

Referenced by handle_new_packet_event(), and print_stats().

7.70.4.2 unsigned int NI::last_vc [private]

Definition at line 82 of file NI.h.

7.70.4.3 unsigned long long int NI::max_sim_time [private]

Definition at line 73 of file NI.h.

Referenced by handle_ready_event(), and setup().

7.70.4.4 Component* NI::mc

Definition at line 48 of file NI.h.

Referenced by handle_new_packet_event(), handle_out_pull_event(), NI(), print_stats(), and setup().

7.70.4.5 uint NI::missed_time [private]

Definition at line 67 of file NI.h.

Referenced by setup().

7.70.4.6 vector<Request> NI::niQueue

Definition at line 49 of file NI.h.

Referenced by GetFromNIQueue(), and handle_out_pull_event().

7.70.4.7 uint NI::no_nodes [private]

Definition at line 71 of file NI.h.

Referenced by setup().

7.70.4.8 uint NI::node_ip

Reimplemented from **NetworkComponent** (p. 276).

Definition at line 54 of file NI.h.

Referenced by print_stats(), set_output_path(), and setup().

7.70.4.9 ofstream NI::out_file [private]

Definition at line 79 of file NI.h.

Referenced by finish(), and set_output_path().

7.70.4.10 string NI::out_filename [private]

Definition at line 77 of file NI.h.

Referenced by set_output_path(), and toString().

7.70.4.11 deque< HighLevelPacket > NI::out_packets [private]

Definition at line 74 of file NI.h.

7.70.4.12 HighLevelPacket* NI::outstanding_hlp [private]

Definition at line 76 of file NI.h.

Referenced by handle_new_packet_event().

7.70.4.13 unsigned int NI::packets [private]

Definition at line 72 of file NI.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.70.4.14 unsigned long long int NI::packets_out [private]

Definition at line 68 of file NI.h.

Referenced by handle_out_pull_event(), print_stats(), and setup().

7.70.4.15 vector< bool > NI::ready [private]

Definition at line 81 of file NI.h.

Referenced by handle_out_pull_event(), handle_ready_event(), and setup().

7.70.4.16 bool NI::sending

Definition at line 59 of file NI.h.

Referenced by handle_out_pull_event(), and handle_ready_event().

7.70.4.17 deque< HighLevelPacket > NI::sent_packets [private]

Definition at line 75 of file NI.h.

7.70.4.18 unsigned long long int NI::total_backward_time [private]

Definition at line 66 of file NI.h.

Referenced by handle_out_pull_event(), print_stats(), and setup().

7.70.4.19 unsigned long long int NI::total_missed_time [private]

Definition at line 65 of file NI.h.

Referenced by handle_new_packet_event(), print_stats(), and setup().

7.70.4.20 fstream NI::trace_filename [private]

Definition at line 80 of file NI.h.

7.70.4.21 string NI::trace_name [private]

Definition at line 78 of file NI.h.

7.70.4.22 uint NI::vcs [private]

Definition at line 70 of file NI.h.

Referenced by set_no_vcs(), and setup().

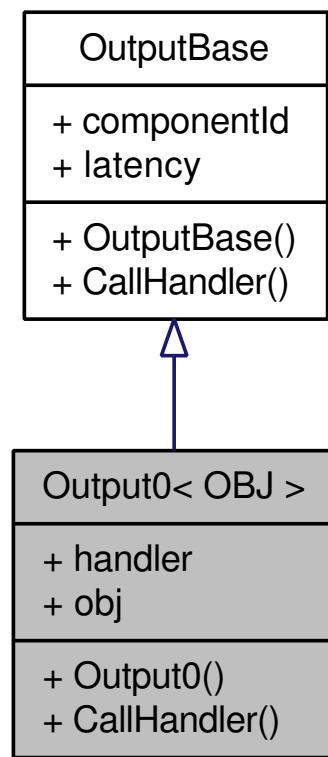
The documentation for this class was generated from the following files:

- **NI.h**
- **NI.cc**

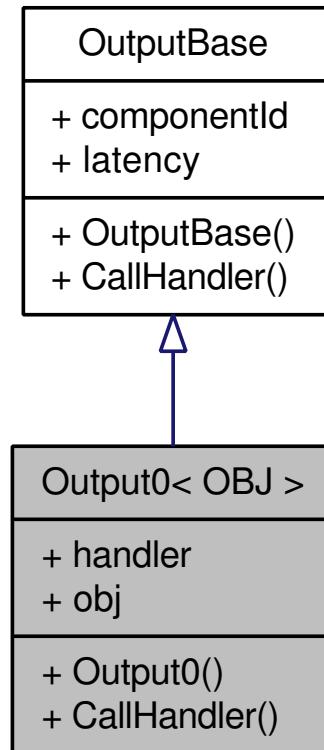
7.71 Output0< OBJ > Class Template Reference

```
#include <link.h>
```

Inheritance diagram for Output0< OBJ >:



Collaboration diagram for Output0< OBJ >:



Public Member Functions

- **Output0** (int **componentId**, double **l**, void(OBJ::***f**)(uint64_t, int), OBJ ***obj0**)
- void **CallHandler** (uint64_t **data**, int **src**)

Public Attributes

- void(OBJ::*** handler**)(uint64_t, int)
- OBJ * **obj**

7.71.1 Detailed Description

`template<typename OBJ> class Output0< OBJ >`

Definition at line 21 of file link.h.

7.71.2 Constructor & Destructor Documentation

7.71.2.1 template<typename OBJ> Output0< OBJ >::Output0 (int componentId, double l, void(OBJ::***)(uint64_t, int) **f**, OBJ * **obj0**) [inline]**

Definition at line 24 of file link.h.

7.71.3 Member Function Documentation

**7.71.3.1 template<typename OBJ> void Output0< OBJ >::CallHandler
(uint64_t *data*, int *src*) [inline, virtual]**

Implements **OutputBase** (p. 287).

Definition at line 28 of file link.h.

References `Output0< OBJ >::handler`, and `Output0< OBJ >::obj`.

7.71.4 Member Data Documentation

**7.71.4.1 template<typename OBJ> void(OBJ::* Output0< OBJ
>::handler)(uint64_t, int)**

Referenced by `Output0< OBJ >::CallHandler()`.

7.71.4.2 template<typename OBJ> OBJ* Output0< OBJ >::obj

Definition at line 27 of file link.h.

Referenced by `Output0< OBJ >::CallHandler()`.

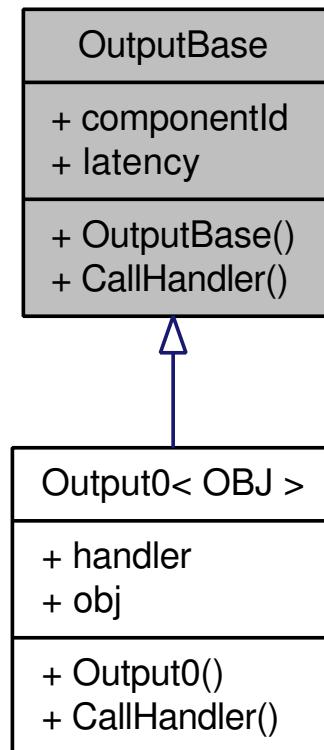
The documentation for this class was generated from the following file:

– **link.h**

7.72 OutputBase Class Reference

```
#include <link.h>
```

Inheritance diagram for OutputBase:



Public Member Functions

- **OutputBase** (int ID, double l)
- virtual void **CallHandler** (uint64_t, int)=0

Public Attributes

- int **componentId**
- double **latency**

7.72.1 Detailed Description

Definition at line 11 of file link.h.

7.72.2 Constructor & Destructor Documentation

7.72.2.1 OutputBase::OutputBase (int *ID*, double *l*) [inline]

Definition at line 14 of file link.h.

7.72.3 Member Function Documentation

7.72.3.1 virtual void OutputBase::CallHandler (uint64_t, int) [pure virtual]

Implemented in **Output0< OBJ >** (p. 285).

Referenced by **Link::Send()**.

Here is the caller graph for this function:



7.72.4 Member Data Documentation

7.72.4.1 int OutputBase::componentId

Definition at line 16 of file **link.h**.

7.72.4.2 double OutputBase::latency

Definition at line 17 of file **link.h**.

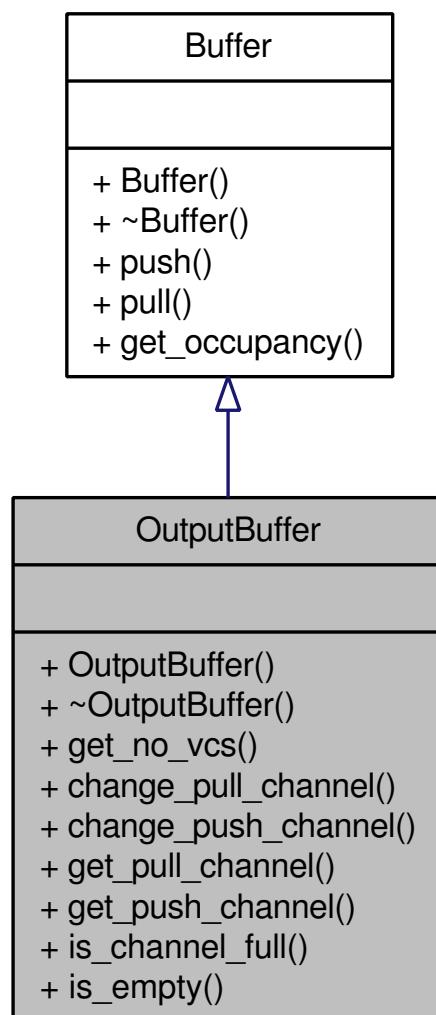
The documentation for this class was generated from the following file:

- **link.h**

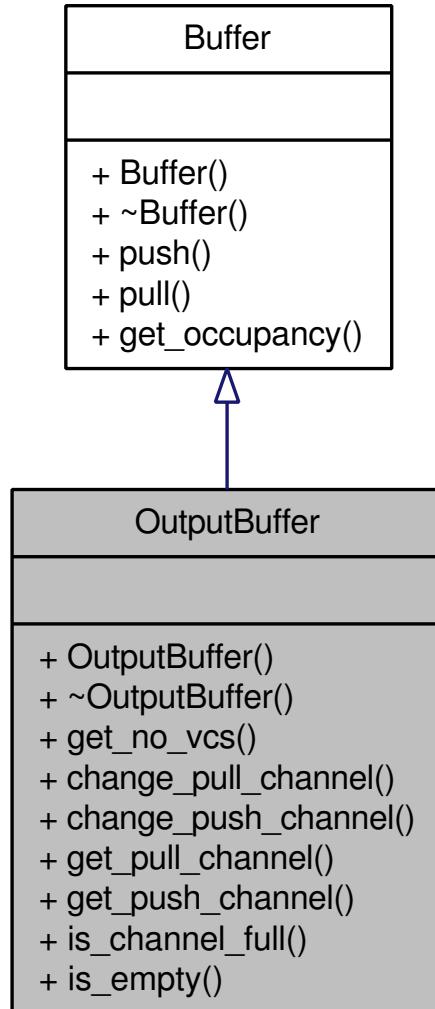
7.73 OutputBuffer Class Reference

```
#include <outputBuffer.h>
```

Inheritance diagram for OutputBuffer:



Collaboration diagram for OutputBuffer:



Public Member Functions

- **OutputBuffer ()**
- virtual **~OutputBuffer ()**
- virtual **uint get_no_vcs () const =0**
- virtual **void change_pull_channel (uint channel)=0**
- virtual **void change_push_channel (uint channel)=0**
- virtual **uint get_pull_channel () const =0**
- virtual **uint get_push_channel () const =0**
- virtual **bool is_channel_full (uint channel) const =0**
- virtual **bool is_empty (uint channel) const =0**

7.73.1 Detailed Description

Definition at line 31 of file outputBuffer.h.

7.73.2 Constructor & Destructor Documentation

7.73.2.1 OutputBuffer::OutputBuffer () [inline]

Definition at line 34 of file outputBuffer.h.

7.73.2.2 virtual OutputBuffer::~OutputBuffer () [inline, virtual]

Definition at line 35 of file outputBuffer.h.

7.73.3 Member Function Documentation

7.73.3.1 virtual void OutputBuffer::change_pull_channel (uint *channel*) [pure virtual]

7.73.3.2 virtual void OutputBuffer::change_push_channel (uint *channel*) [pure virtual]

7.73.3.3 virtual uint OutputBuffer::get_no_vcs () const [pure virtual]

7.73.3.4 virtual uint OutputBuffer::get_pull_channel () const [pure virtual]

7.73.3.5 virtual uint OutputBuffer::get_push_channel () const [pure virtual]

7.73.3.6 virtual bool OutputBuffer::is_channel_full (uint *channel*) const [pure virtual]

7.73.3.7 virtual bool OutputBuffer::is_empty (uint *channel*) const [pure virtual]

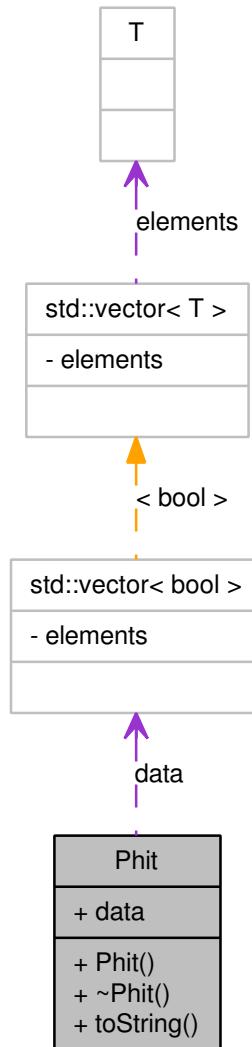
The documentation for this class was generated from the following file:

– **outputBuffer.h**

7.74 Phit Class Reference

```
#include <flit.h>
```

Collaboration diagram for Phit:



Public Member Functions

- **Phit ()**
- **~Phit ()**
- string **toString () const**

Public Attributes

- vector< bool > **data**

7.74.1 Detailed Description

Definition at line 40 of file flit.h.

7.74.2 Constructor & Destructor Documentation

7.74.2.1 Phit::Phit ()

Definition at line 30 of file flit.cc.

7.74.2.2 Phit::~Phit ()

Definition at line 34 of file flit.cc.

References data.

7.74.3 Member Function Documentation

7.74.3.1 string Phit::toString () const

Definition at line 40 of file flit.cc.

References data.

7.74.4 Member Data Documentation

7.74.4.1 vector<bool> Phit::data

Definition at line 45 of file flit.h.

Referenced by `toString()`, and `~Phit()`.

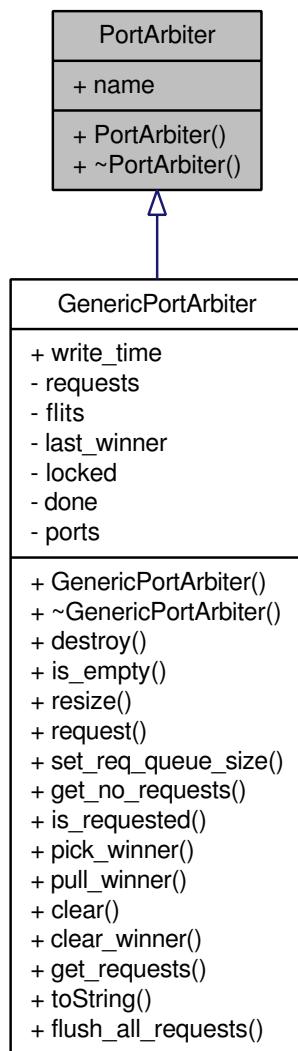
The documentation for this class was generated from the following files:

- **flit.h**
- **flit.cc**

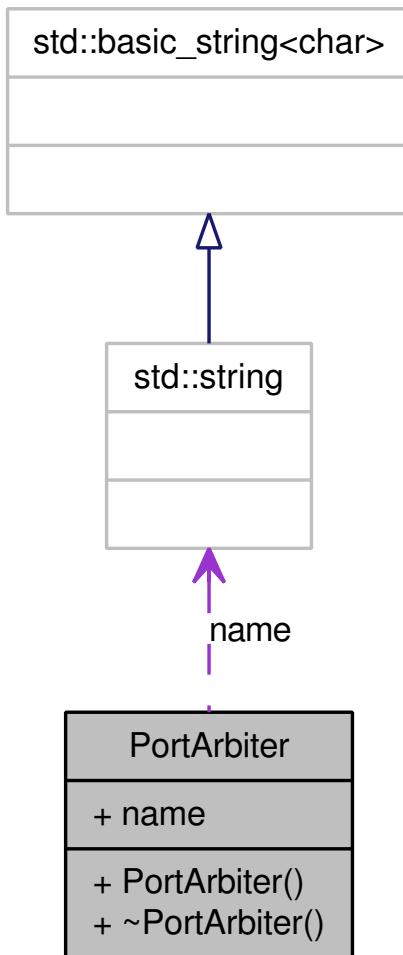
7.75 PortArbiter Class Reference

```
#include <portArbiter.h>
```

Inheritance diagram for PortArbiter:



Collaboration diagram for PortArbiter:



Public Member Functions

- `PortArbiter ()`
- `~PortArbiter ()`

Public Attributes

- string `name`

7.75.1 Detailed Description

Definition at line 31 of file portArbiter.h.

7.75.2 Constructor & Destructor Documentation

7.75.2.1 PortArbiter::PortArbiter () [inline]

Definition at line 34 of file portArbiter.h.

7.75.2.2 PortArbiter::~PortArbiter () [inline]

Definition at line 35 of file portArbiter.h.

7.75.3 Member Data Documentation**7.75.3.1 string PortArbiter::name**

Definition at line 36 of file portArbiter.h.

Referenced by GenericPortArbiter::GenericPortArbiter().

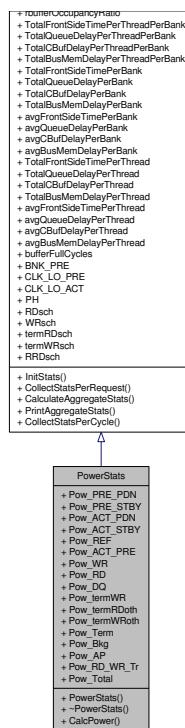
The documentation for this class was generated from the following file:

- **portArbiter.h**

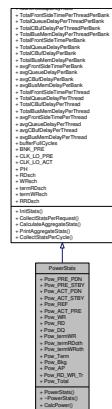
7.76 PowerStats Class Reference

```
#include <stats.h>
```

Inheritance diagram for PowerStats:



Collaboration diagram for PowerStats:



Public Member Functions

- **PowerStats ()**
- **~PowerStats ()**
- void **CalcPower** (float BNK_PRE, float CLK_LO_PRE, float CLK_LO_ACT, float RDsch, float WRsch, float termRDsch, float termWRsch, float RRDsch)

Public Attributes

- float **Pow_PRE_PDN**
- float **Pow_PRE_STBY**
- float **Pow_ACT_PDN**
- float **Pow_ACT_STBY**
- float **Pow_REF**
- float **Pow_CLK_LO_PRE**
- float **Pow_RD**
- float **Pow_actRd**
- float **Pow_actWr**
- float **Pow_actWt**
- float **Pow_Sig**
- float **Pow_RD_MRL_T**
- float **Pow_WT**

- float Pow_RD
- float Pow_DQ
- float Pow_termWR
- float Pow_termRDOTH
- float Pow_termWRoth
- float Pow_Term
- float Pow_Bkg
- float Pow_AP
- float Pow_RD_WR_Tr
- float Pow_Total

7.76.1 Detailed Description

Definition at line 161 of file stats.h.

7.76.2 Constructor & Destructor Documentation

7.76.2.1 PowerStats::PowerStats ()

Definition at line 22 of file stats.cc.

References Pow_ACT_PDN, Pow_ACT_PRE, Pow_ACT_STBY, Pow_AP, Pow_Bkg, Pow_DQ, Pow_PRE_PDN, Pow_PRE_STBY, Pow_RD, Pow_RD_WR_Tr, Pow_REF, Pow_Term, Pow_termRDOTH, Pow_termWR, Pow_termWRoth, Pow_Total, and Pow_WR.

7.76.2.2 PowerStats::~PowerStats ()

Definition at line 43 of file stats.cc.

7.76.3 Member Function Documentation

7.76.3.1 void PowerStats::CalcPower (float BNK_PRE, float CLK_LO_PRE, float CLK_LO_ACT, float RDsch, float WRsch, float termRDsch, float termWRsch, float RRDSch)

Definition at line 47 of file stats.cc.

References BL, DM, DQS, IDD0, IDD2N, IDD2PF, IDD3N, IDD3P, IDD4R, IDD4W, IDD5A, MaxVcc, PdqRD, PdqWR, Pow_ACT_PDN, Pow_ACT_PRE, Pow_ACT_STBY, Pow_AP, Pow_Bkg, Pow_DQ, Pow_PRE_PDN, Pow_PRE_STBY, Pow_RD, Pow_RD_WR_Tr, Pow_REF, Pow_Term, Pow_termRDOTH, Pow_termWR, Pow_termWRoth, Pow_Total, Pow_WR, SysClk, SysVdd, t_CK, tRAS, tRC, tREFI, and tRFC.

7.76.4 Member Data Documentation

7.76.4.1 float PowerStats::Pow_ACT_PDN

Definition at line 172 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.2 float PowerStats::Pow_ACT_PRE

Definition at line 175 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.3 float PowerStats::Pow_ACT_STBY

Definition at line 173 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.4 float PowerStats::Pow_AP

Definition at line 184 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.5 float PowerStats::Pow_Bkg

Definition at line 183 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.6 float PowerStats::Pow_DQ

Definition at line 178 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.7 float PowerStats::Pow_PRE_PDN

Definition at line 170 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.8 float PowerStats::Pow_PRE_STBY

Definition at line 171 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.9 float PowerStats::Pow_RD

Definition at line 177 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.10 float PowerStats::Pow_RD_WR_Tr

Definition at line 185 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.11 float PowerStats::Pow_REF

Definition at line 174 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.12 float PowerStats::Pow_Term

Definition at line 182 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.13 float PowerStats::Pow_termRDoth

Definition at line 180 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.14 float PowerStats::Pow_termWR

Definition at line 179 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.15 float PowerStats::Pow_termWRoth

Definition at line 181 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.16 float PowerStats::Pow_Total

Definition at line 186 of file stats.h.

Referenced by CalcPower(), and PowerStats().

7.76.4.17 float PowerStats::Pow_WR

Definition at line 176 of file stats.h.

Referenced by CalcPower(), and PowerStats().

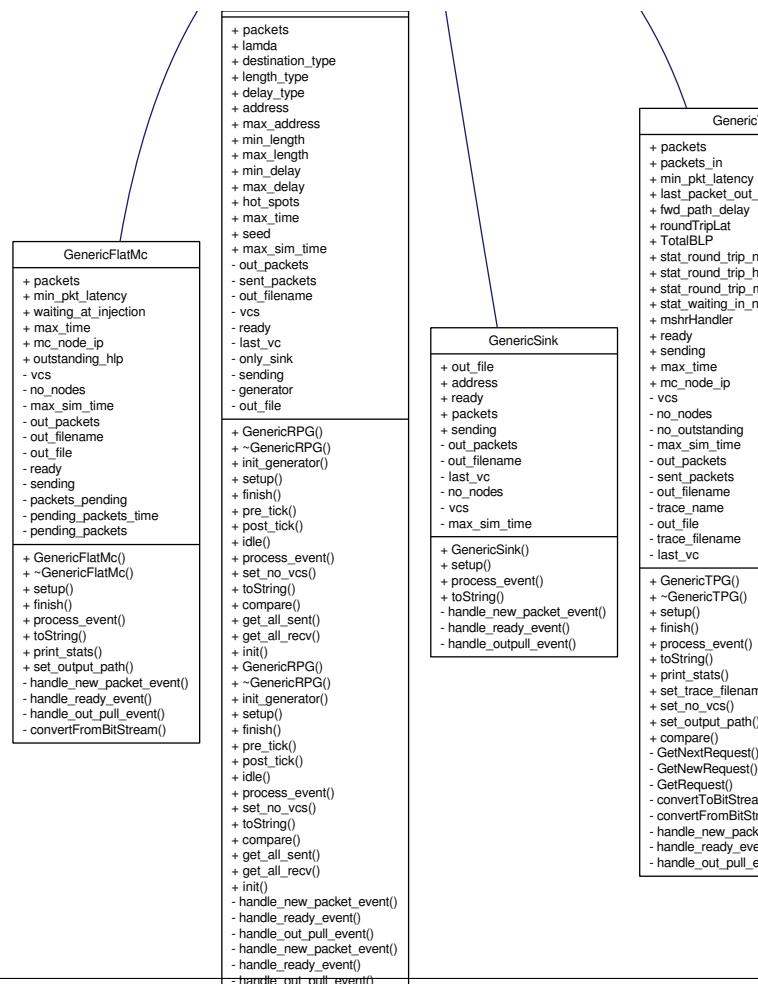
The documentation for this class was generated from the following files:

- **stats.h**
- **stats.cc**

7.77 Processor Class Reference

```
#include <processor.h>
```

Inheritance diagram for Processor:



Collaboration diagram for Processor:

Public Member Functions

- **Processor ()**
- virtual ~**Processor ()**
- void **init ()**
- virtual string **toString () const**
- virtual void **process_event (IrisEvent *e)=0**
- virtual void **setup (uint n, uint v, uint time)=0**
- virtual void **set_output_path (string outpath_name)=0**
- virtual string **print_stats () const =0**

Public Attributes

- vector< **Interface *** > **interface_connections**

7.77.1 Detailed Description

Definition at line 32 of file processor.h.

7.77.2 Constructor & Destructor Documentation

7.77.2.1 Processor::Processor ()

Definition at line 22 of file processor.cc.

References NetworkComponent::processor, and NetworkComponent::type.

7.77.2.2 Processor::~Processor () [virtual]

Definition at line 27 of file processor.cc.

7.77.3 Member Function Documentation

7.77.3.1 void Processor::init ()

Reimplemented in **GenericRPG** (p. 182), and **GenericRPG** (p. 182).

Definition at line 39 of file processor.cc.

7.77.3.2 virtual string Processor::print_stats () const [pure virtual]

Implemented in **GenericFlatMc** (p. 129), **GenericTPG** (p. 197), and **NI** (p. 280).

7.77.3.3 virtual void Processor::process_event (IrisEvent * e) [pure virtual]

Implements **NetworkComponent** (p. 274).

Implemented in **GenericRPG** (p. 182), **GenericFlatMc** (p. 129), **GenericRPG** (p. 182), **GenericSink** (p. 189), **GenericTPG** (p. 197), and **NI** (p. 280).

7.77.3.4 virtual void Processor::set_output_path (string outpath_name) [pure virtual]

Implemented in **GenericFlatMc** (p. 129), **GenericTPG** (p. 197), and **NI** (p. 281).

7.77.3.5 virtual void Processor::setup (uint *n*, uint *v*, uint *time*) [pure virtual]

Implemented in **GenericFlatMc** (p. 129), **GenericRPG** (p. 183), **GenericSink** (p. 189), **GenericTPG** (p. 198), and **NI** (p. 281).

7.77.3.6 string Processor::toString () const [virtual]

Reimplemented from **NetworkComponent** (p. 275).

Reimplemented in **GenericRPG** (p. 183), **GenericFlatMc** (p. 129), **GenericRPG** (p. 183), **GenericSink** (p. 189), **GenericTPG** (p. 198), and **NI** (p. 281).

Definition at line 31 of file processor.cc.

7.77.4 Member Data Documentation

7.77.4.1 vector< Interface* > Processor::interface_connections

Definition at line 36 of file processor.h.

Referenced by **GenericFlatMc::GenericFlatMc()**, **GenericRPG::GenericRPG()**, **GenericTPG::GenericTPG()**, **NI::handle_new_packet_event()**, **GenericTPG::handle_new_packet_event()**, **GenericSink::handle_new_packet_event()**, **GenericRPG::handle_new_packet_event()**, **GenericFlatMc::handle_new_packet_event()**, **NI::handle_out_pull_event()**, **GenericTPG::handle_out_pull_event()**, **GenericRPG::handle_out_pull_event()**, **GenericFlatMc::handle_out_pull_event()**, **GenericSink::handle_outpull_event()**, **NI::NI()**, and **GenericSink::setup()**.

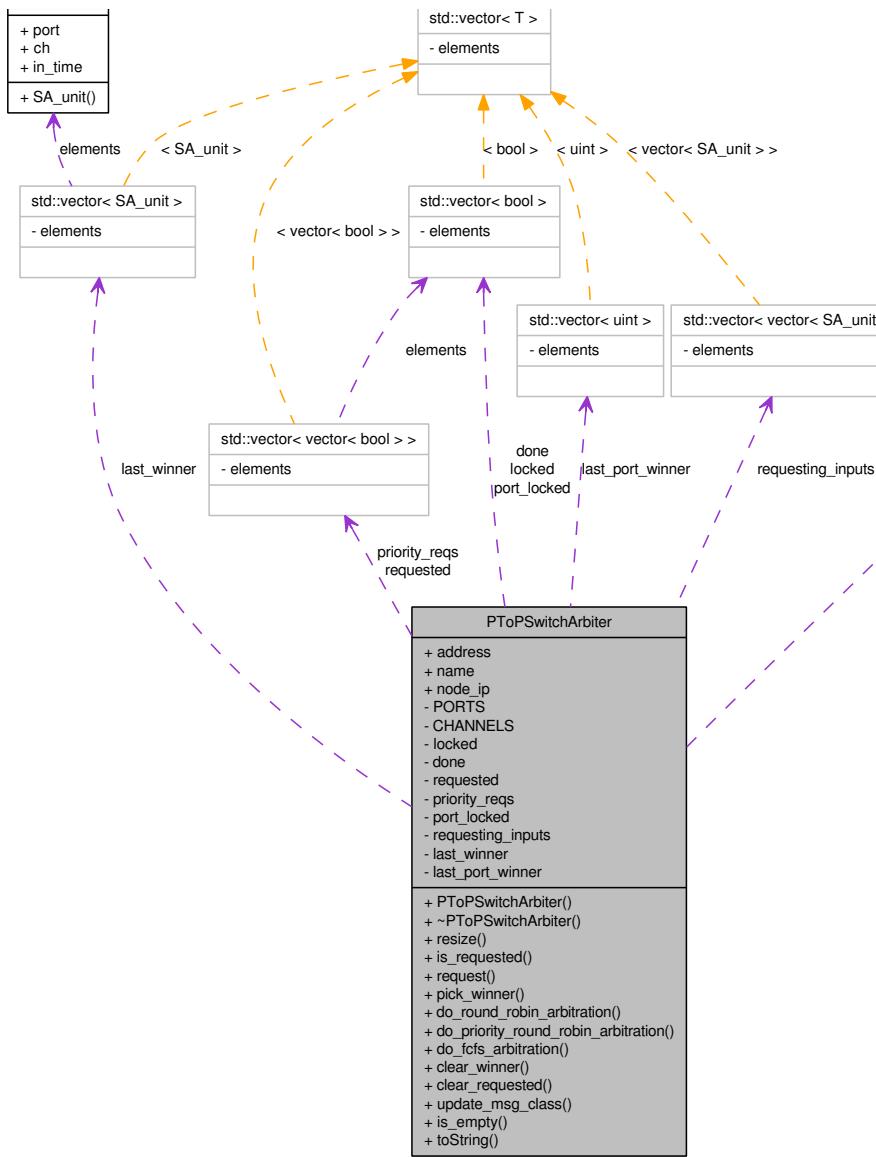
The documentation for this class was generated from the following files:

- **processor.h**
- **processor.cc**

7.78 PToPSwitchArbiter Class Reference

```
#include <p top_swa.h>
```

Collaboration diagram for PToPSwitchArbiter:



Public Member Functions

- **PToPSwitchArbiter ()**
- **~PToPSwitchArbiter ()**
- **void resize (uint p)**

- bool **is_requested** (uint outp, uint inp)
- void **request** (uint p, uint inp)
- SA_unit **pick_winner** (uint p)
- SA_unit **do_round_robin_arbitration** (uint p)
- SA_unit **do_priority_round_robin_arbitration** (uint p)
- SA_unit **do_fcfs_arbitration** (uint p)
- void **clear_winner** (uint p, uint ip)
- void **clear_requested** (uint p, uint ip)
- void **update_msg_class** (uint oport, uint import, message_class m)
- bool **is_empty** ()
- string **toString** () const

Public Attributes

- uint **address**
- string **name**
- uint **node_ip**

Private Attributes

- uint **PORTS**
- uint **CHANNELS**
- vector< bool > **locked**
- vector< bool > **done**
- vector< vector< bool > > **requested**
- vector< vector< bool > > **priority_reqs**
- vector< bool > **port_locked**
- vector< vector< SA_unit > > **requesting_inputs**
- vector< SA_unit > **last_winner**
- vector< uint > **last_port_winner**

7.78.1 Detailed Description

Definition at line 29 of file ptop_swa.h.

7.78.2 Constructor & Destructor Documentation

7.78.2.1 PToPSwitchArbiter::PToPSwitchArbiter ()

Definition at line 24 of file ptop_swa.cc.

References name.

7.78.2.2 PToPSwitchArbiter::~PToPSwitchArbiter ()

Definition at line 29 of file ptop_swa.cc.

7.78.3 Member Function Documentation

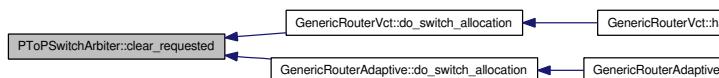
7.78.3.1 void PToPSwitchArbiter::clear_requested (uint p, uint ip)

Definition at line 285 of file ptop_swa.cc.

References priority_reqs, and requested.

Referenced by GenericRouterVct::do_switch_allocation(), and GenericRouterAdaptive::do_switch_allocation().

Here is the caller graph for this function:



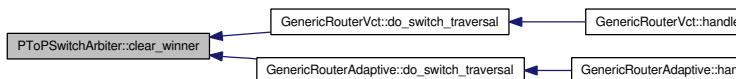
7.78.3.2 void PToPSwitchArbiter::clear_winner (uint p, uint ip)

Definition at line 274 of file ptop_swa.cc.

References done, locked, priority_reqs, and requested.

Referenced by GenericRouterVct::do_switch_traversal(), and GenericRouterAdaptive::do_switch_traversal().

Here is the caller graph for this function:



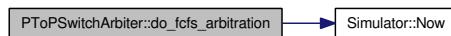
7.78.3.3 SA_unit PToPSwitchArbiter::do_fcfs_arbitration (uint p)

Definition at line 230 of file ptop_swa.cc.

References _DBG_NOARG, done, last_winner, locked, Simulator::Now(), PORTS, requested, and requesting_inputs.

Referenced by pick_winner().

Here is the call graph for this function:



Here is the caller graph for this function:



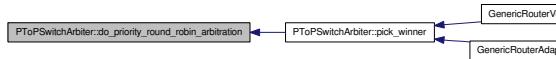
7.78.3.4 SA_unit PToPSwitchArbiter::do_priority_round_robin_arbitration (uint p)

Definition at line 162 of file ptop_swa.cc.

References _DBG_NOARG, done, last_port_winner, last_winner, locked, PORTS, priority_reqs, requested, and requesting_inputs.

Referenced by pick_winner().

Here is the caller graph for this function:



7.78.3.5 SA_unit PToPSwitchArbiter::do_round_robin_arbitration (uint p)

Definition at line 106 of file ptop_swa.cc.

References _DBG_NOARG, done, last_port_winner, last_winner, locked, PORTS, requested, and requesting_inputs.

Referenced by pick_winner().

Here is the caller graph for this function:



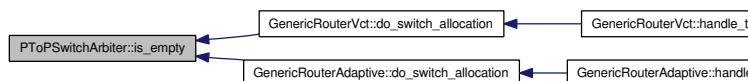
7.78.3.6 bool PToPSwitchArbiter::is_empty ()

Definition at line 302 of file ptop_swa.cc.

References PORTS, and requested.

Referenced by GenericRouterVct::do_switch_allocation(), and GenericRouterAdaptive::do_switch_allocation().

Here is the caller graph for this function:



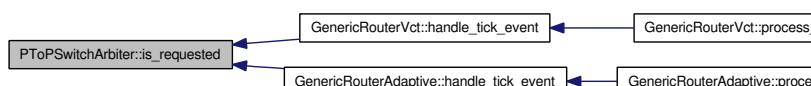
7.78.3.7 bool PToPSwitchArbiter::is_requested (uint outp, uint inp)

Definition at line 69 of file ptop_swa.cc.

References requested.

Referenced by GenericRouterVct::handle_tick_event(), and GenericRouterAdaptive::handle_tick_event().

Here is the caller graph for this function:



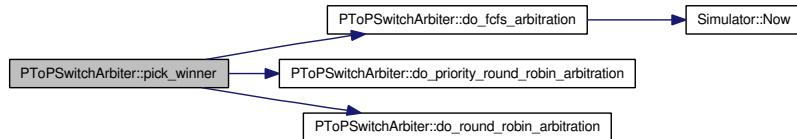
7.78.3.8 SA_unit PToPSwitchArbiter::pick_winner (uint p)

Definition at line 85 of file ptop_swa.cc.

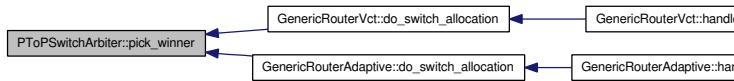
References do_fcfs_arbitration(), do_priority_round_robin_arbitration(), do_round_robin_arbitration(), FCFS, ROUND_ROBIN, ROUND_ROBIN_PRIORITY, and sw_arbitration.

Referenced by GenericRouterVct::do_switch_allocation() and GenericRouterAdaptive::do_switch_allocation().

Here is the call graph for this function:



Here is the caller graph for this function:



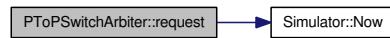
7.78.3.9 void PToPSwitchArbiter::request (uint p, uint inp)

Definition at line 75 of file ptop_swa.cc.

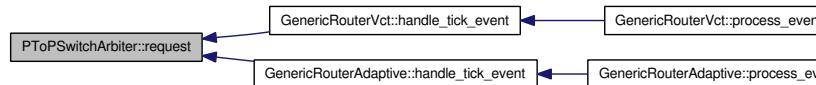
References done, Simulator::Now(), requested, and requesting_inputs.

Referenced by GenericRouterVct::handle_tick_event() and GenericRouterAdaptive::handle_tick_event().

Here is the call graph for this function:



Here is the caller graph for this function:



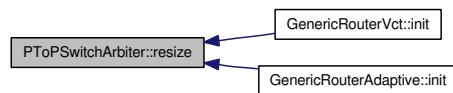
7.78.3.10 void PToPSwitchArbiter::resize (uint p)

Definition at line 34 of file ptop_swa.cc.

References done, last_port_winner, last_winner, locked, port_locked, PORTS, priority_reqs, requested, and requesting_inputs.

Referenced by GenericRouterVct::init() and GenericRouterAdaptive::init().

Here is the caller graph for this function:



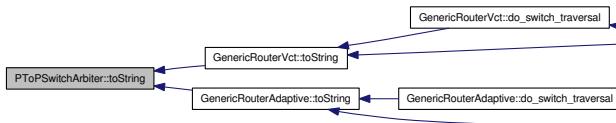
7.78.3.11 string PToPSwitchArbiter::toString () const

Definition at line 315 of file ptop_swa.cc.

References requested.

Referenced by GenericRouterVct::toString(), and GenericRouterAdaptive::toString().

Here is the caller graph for this function:



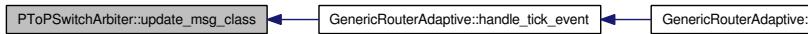
7.78.3.12 void PToPSwitchArbiter::update_msg_class (uint oport, uint import, message_class m)

Definition at line 293 of file ptop_swa.cc.

References priority_reqs, and RESPONSE_PKT.

Referenced by GenericRouterAdaptive::handle_tick_event().

Here is the caller graph for this function:



7.78.4 Member Data Documentation

7.78.4.1 uint PToPSwitchArbiter::address

Definition at line 46 of file ptop_swa.h.

Referenced by GenericRouterAdaptive::init().

7.78.4.2 uint PToPSwitchArbiter::CHANNELS [private]

Definition at line 54 of file ptop_swa.h.

7.78.4.3 vector< bool> PToPSwitchArbiter::done [private]

Definition at line 56 of file ptop_swa.h.

Referenced by clear_winner(), do_fcfs_arbitration(), do_priority_round_robin_arbitration(), do_round_robin_arbitration(), request(), and resize().

7.78.4.4 vector< uint> PToPSwitchArbiter::last_port_winner [private]

Definition at line 62 of file ptop_swa.h.

Referenced by do_priority_round_robin_arbitration(), do_round_robin_arbitration(), and resize().

7.78.4.5 vector< SA_unit > PToPSwitchArbiter::last_winner [private]

Definition at line 61 of file ptop_swa.h.

Referenced by do_fcfs_arbitration(), do_priority_round_robin_arbitration(), do_round_robin_arbitration(), and resize().

7.78.4.6 vector< bool> PToPSwitchArbiter::locked [private]

Definition at line 55 of file ptop_swa.h.

Referenced by clear_winner(), do_fcfs_arbitration(), do_priority_round_robin_arbitration(), do_round_robin_arbitration(), and resize().

7.78.4.7 string PToPSwitchArbiter::name

Definition at line 47 of file ptop_swa.h.

Referenced by PToPSwitchArbiter().

7.78.4.8 uint PToPSwitchArbiter::node_ip

Definition at line 48 of file ptop_swa.h.

Referenced by GenericRouterAdaptive::init().

7.78.4.9 vector< bool> PToPSwitchArbiter::port_locked [private]

Definition at line 59 of file ptop_swa.h.

Referenced by resize().

7.78.4.10 uint PToPSwitchArbiter::PORTS [private]

Definition at line 53 of file ptop_swa.h.

Referenced by do_fcfs_arbitration(), do_priority_round_robin_arbitration(), do_round_robin_arbitration(), is_empty(), and resize().

7.78.4.11 vector< vector <bool> > PToPSwitchArbiter::priority_reqs [private]

Definition at line 58 of file ptop_swa.h.

Referenced by clear_requested(), clear_winner(), do_priority_round_robin_arbitration(), resize(), and update_msg_class().

7.78.4.12 vector< vector <bool> > PToPSwitchArbiter::requested [private]

Definition at line 57 of file ptop_swa.h.

Referenced by clear_requested(), clear_winner(), do_fcfs_arbitration(), do_priority_round_robin_arbitration(), do_round_robin_arbitration(), is_empty(), is_requested(), request(), resize(), and toString().

**7.78.4.13 `vector<vector<SA_unit>>` PToPSwitchArbiter::requesting_inputs
[private]**

Definition at line 60 of file ptop_swa.h.

Referenced by do_fcfs_arbitration(), do_priority_round_robin_arbitration(), do_round_robin_arbitration(), request(), and resize().

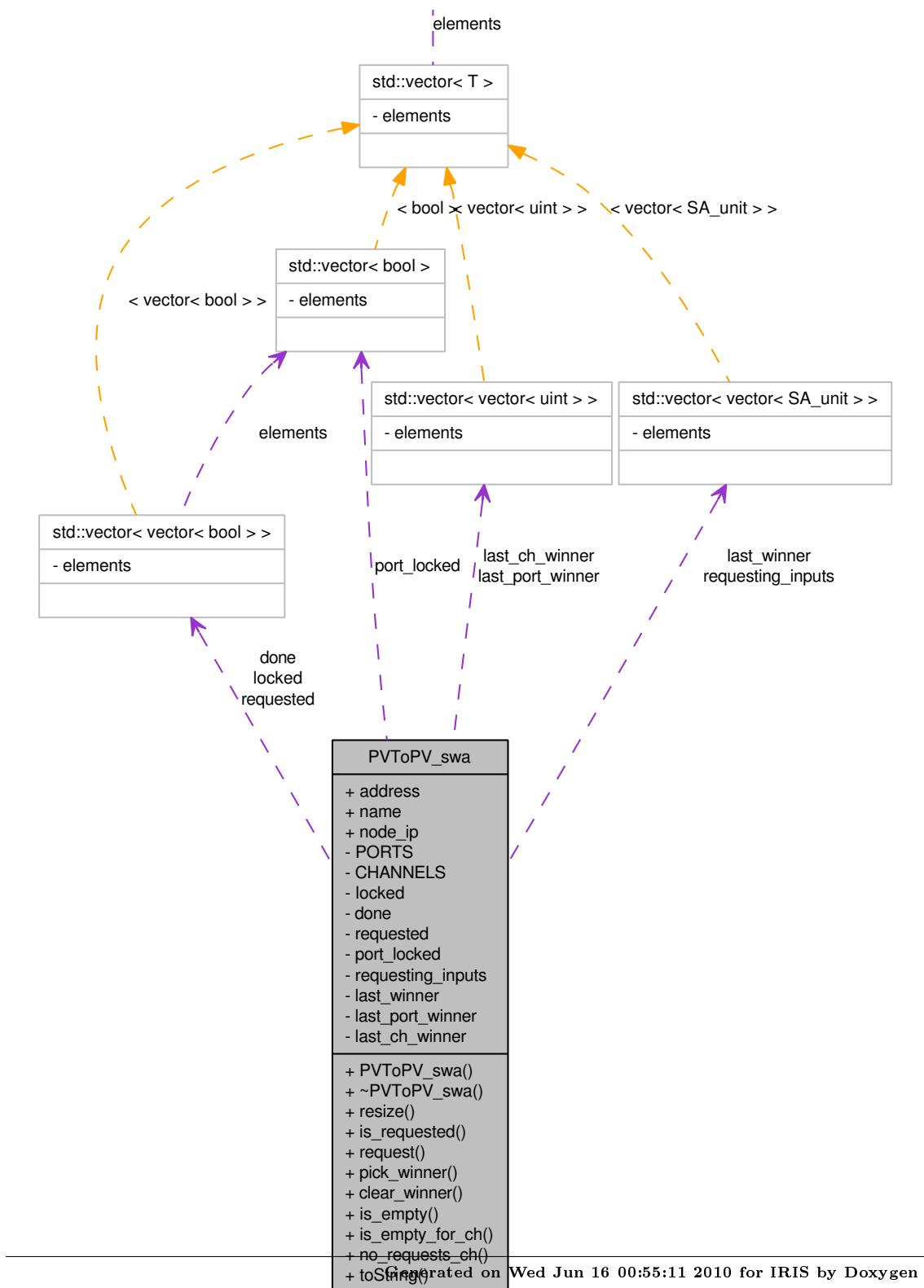
The documentation for this class was generated from the following files:

- **ptop_swa.h**
- **ptop_swa.cc**

7.79 PVToPV_swa Class Reference

```
#include <pvttopv_swa.h>
```

Collaboration diagram for PVToPV_swa:



Public Member Functions

- `PVToPV_swa()`
- `~PVToPV_swa()`
- `void resize (uint p, uint c)`
- `bool is_requested (uint inp, uint inch, uint p, uint c)`
- `void request (uint p, uint c, uint inp, uint inch)`
- `SA_unit pick_winner (uint p, uint c)`
- `void clear_winner (uint p, uint c, uint ip, uint ic)`
- `bool is_empty ()`
- `bool is_empty_for_ch (uint ch)`
- `uint no_requests_ch (uint ch)`
- `string toString () const`

Public Attributes

- `uint address`
- `uint name`
- `uint node_ip`

Private Attributes

- `uint PORTS`
- `uint CHANNELS`
- `vector< vector< bool > > locked`
- `vector< vector< bool > > done`
- `vector< vector< bool > > requested`
- `vector< bool > port_locked`
- `vector< vector< SA_unit > > requesting_inputs`
- `vector< vector< SA_unit > > last_winner`
- `vector< vector< uint > > last_port_winner`
- `vector< vector< uint > > last_ch_winner`

7.79.1 Detailed Description

Definition at line 27 of file pvtov_swa.h.

7.79.2 Constructor & Destructor Documentation

7.79.2.1 PVToPV_swa::PVToPV_swa ()

Definition at line 24 of file pvtov_swa.cc.

7.79.2.2 PVToPV_swa::~PVToPV_swa ()

Definition at line 29 of file pvtov_swa.cc.

7.79.3 Member Function Documentation

7.79.3.1 void PVToPV_swa::clear_winner (uint p, uint c, uint ip, uint ic)

Definition at line 150 of file pvtov_swa.cc.

References CHANNELS, done, locked, and requested.

7.79.3.2 bool PVToPV_swa::is_empty ()

Definition at line 160 of file pvtopy_swa.cc.

References CHANNELS, PORTS, and requested.

7.79.3.3 bool PVToPV_swa::is_empty_for_ch (uint ch)

Definition at line 174 of file pvtopy_swa.cc.

References requested.

7.79.3.4 bool PVToPV_swa::is_requested (uint inp, uint inch, uint p, uint c)

Definition at line 82 of file pvtopy_swa.cc.

References locked.

7.79.3.5 uint PVToPV_swa::no_requests_ch (uint ch)

Definition at line 183 of file pvtopy_swa.cc.

References requested.

7.79.3.6 SA_unit PVToPV_swa::pick_winner (uint p, uint c)

Definition at line 100 of file pvtopy_swa.cc.

References CHANNELS, done, last_port_winner, last_winner, locked, PORTS, requested, and requesting_inputs.

7.79.3.7 void PVToPV_swa::request (uint p, uint c, uint inp, uint inch)

Definition at line 89 of file pvtopy_swa.cc.

References CHANNELS, done, requested, and requesting_inputs.

7.79.3.8 void PVToPV_swa::resize (uint p, uint c)

Definition at line 35 of file pvtopy_swa.cc.

References CHANNELS, done, last_ch_winner, last_port_winner, last_winner, locked, port_locked, PORTS, requested, and requesting_inputs.

7.79.3.9 string PVToPV_swa::toString () const

Definition at line 195 of file pvtopy_swa.cc.

References requested.

7.79.4 Member Data Documentation**7.79.4.1 uint PVToPV_swa::address**

Definition at line 41 of file pvtopy_swa.h.

7.79.4.2 uint PVToPV_swa::CHANNELS [private]

Definition at line 49 of file pvtovpv_swa.h.

Referenced by clear_winner(), is_empty(), pick_winner(), request(), and resize().

7.79.4.3 vector<vector<bool>> PVToPV_swa::done [private]

Definition at line 51 of file pvtovpv_swa.h.

Referenced by clear_winner(), pick_winner(), request(), and resize().

7.79.4.4 vector<vector<uint>> PVToPV_swa::last_ch_winner [private]

Definition at line 57 of file pvtovpv_swa.h.

Referenced by resize().

7.79.4.5 vector<vector<uint>> PVToPV_swa::last_port_winner [private]

Definition at line 56 of file pvtovpv_swa.h.

Referenced by pick_winner(), and resize().

7.79.4.6 vector<vector<SA_unit>> PVToPV_swa::last_winner [private]

Definition at line 55 of file pvtovpv_swa.h.

Referenced by pick_winner(), and resize().

7.79.4.7 vector<vector<bool>> PVToPV_swa::locked [private]

Definition at line 50 of file pvtovpv_swa.h.

Referenced by clear_winner(), is_requested(), pick_winner(), and resize().

7.79.4.8 uint PVToPV_swa::name

Definition at line 42 of file pvtovpv_swa.h.

7.79.4.9 uint PVToPV_swa::node_ip

Definition at line 43 of file pvtovpv_swa.h.

7.79.4.10 vector<bool> PVToPV_swa::port_locked [private]

Definition at line 53 of file pvtovpv_swa.h.

Referenced by resize().

7.79.4.11 uint PVToPV_swa::PORTS [private]

Definition at line 48 of file pvtovpv_swa.h.

Referenced by is_empty(), pick_winner(), and resize().

7.79.4.12 `vector< vector<bool> > PVToPV_swa::requested` [private]

Definition at line 52 of file pvtovpv_swa.h.

Referenced by clear_winner(), is_empty(), is_empty_for_ch(), no_requests_ch(), pick_winner(), request(), resize(), and toString().

7.79.4.13 `vector< vector<SA_unit> > PVToPV_swa::requesting_inputs` [private]

Definition at line 54 of file pvtovpv_swa.h.

Referenced by pick_winner(), request(), and resize().

The documentation for this class was generated from the following files:

- **pvtovpv_swa.h**
- **pvtovpv_swa.cc**

7.80 RankHandler Class Reference

```
#include <rank_handler.h>
```

Collaboration diagram for RankHandler:

Public Member Functions

- **RankHandler ()**
- **~RankHandler ()**

Public Attributes

- short **rankId**
- **ReqBuffer rbuffer [NO_OF_BUFFERS]**
- **BankHandler bank [NO_OF_BANKS]**
- unsigned int **readsOWrite**
- unsigned int **prevReadsOWrite**

7.80.1 Detailed Description

Definition at line 46 of file rank_handler.h.

7.80.2 Constructor & Destructor Documentation

7.80.2.1 RankHandler::RankHandler ()

Definition at line 32 of file rank_handler.cc.

References prevReadsOWrite, and readsOWrite.

7.80.2.2 RankHandler::~RankHandler ()

Definition at line 45 of file rank_handler.cc.

7.80.3 Member Data Documentation

7.80.3.1 BankHandler RankHandler::bank[NO_OF_BANKS]

Definition at line 53 of file rank_handler.h.

Referenced by RequestHandler::RequestHandler().

7.80.3.2 unsigned int RankHandler::prevReadsOWrite

Definition at line 55 of file rank_handler.h.

Referenced by RankHandler(), BankHandler::RestorePrevState(), and BankHandler::SetPrevState().

7.80.3.3 short RankHandler::rankId

Definition at line 51 of file rank_handler.h.

Referenced by RequestHandler::RequestHandler().

7.80.3.4 ReqBuffer RankHandler::rbuffer[NO_OF_BUFFERS]

Definition at line 52 of file rank_handler.h.

Referenced by BankHandler::FCFS(), BankHandler::FindHighest(), BankHandler::FRFCFS(), BankHandler::HighestRankedFirst(), RequestHandler::MarkAll(), RequestHandler::MarkBatchOnly(), BankHandler::OldestFirst(), BankHandler::PARBS(), BankHandler::ReadsFirst(), BankHandler::RowHitFirst(), BankHandler::ScheduleUnmarked(), and BankHandler::SetBypasses().

7.80.3.5 unsigned int RankHandler::readsOWrite

Definition at line 54 of file rank_handler.h.

Referenced by RankHandler(), BankHandler::ReadsFirst(), BankHandler::RestorePrevState(), BankHandler::SetPrevState(), and BankHandler::SetReadsOWrite().

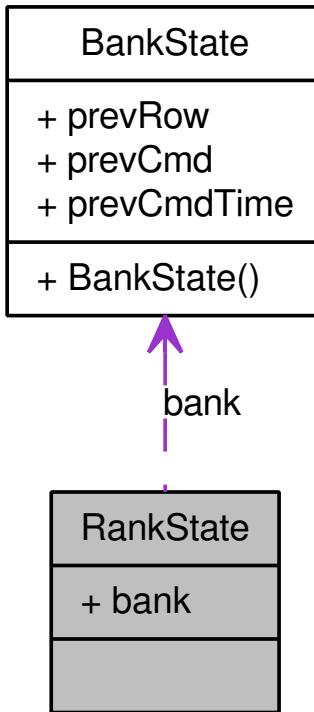
The documentation for this class was generated from the following files:

- [rank_handler.h](#)
- [rank_handler.cc](#)

7.81 RankState Struct Reference

#include <cmd_issuer.h>

Collaboration diagram for RankState:



Public Attributes

- BankState bank [NO_OF_BANKS]

7.81.1 Detailed Description

Definition at line 90 of file cmd_issuer.h.

7.81.2 Member Data Documentation

7.81.2.1 BankState RankState::bank[NO_OF_BANKS]

Definition at line 92 of file cmd_issuer.h.

Referenced by CmdIssuer::BankNotBusy(), and CmdIssuer::SetPrevState().

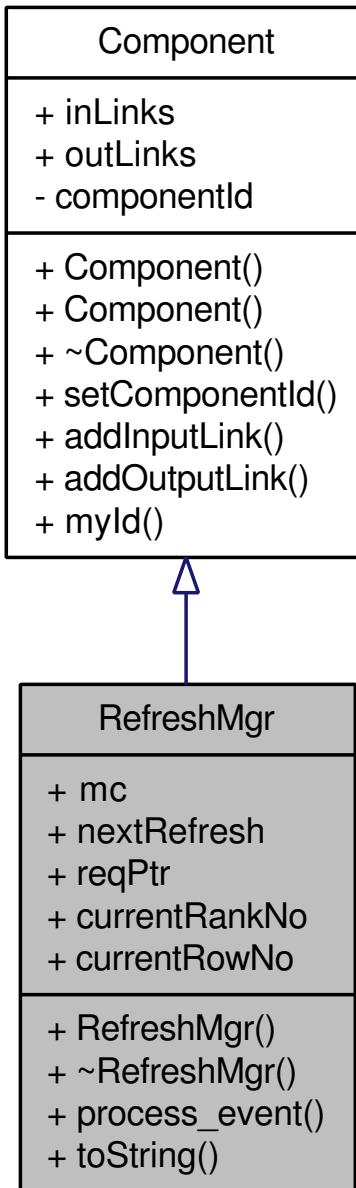
The documentation for this struct was generated from the following file:

- cmd_issuer.h

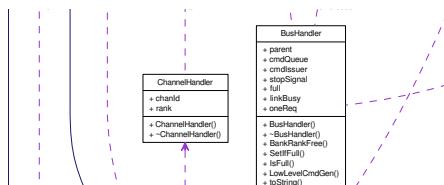
7.82 RefreshMgr Class Reference

```
#include <refresh_manager.h>
```

Inheritance diagram for RefreshMgr:



Collaboration diagram for RefreshMgr:



Public Member Functions

- **RefreshMgr ()**
- **~RefreshMgr ()**
- void **process_event (IrisEvent *e)**
- std::string **toString ()**

Public Attributes

- Component * **mc**
- Time **nextRefresh**
- RequestHandler * **reqPtr**
- unsigned int **currentRankNo**
- unsigned int **currentRowNo**

7.82.1 Detailed Description

Definition at line 44 of file refresh_manager.h.

7.82.2 Constructor & Destructor Documentation

7.82.2.1 RefreshMgr::RefreshMgr ()

Definition at line 39 of file refresh_manager.cc.

References currentRankNo, currentRowNo, and nextRefresh.

7.82.2.2 RefreshMgr::~RefreshMgr ()

Definition at line 53 of file refresh_manager.cc.

7.82.3 Member Function Documentation

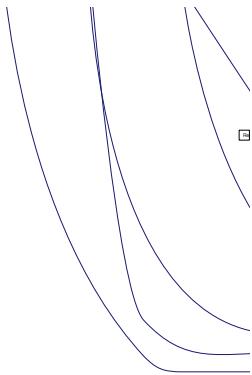
7.82.3.1 void RefreshMgr::process_event (IrisEvent * e)

Definition at line 65 of file refresh_manager.cc.

References RequestHandler::busHandler, BusHandler::cmdIssuer, Request::cmdType, currentRankNo, currentRowNo, IrisEvent::dst, BusHandler::LowLevelCmdGen(), nextRefresh, NO_OF_CHANNELS, NO_OF_RANKS, NO_OF_ROWS, Simulator::Now(), CmdIssuer::process_event(), Request::rankNo, REFRESH, REFRESH_INC, reqPtr, ROW_SIZE, Request::rowNo, Simulator::Schedule(), IrisEvent::src, and START.

Referenced by MC::StartRefresh().

Here is the call graph for this function:



Here is the caller graph for this function:



7.82.3.2 std::string RefreshMgr::toString ()

7.82.4 Member Data Documentation

7.82.4.1 unsigned int RefreshMgr::currentRankNo

Definition at line 53 of file refresh_manager.h.

Referenced by process_event(), and RefreshMgr().

7.82.4.2 unsigned int RefreshMgr::currentRowNo

Definition at line 54 of file refresh_manager.h.

Referenced by process_event(), and RefreshMgr().

7.82.4.3 Component* RefreshMgr::mc

Definition at line 49 of file refresh_manager.h.

Referenced by MC::Init().

7.82.4.4 Time RefreshMgr::nextRefresh

Definition at line 50 of file refresh_manager.h.

Referenced by process_event(), RefreshMgr(), and MC::StartRefresh().

7.82.4.5 RequestHandler* RefreshMgr::reqPtr

Definition at line 51 of file refresh_manager.h.

Referenced by MC::Init(), and process_event().

The documentation for this class was generated from the following files:

- [refresh_manager.h](#)
- [refresh_manager.cc](#)

7.83 Request Class Reference

```
#include <request.h>
```

Collaboration diagram for Request:



Public Member Functions

- **Request ()**
- **~Request ()**

Public Attributes

- **UInt mcNo**
- **UInt channelNo**
- **UInt dimmNo**
- **UInt rankNo**
- **UInt bankNo**
- **UInt columnNo**
- **UInt rowNo**

- **UInt lowerBits**
- **Command_t cmdType**
- **UInt threadId**
- **Addr_t address**
- **Data data**
- **bool mark**
- **bool local**
- **bool scheduledInMSHR**
- **CStatus status**
- **int tag**
- **bool serviced**
- **Time startTime**
- **Time arrivalTime**
- **Time retireTime**
- **Time busInsertionTime**
- **Time rbufferInsertionTime**
- **Time cbufferInsertionTime**
- **Time throttleTime**
- **uint hop_count**
- **double avg_network_latency**

7.83.1 Detailed Description

Definition at line 52 of file request.h.

7.83.2 Constructor & Destructor Documentation

7.83.2.1 Request::Request ()

Definition at line 28 of file request.cc.

References address, arrivalTime, avg_network_latency, bankNo, busInsertionTime, cbufferInsertionTime, channelNo, columnNo, dimmNo, hop_count, local, lowerBits, mark, mcNo, NO_OF_BANKS, NO_OF_CHANNELS, NO_OF_COLUMNS, NO_OF_RANKS, NO_OF_ROWS, rankNo, rbufferInsertionTime, retireTime, rowNo, scheduledInMSHR, serviced, startTime, tag, and threadId.

7.83.2.2 Request::~Request ()

Definition at line 66 of file request.cc.

7.83.3 Member Data Documentation

7.83.3.1 Addr_t Request::address

Definition at line 67 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), NI::convertFromBitStream(), MCFrontEnd::convertFromBitStream(), GenericTPG::convertFromBitStream(), GenericFlatMc::convertFromBitStream(), NI::convertToBitStream(), GenericTPG::convertToBitStream(), MSHR_H::DeleteInMSHR(), GenericTPG::GetNextRequest(), NI::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), MSHR_H::local_map_addr(), main(), AddrMap::map_addr(), ResponseHandler::process_event(), RequestHandler::process_event(), MSHR_H::process_event(), DRAMChannel::process_event(), CmdIssuer::process_event(), BankHandler::process_event(), AddrMap::process_event(), Request(), and GenericTPG::setup().

7.83.3.2 Time Request::arrivalTime

Definition at line 78 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), GenericTPG::convertToBitStream(), GenericTPG::GetNextRequest(), NI::handle_new_packet_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), main(), ResponseHandler::process_event(), MSHR_H::process_event(), DRAMChannel::process_event(), Request(), and GenericTPG::setUp().

7.83.3.3 double Request::avg_network_latency

Definition at line 87 of file request.h.

Referenced by NI::handle_new_packet_event(), NI::handle_out_pull_event(), and Request().

7.83.3.4 UInt Request::bankNo

Definition at line 61 of file request.h.

Referenced by CmdIssuer::BankNotBusy(), CmdIssuer::BanksFirstReq(), CmdIssuer::CanSchedule(), Statistic::CollectStatsPerRequest(), BankHandler::HighestRankedFirst(), MSHR_H::local_map_addr(), AddrMap::map_addr(), BankHandler::OldestFirst(), DRAMChannel::process_event(), DataBusHandler::process_event(), CmdIssuer::process_event(), BankHandler::process_event(), AddrMap::process_event(), BankHandler::ReadsFirst(), Request(), BankHandler::RowHitFirst(), BankHandler::ScheduleUnmarked(), and CmdIssuer::SetPrevState().

7.83.3.5 Time Request::busInsertionTime

Definition at line 80 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), ResponseHandler::process_event(), CmdIssuer::process_event(), and Request().

7.83.3.6 Time Request::cbufferInsertionTime

Definition at line 82 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), BankHandler::process_event(), and Request().

7.83.3.7 UInt Request::channelNo

Definition at line 58 of file request.h.

Referenced by MSHR_H::local_map_addr(), BusHandler::LowLevelCmdGen(), AddrMap::map_addr(), CmdIssuer::process_event(), BankHandler::process_event(), AddrMap::process_event(), and Request().

7.83.3.8 Command_t Request::cmdType

Definition at line 65 of file request.h.

Referenced by NI::convertFromBitStream(), MCFrontEnd::convertFromBitStream(), GenericTPG::convertFromBitStream(), GenericFlatMc::convertFromBitStream(), NI::convertToBitStream(), GenericTPG::convertToBitStream(), GenericTPG::GetNextRequest(), NI::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), BankHandler::HighestRankedFirst(), BusHandler::LowLevelCmdGen(), main(), BankHandler::OldestFirst(), ResponseHandler::process_event(), RefreshMgr::process_event(), MSHR_H::process_event(), BankHandler::process_event(), BankHandler::ReadsFirst(), BankHandler::RowHitFirst(), BankHandler::ScheduleUnmarked(), and GenericTPG::setup().

7.83.3.9 UInt Request::columnNo

Definition at line 62 of file request.h.

Referenced by MSHR_H::local_map_addr(), AddrMap::map_addr(), AddrMap::process_event(), and Request().

7.83.3.10 Data Request::data

Definition at line 68 of file request.h.

Referenced by by MCFrontEnd::convertFromBitStream(), GenericTPG::convertFromBitStream(), DataBusHandler::process_event(), and DRAMCmState::set().

7.83.3.11 UInt Request::dimmNo

Definition at line 59 of file request.h.

Referenced by Request().

7.83.3.12 uint Request::hop_count

Definition at line 86 of file request.h.

Referenced by NI::handle_new_packet_event(), NI::handle_out_pull_event(), and Request().

7.83.3.13 bool Request::local

Definition at line 70 of file request.h.

Referenced by Request().

7.83.3.14 UInt Request::lowerBits

Definition at line 64 of file request.h.

Referenced by MSHR_H::local_map_addr(), AddrMap::map_addr(), and Request().

7.83.3.15 bool Request::mark

Definition at line 69 of file request.h.

Referenced by Request().

7.83.3.16 UInt Request::mcNo

Definition at line 57 of file request.h.

Referenced by GenericTPG::convertFromBitStream(), MSHR_H::DeleteInMSHR(), GenericTPG::handle_out_pull_event(), MSHR_H::process_event(), and Request().

7.83.3.17 UInt Request::rankNo

Definition at line 60 of file request.h.

Referenced by CmdIssuer::BankNotBusy(), CmdIssuer::CanSchedule(), MSHR_H::local_map_addr(), AddrMap::map_addr(), RefreshMgr::process_event(), DRAMChannel::process_event(), DataBusHandler::process_event(), CmdIssuer::process_event(), AddrMap::process_event(), Request(), and CmdIssuer::SetPrevState().

7.83.3.18 Time Request::rbufferInsertionTime

Definition at line 81 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), AddrMap::process_event(), and Request().

7.83.3.19 Time Request::retireTime

Definition at line 79 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), NI::handle_out_pull_event(), ResponseHandler::process_event(), DRAMChannel::process_event(), and Request().

7.83.3.20 UInt Request::rowNo

Definition at line 63 of file request.h.

Referenced by BankHandler::HighestRankedFirst(), MSHR_H::local_map_addr(), AddrMap::map_addr(), BankHandler::OldestFirst(), RefreshMgr::process_event(), BankHandler::process_event(), AddrMap::process_event(), BankHandler::ReadsFirst(), Request(), BankHandler::RowHitFirst(), BankHandler::ScheduleUnmarked(), and CmdIssuer::SetPrevState().

7.83.3.21 bool Request::scheduledInMSHR

Definition at line 72 of file request.h.

Referenced by Request().

7.83.3.22 bool Request::serviced

Definition at line 75 of file request.h.

Referenced by Request().

7.83.3.23 Time Request::startTime

Definition at line 77 of file request.h.

Referenced by MSHR_H::DeleteInMSHR(), NI::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), MSHR_H::process_event(), and Request().

7.83.3.24 CStatus Request::status

Definition at line 73 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), BankHandler::HighestRankedFirst(), BusHandler::LowLevelCmdGen(), BankHandler::OldestFirst(), BankHandler::ReadsFirst(), BankHandler::RowHitFirst(), and BankHandler::ScheduleUnmarked().

7.83.3.25 int Request::tag

Definition at line 74 of file request.h.

Referenced by ResponseHandler::process_event(), BankHandler::process_event(), Request(), and ResponseHandler::SearchBuffer().

7.83.3.26 UInt Request::threadId

Definition at line 66 of file request.h.

Referenced by Statistic::CollectStatsPerRequest(), NI::convertFromBitStream(), MCFrontEnd::convertFromBitStream(), GenericFlatMc::convertFromBitStream(), GenericTPG::convertToBitStream(), GenericTPG::GetNextRequest(), NI::handle_new_packet_event(), NI::handle_out_pull_event(), BankHandler::HighestRankedFirst(), main(), MSHR_H::process_event(), Request(), and GenericTPG::setup().

7.83.3.27 Time Request::throttleTime

Definition at line 83 of file request.h.

Referenced by Statistic::CollectStatsPerRequest().

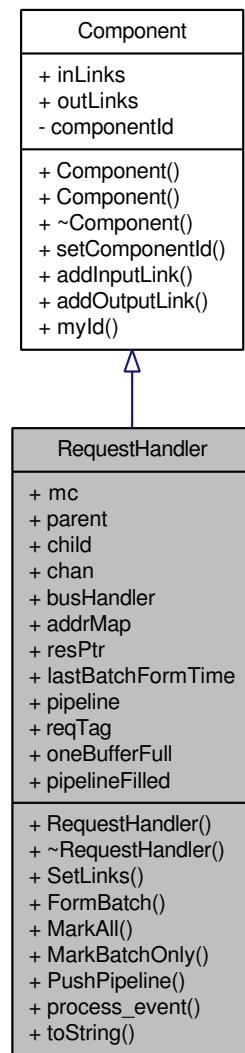
The documentation for this class was generated from the following files:

- **request.h**
- **request.cc**

7.84 RequestHandler Class Reference

```
#include <request_handler.h>
```

Inheritance diagram for RequestHandler:



Collaboration diagram for RequestHandler:



Public Member Functions

- **RequestHandler ()**
- **~RequestHandler ()**
- void **SetLinks ()**
- void **FormBatch ()**
- void **MarkAll ()**
- void **MarkBatchOnly ()**
- void **PushPipeline (Request *req)**
- void **process_event (IrisEvent *e)**
- std::string **toString ()**

Public Attributes

- **Component * mc**
- **Component * parent**
- **Component * child**
- **ChannelHandler chan [NO_OF_CHANNELS]**
- **BusHandler * busHandler**
- **AddrMap * addrMap**
- **Component * resPtr**
- **Time lastBatchFormTime**
- **Request pipeline**
- int **reqTag**
- bool **oneBufferFull**
- bool **pipelineFilled**

7.84.1 Detailed Description

Definition at line 49 of file request_handler.h.

7.84.2 Constructor & Destructor Documentation

7.84.2.1 RequestHandler::RequestHandler ()

Definition at line 36 of file request_handler.cc.

References `addrMap`, `RankHandler::bank`, `BankHandler::bankId`, `BankHandler::bufferId`, `busHandler`, `chan`, `ChannelHandler::chanId`, `lastBatchFormTime`, `BankHandler::myChannel`, `BankHandler::myRank`, `NO_OF_BANKS`, `NO_OF_BUFFERS`, `NO_OF_CHANNELS`, `NO_OF_RANKS`, `oneBufferFull`, `BankHandler::parent`, `BusHandler::parent`, `AddrMap::parent`, `pipelineFilled`, `ChannelHandler::rank`, `RankHandler::rankId`, and `reqTag`.

7.84.2.2 RequestHandler::~RequestHandler ()

Definition at line 71 of file request_handler.cc.

References `addrMap`, and `busHandler`.

7.84.3 Member Function Documentation

7.84.3.1 void RequestHandler::FormBatch ()

Definition at line 184 of file request_handler.cc.

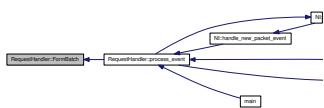
References BATCH_FORM_TIME, lastBatchFormTime, and MarkBatchOnly().

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.84.3.2 void RequestHandler::MarkAll ()

Definition at line 195 of file request_handler.cc.

References chan, NO_OF_BUFFERS, NO_OF_CHANNELS, NO_OF_RANKS, ChannelHandler::rank, and RankHandler::rbuffer.

7.84.3.3 void RequestHandler::MarkBatchOnly ()

Definition at line 204 of file request_handler.cc.

References chan, MAX_BATCH_SIZE, NO_OF_BUFFERS, NO_OF_CHANNELS, NO_OF_RANKS, NO_OF_THREADS, ChannelHandler::rank, and RankHandler::rbuffer.

Referenced by FormBatch().

Here is the caller graph for this function:



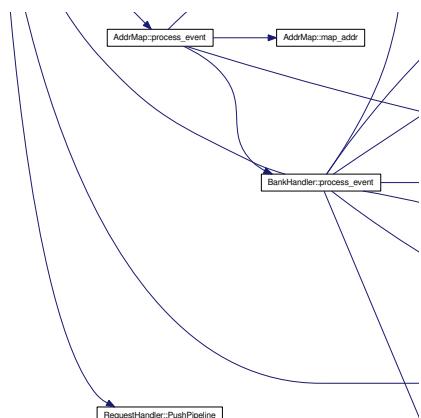
7.84.3.4 void RequestHandler::process_event (IrisEvent * e)

Definition at line 86 of file request_handler.cc.

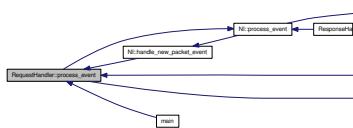
References Request::address, addrMap, BATCH_FORM_TIME, busHandler, CONTINUE, IrisEvent::dst, IrisEvent::event_data, FormBatch(), BusHandler::full, lastBatchFormTime, mc, NEW_PACKET_EVENT, NO_OF_CHANNELS, Simulator::Now(), oneBufferFull, pipeline, pipelineFilled, NI::process_event(), AddrMap::process_event(), PushPipeline(), Simulator::Schedule(), IrisEvent::src, START, START_CMD_QUEUE, STOP_CMD_QUEUE, BusHandler::stopSignal, and IrisEvent::type.

Referenced by NI::handle_new_packet_event(), main(), and BankHandler::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



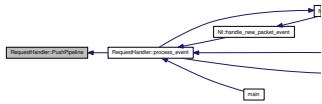
7.84.3.5 void RequestHandler::PushPipeline (Request * *req*)

Definition at line 174 of file request_handler.cc.

References pipeline.

Referenced by process_event().

Here is the caller graph for this function:



7.84.3.6 void RequestHandler::SetLinks ()

Definition at line 77 of file request_handler.cc.

References busHandler, child, CmdIssuer::cmdBus, BusHandler::cmdIssuer, CmdIssuer::dataBus, and NO_OF_CHANNELS.

Referenced by MC::Init().

Here is the caller graph for this function:



7.84.3.7 std::string RequestHandler::toString (void)

Definition at line 179 of file request_handler.cc.

7.84.4 Member Data Documentation

7.84.4.1 AddrMap* RequestHandler::addrMap

Definition at line 59 of file request_handler.h.

Referenced by process_event(), RequestHandler(), and ~RequestHandler().

7.84.4.2 BusHandler* RequestHandler::busHandler

Definition at line 58 of file request_handler.h.

Referenced by process_event(), RefreshMgr::process_event(), RequestHandler(), SetLinks(), and ~RequestHandler().

7.84.4.3 ChannelHandler RequestHandler::chan[NO_OF_CHANNELS]

Definition at line 57 of file request_handler.h.

Referenced by MarkAll(), MarkBatchOnly(), and RequestHandler().

7.84.4.4 Component* RequestHandler::child

Definition at line 56 of file request_handler.h.

Referenced by MC::Init(), and SetLinks().

7.84.4.5 Time RequestHandler::lastBatchFormTime

Definition at line 61 of file request_handler.h.

Referenced by FormBatch(), process_event(), and RequestHandler().

7.84.4.6 Component* RequestHandler::mc

Definition at line 54 of file request_handler.h.

Referenced by MC::Init(), and process_event().

7.84.4.7 bool RequestHandler::oneBufferFull

Definition at line 64 of file request_handler.h.

Referenced by main(), process_event(), and RequestHandler().

7.84.4.8 Component* RequestHandler::parent

Definition at line 55 of file request_handler.h.

Referenced by MC::Init().

7.84.4.9 Request RequestHandler::pipeline

Definition at line 62 of file request_handler.h.

Referenced by process_event(), and PushPipeline().

7.84.4.10 bool RequestHandler::pipelineFilled

Definition at line 65 of file request_handler.h.

Referenced by process_event(), and RequestHandler().

7.84.4.11 int RequestHandler::reqTag

Definition at line 63 of file request_handler.h.

Referenced by RequestHandler().

7.84.4.12 Component* RequestHandler::resPtr

Definition at line 60 of file request_handler.h.

Referenced by MC::Init().

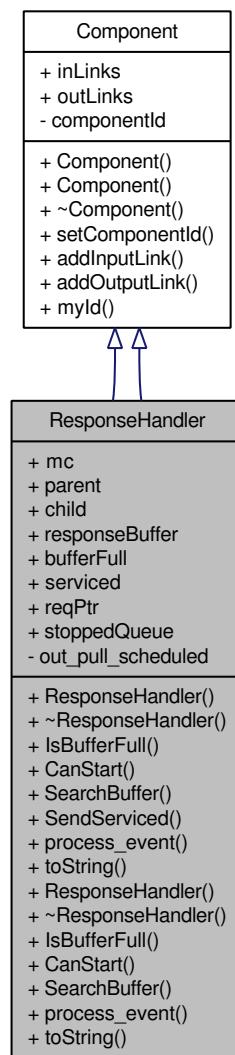
The documentation for this class was generated from the following files:

- [request_handler.h](#)
- [request_handler.cc](#)

7.85 ResponseHandler Class Reference

```
#include <response_handler.h>
```

Inheritance diagram for ResponseHandler:



Collaboration diagram for ResponseHandler:

Public Member Functions

- **ResponseHandler ()**
- **~ResponseHandler ()**
- **bool IsBufferFull ()**
- **bool CanStart ()**
- **unsigned int SearchBuffer (DRAMCmdState *cmd)**
- **unsigned int SendServiced ()**
- **void process_event (IrisEvent *e)**
- **std::string toString ()**
- **ResponseHandler ()**
- **~ResponseHandler ()**
- **bool IsBufferFull ()**
- **bool CanStart ()**
- **unsigned int SearchBuffer (DRAMCmdState *cmd)**
- **void process_event (IrisEvent *e)**
- **std::string toString ()**

Public Attributes

- **Component * mc**
- **Component * parent**
- **Component * child**
- **vector< Request > responseBuffer**
- **bool bufferFull**
- **bool serviced [RESPONSE_BUFFER_SIZE]**
- **Component * reqPtr**
- **bool stoppedQueue**

Private Attributes

- **bool out_pull_scheduled**

7.85.1 Detailed Description

Definition at line 46 of file response_handler.h.

7.85.2 Constructor & Destructor Documentation

7.85.2.1 ResponseHandler::ResponseHandler ()

Definition at line 34 of file response_handler.cc.

References `bufferFull`, `RESPONSE_BUFFER_SIZE`, `responseBuffer`, `serviced`, and `stoppedQueue`.

7.85.2.2 ResponseHandler::~ResponseHandler ()

Definition at line 52 of file response_handler.cc.

7.85.2.3 ResponseHandler::ResponseHandler ()

7.85.2.4 ResponseHandler::~ResponseHandler ()

7.85.3 Member Function Documentation

7.85.3.1 bool ResponseHandler::CanStart ()

7.85.3.2 bool ResponseHandler::CanStart ()

Definition at line 243 of file response_handler.cc.

References RESPONSE_BUFFER_SIZE, and responseBuffer.

7.85.3.3 bool ResponseHandler::IsBufferFull ()

7.85.3.4 bool ResponseHandler::IsBufferFull ()

Definition at line 229 of file response_handler.cc.

References bufferFull, RESPONSE_BUFFER_SIZE, and responseBuffer.

7.85.3.5 void ResponseHandler::process_event (IrisEvent * e)

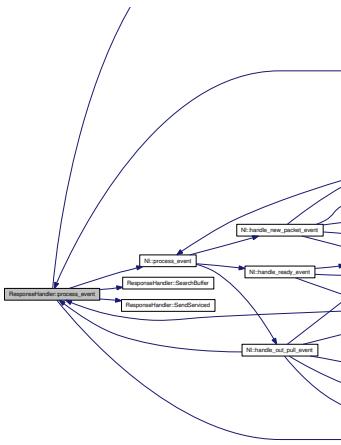
7.85.3.6 void ResponseHandler::process_event (IrisEvent * e)

Definition at line 64 of file response_handler.cc.

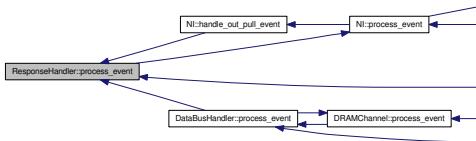
References Request::address, Request::arrivalTime, Request::busInsertionTime, Request::cmdType, IrisEvent::dst, IrisEvent::event_data, mc, Simulator::Now(), OUT_PULL_EVENT, parent, NI::process_event(), PUSH_BUFFER, REPLY, DRAMCmdState::req, responseBuffer, Request::retireTime, Simulator::Schedule(), SearchBuffer(), SEND_TO_NI, SendServiced(), serviced, Request::tag, and IrisEvent::type.

Referenced by NI::handle_out_pull_event(), DataBusHandler::process_event(), and BankHandler::process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



7.85.3.7 `unsigned int ResponseHandler::SearchBuffer (DRAMCmdState * cmd)`

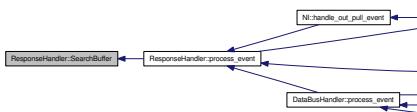
7.85.3.8 `unsigned int ResponseHandler::SearchBuffer (DRAMCmdState * cmd)`

Definition at line 216 of file response_handler.cc.

References DRAMCmdState::req, responseBuffer, and Request::tag.

Referenced by process_event().

Here is the caller graph for this function:



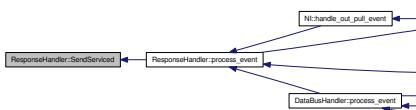
7.85.3.9 unsigned int ResponseHandler::SendServiced ()

Definition at line 203 of file response_handler.cc.

References responseBuffer, and serviced.

Referenced by process_event().

Here is the caller graph for this function:

**7.85.3.10 std::string ResponseHandler::toString ()****7.85.3.11 std::string ResponseHandler::toString (void)**

Definition at line 251 of file response_handler.cc.

7.85.4 Member Data Documentation**7.85.4.1 bool ResponseHandler::bufferFull**

Definition at line 55 of file response_handler.h.

Referenced by IsBufferFull(), and ResponseHandler().

7.85.4.2 Component * ResponseHandler::child

Definition at line 53 of file response_handler.h.

Referenced by MC::Init().

7.85.4.3 Component* ResponseHandler::mc

Definition at line 51 of file response_handler.h.

Referenced by MC::Init(), and process_event().

7.85.4.4 bool ResponseHandler::out_pull_scheduled [private]

Definition at line 69 of file response_handler.h.

7.85.4.5 Component * ResponseHandler::parent

Definition at line 52 of file response_handler.h.

Referenced by MC::Init(), and process_event().

7.85.4.6 Component * ResponseHandler::reqPtr

Definition at line 58 of file response_handler.h.

Referenced by MC::Init().

7.85.4.7 `vector< Request > ResponseHandler::responseBuffer`

Definition at line 54 of file response_handler.h.

Referenced by CanStart(), IsBufferFull(), process_event(), ResponseHandler(), SearchBuffer(), and SendServiced().

7.85.4.8 `bool ResponseHandler::serviced[RESPONSE_BUFFER_SIZE]`

Definition at line 56 of file response_handler.h.

Referenced by process_event(), ResponseHandler(), and SendServiced().

7.85.4.9 `bool ResponseHandler::stoppedQueue`

Definition at line 59 of file response_handler.h.

Referenced by ResponseHandler().

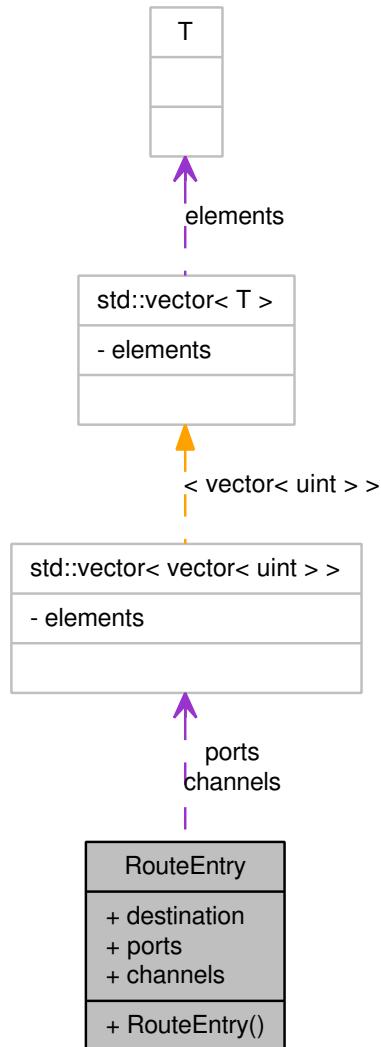
The documentation for this class was generated from the following files:

- `response_handler.h`
- `response_handler_temp.h`
- `response_handler.cc`
- `response_handler_temp.cc`

7.86 RouteEntry Class Reference

```
#include <genericData.h>
```

Collaboration diagram for RouteEntry:



Public Member Functions

- **RouteEntry ()**

Public Attributes

- **uint destination**
- **vector< vector< uint >> ports**
- **vector< vector< uint >> channels**

7.86.1 Detailed Description

Definition at line 99 of file genericData.h.

7.86.2 Constructor & Destructor Documentation

7.86.2.1 RouteEntry::RouteEntry ()

7.86.3 Member Data Documentation

7.86.3.1 `vector< vector<uint> > RouteEntry::channels`

Definition at line 105 of file genericData.h.

7.86.3.2 `uint RouteEntry::destination`

Definition at line 103 of file genericData.h.

7.86.3.3 `vector< vector<uint> > RouteEntry::ports`

Definition at line 104 of file genericData.h.

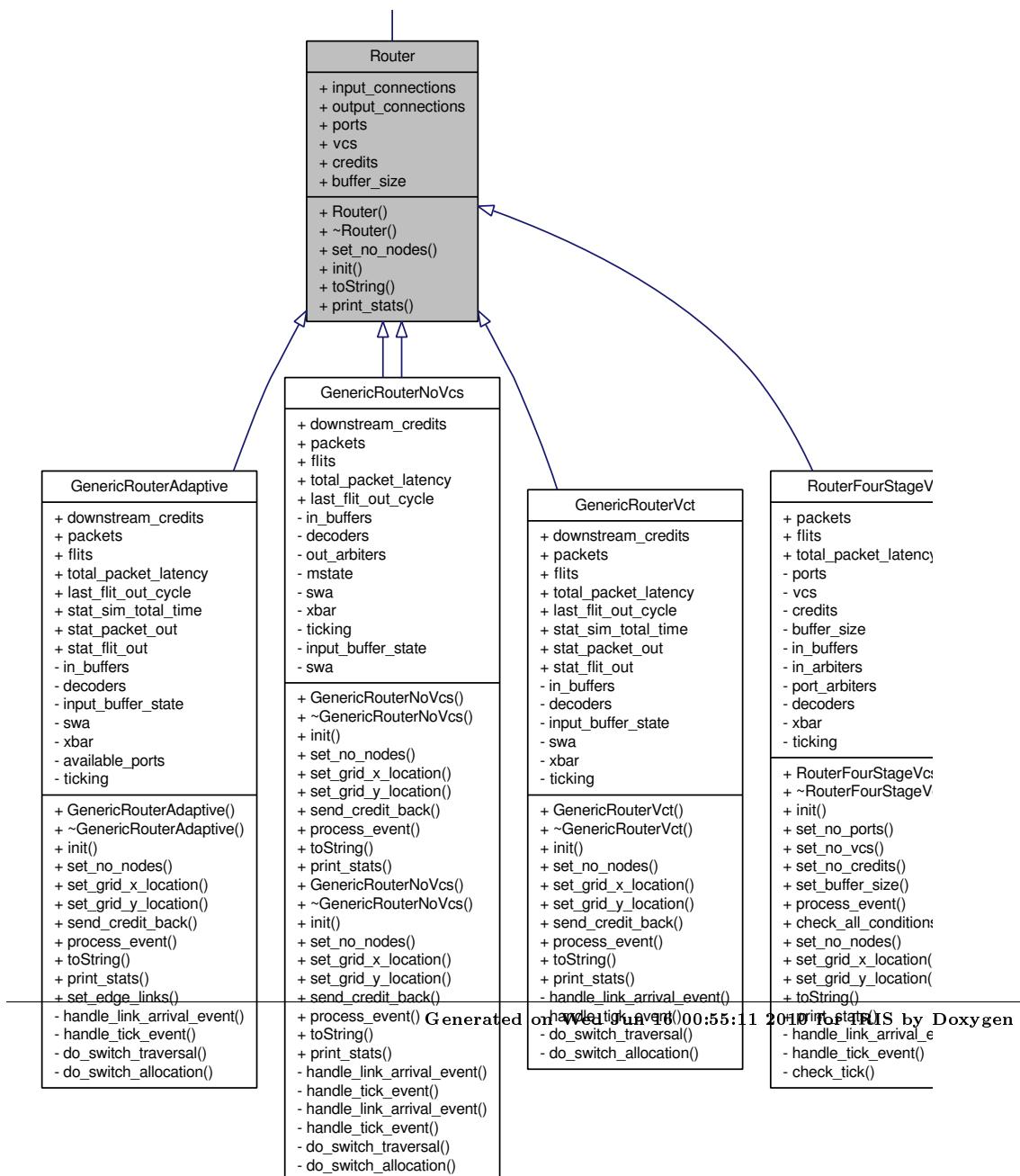
The documentation for this class was generated from the following file:

– `genericData.h`

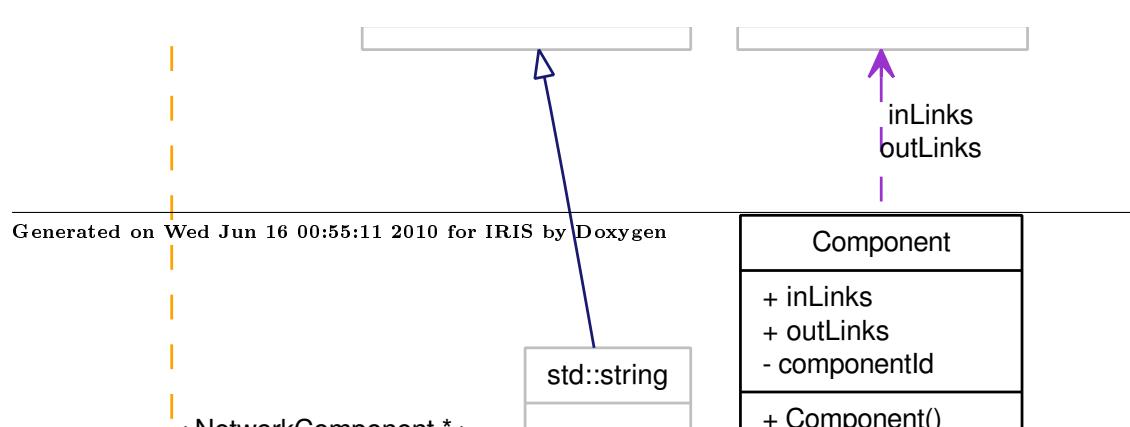
7.87 Router Class Reference

```
#include <router.h>
```

Inheritance diagram for Router:



Collaboration diagram for Router:



Public Member Functions

- **Router ()**
- **~Router ()**
- virtual void **set_no_nodes** (unsigned int nodes)=0
- virtual void **init**(**uint** p, **uint** v, **uint** c, **uint** b)=0
- virtual string **toString** () const
- virtual string **print_stats** ()=0

Public Attributes

- vector< NetworkComponent * > **input_connections**
- vector< NetworkComponent * > **output_connections**
- **uint ports**
- **uint vcs**
- **uint credits**
- **uint buffer_size**

7.87.1 Detailed Description

Definition at line 31 of file router.h.

7.87.2 Constructor & Destructor Documentation

7.87.2.1 Router::Router () [inline]

Definition at line 34 of file router.h.

References NetworkComponent::router, and NetworkComponent::type.

7.87.2.2 Router::~Router () [inline]

Definition at line 37 of file router.h.

7.87.3 Member Function Documentation

7.87.3.1 virtual void Router::init (**uint** p, **uint** v, **uint** c, **uint** b) [pure virtual]

Implemented in **GenericRouterNoVcs** (p. 166), **GenericRouterAdaptive** (p. 158), **GenericRouterNoVcs** (p. 166), and **GenericRouterVct** (p. 174).

7.87.3.2 virtual string Router::print_stats () [pure virtual]

Implemented in **GenericRouterNoVcs** (p. 167), **RouterFourStageVcs** (p. 333), **GenericRouterAdaptive** (p. 158), **GenericRouterNoVcs** (p. 167), and **GenericRouterVct** (p. 174).

7.87.3.3 virtual void Router::set_no_nodes (unsigned int *nodes*) [pure virtual]

Implemented in **GenericRouterNoVcs** (p. 168), **RouterFourStageVcs** (p. 334), **GenericRouterAdaptive** (p. 159), **GenericRouterNoVcs** (p. 168), and **GenericRouterVct** (p. 175).

7.87.3.4 string Router::toString () const [virtual]

Reimplemented from **NetworkComponent** (p. 275).

Reimplemented in **GenericRouterNoVcs** (p. 168), **RouterFourStageVcs** (p. 334), **GenericRouterAdaptive** (p. 160), **GenericRouterNoVcs** (p. 168), and **GenericRouterVct** (p. 175).

Definition at line 26 of file router.cc.

7.87.4 Member Data Documentation**7.87.4.1 uint Router::buffer_size**

Reimplemented in **RouterFourStageVcs** (p. 334).

Definition at line 48 of file router.h.

Referenced by **GenericRouterVct::init()**, **GenericRouterNoVcs::init()**, and **GenericRouterAdaptive::init()**.

7.87.4.2 uint Router::credits

Reimplemented in **RouterFourStageVcs** (p. 335).

Definition at line 47 of file router.h.

Referenced by **GenericRouterVct::handle_tick_event()**, **GenericRouterAdaptive::handle_tick_event()**, **GenericRouterVct::init()**, **GenericRouterNoVcs::init()**, and **GenericRouterAdaptive::init()**.

7.87.4.3 vector<NetworkComponent* > Router::input_connections

Definition at line 38 of file router.h.

Referenced by **RouterFourStageVcs::handle_link_arrival_event()**, **GenericRouterVct::handle_link_arrival_event()**, **GenericRouterNoVcs::handle_link_arrival_event()**, **GenericRouterAdaptive::handle_link_arrival_event()**, **RouterFourStageVcs::handle_tick_event()**, **GenericRouterVct::send_credit_back()**, **GenericRouterNoVcs::send_credit_back()**, **GenericRouterAdaptive::send_credit_back()**, and **GenericRouterAdaptive::set_edge_links()**.

7.87.4.4 vector<NetworkComponent* > Router::output_connections

Definition at line 39 of file router.h.

Referenced by **GenericRouterVct::do_switch_traversal()**, **GenericRouterNoVcs::do_switch_traversal()**, **GenericRouterAdaptive::do_switch_traversal()**, **RouterFourStageVcs::handle_link_arrival_event()**, **GenericRouterVct::handle_link_arrival_event()**, **GenericRouterNoVcs::handle_link_arrival_event()**, **GenericRouterAdaptive::handle_link_arrival_event()**, **RouterFourStageVcs::handle_tick_event()**, and **GenericRouterNoVcs::handle_tick_event()**.

7.87.4.5 uint Router::ports

Reimplemented in **RouterFourStageVcs** (p. 335).

Definition at line 45 of file router.h.

Referenced by GenericRouterVct::do_switch_allocation(), GenericRouterAdaptive::do_switch_allocation(), GenericRouterVct::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericRouterVct::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericRouterAdaptive::handle_tick_event(), GenericRouterVct::init(), GenericRouterNoVcs::init(), GenericRouterAdaptive::init(), GenericRouterVct::print_stats(), GenericRouterAdaptive::print_stats(), and GenericRouterAdaptive::set_edge_links().

7.87.4.6 uint Router::vcs

Reimplemented in **RouterFourStageVcs** (p. 336).

Definition at line 46 of file router.h.

Referenced by GenericRouterVct::do_switch_allocation(), GenericRouterAdaptive::do_switch_allocation(), GenericRouterVct::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericRouterVct::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericRouterAdaptive::handle_tick_event(), GenericRouterVct::init(), GenericRouterNoVcs::init(), and GenericRouterAdaptive::init().

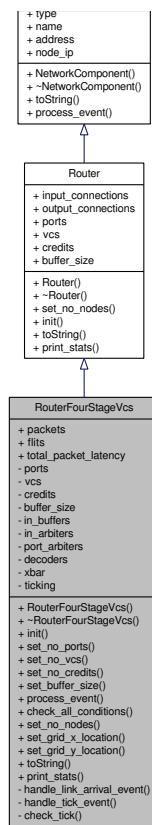
The documentation for this class was generated from the following files:

- **router.h**
- **router.cc**

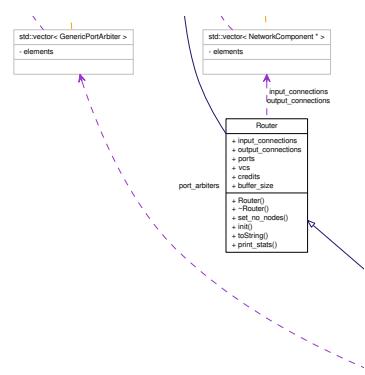
7.88 RouterFourStageVcs Class Reference

```
#include <routerFourStageVcs.h>
```

Inheritance diagram for RouterFourStageVcs:



Collaboration diagram for RouterFourStageVcs:



Public Member Functions

- **RouterFourStageVcs ()**
- **~RouterFourStageVcs ()**
- **void init ()**
- **void set_no_ports (uint p)**
- **void set_no_vcs (uint v)**
- **void set_no_credits (int c)**
- **void set_buffer_size (uint b)**
- **void process_event (IrisEvent *e)**
- **uint check_all_conditions ()**
- **void set_no_nodes (uint nodes)**
- **void set_grid_x_location (uint a, uint b, uint c)**
- **void set_grid_y_location (uint a, uint b, uint c)**
- **string toString () const**
- **string print_stats ()**

Public Attributes

- **uint packets**
- **uint flits**
- **double total_packet_latency**

Private Member Functions

- **void handle_link_arrival_event (IrisEvent *e)**
- **void handle_tick_event (IrisEvent *e)**
- **bool check_tick ()**

Private Attributes

- **uint ports**
- **uint vcs**
- **int credits**
- **uint buffer_size**
- **vector< GenericOutputBuffer > in_buffers**
- **vector< GenericArbiter > in_arbiters**
- **vector< GenericPortArbiter > port_arbiters**
- **vector< GenericAddressDecoder > decoders**
- **GenericCrossbar xbar**
- **bool ticking**

7.88.1 Detailed Description

Definition at line 38 of file routerFourStageVcs.h.

7.88.2 Constructor & Destructor Documentation

7.88.2.1 RouterFourStageVcs::RouterFourStageVcs ()

Definition at line 50 of file routerFourStageVcs.cc.

References NetworkComponent::name, and ticking.

7.88.2.2 RouterFourStageVcs::~RouterFourStageVcs ()

Definition at line 56 of file routerFourStageVcs.cc.

7.88.3 Member Function Documentation

7.88.3.1 uint RouterFourStageVcs::check_all_conditions ()

7.88.3.2 bool RouterFourStageVcs::check_tick () [private]

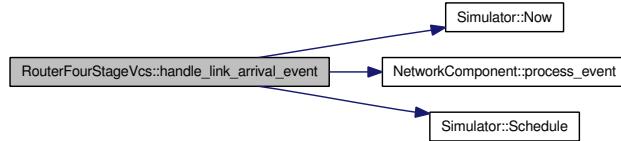
7.88.3.3 void RouterFourStageVcs::handle_link_arrival_event (IrisEvent * e) [private]

Definition at line 190 of file routerFourStageVcs.cc.

References _DBG, CREDIT_ID, decoders, IrisEvent::event_data, FLIT_ID, flits, in_buffers, Router::input_connections, Simulator::Now(), Router::output_connections, packets, ports, NetworkComponent::process_event(), LinkArrivalData::ptr, Simulator::Schedule(), IrisEvent::src_id, TAIL, TICK_EVENT, ticking, Flit::type, LinkArrivalData::type, IrisEvent::vc, and LinkArrivalData::vc.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:



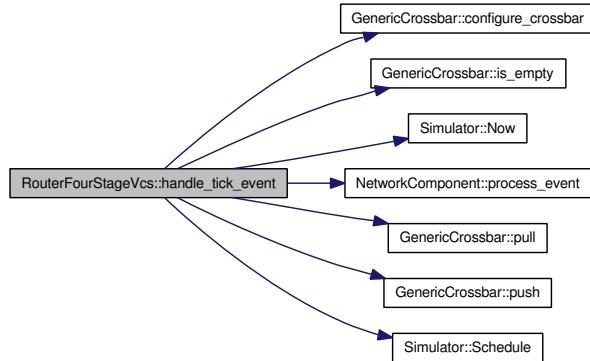
7.88.3.4 void RouterFourStageVcs::handle_tick_event (IrisEvent * e) [private]

Definition at line 280 of file routerFourStageVcs.cc.

References _DBG, NetworkComponent::address, GenericCrossbar::configure_crossbar(), CREDIT_ID, decoders, FLIT_ID, in_arbiters, in_buffers, Router::input_connections, GenericCrossbar::is_empty(), LINK_ARRIVAL_EVENT, Simulator::Now(), Router::output_connections, port_arbiters, ports, NetworkComponent::process_event(), LinkArrivalData::ptr, GenericCrossbar::pull(), GenericCrossbar::push(), Simulator::Schedule(), TAIL, TICK_EVENT, ticking, total_packet_latency, Flit::type, LinkArrivalData::type, IrisEvent::vc, LinkArrivalData::vc, vcs, and xbar.

Referenced by process_event().

Here is the call graph for this function:



Here is the caller graph for this function:

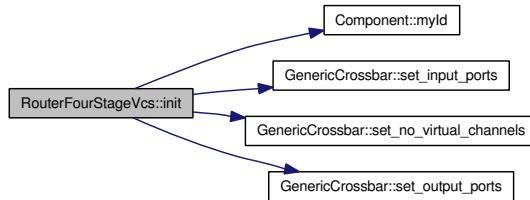


7.88.3.5 void RouterFourStageVcs::init ()

Definition at line 84 of file routerFourStageVcs.cc.

References NetworkComponent::address, buffer_size, credits, decoders, flits, in_arbiters, in_buffers, Component::myId(), NetworkComponent::node_ip, packets, port_arbiters, ports, GenericCrossbar::set_input_ports(), GenericCrossbar::set_no_virtual_channels(), GenericCrossbar::set_output_ports(), total_packet_latency, vcs, and xbar.

Here is the call graph for this function:



7.88.3.6 string RouterFourStageVcs::print_stats () [virtual]

Implements **Router** (p. 329).

Definition at line 176 of file routerFourStageVcs.cc.

References flits, NetworkComponent::node_ip, packets, and total_packet_latency.

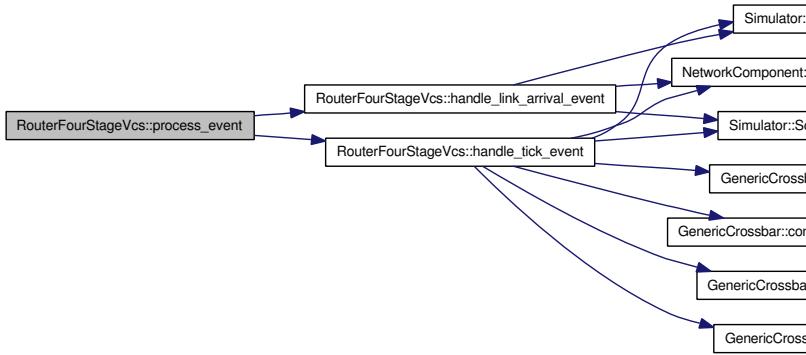
7.88.3.7 void RouterFourStageVcs::process_event (IrisEvent * e) [virtual]

Implements **NetworkComponent** (p. 274).

Definition at line 158 of file routerFourStageVcs.cc.

References __DBG, handle_link_arrival_event(), handle_tick_event(), LINK_ARRIVAL_EVENT, TICK_EVENT, and IrisEvent::type.

Here is the call graph for this function:



7.88.3.8 void RouterFourStageVcs::set_buffer_size (uint b)

Definition at line 79 of file routerFourStageVcs.cc.

References buffer_size.

7.88.3.9 void RouterFourStageVcs::set_grid_x_location (uint a, uint b, uint c)

Definition at line 144 of file routerFourStageVcs.cc.

References decoders.

7.88.3.10 void RouterFourStageVcs::set_grid_y_location (uint a, uint b, uint c)

Definition at line 150 of file routerFourStageVcs.cc.

References decoders.

7.88.3.11 void RouterFourStageVcs::set_no_credits (int c)

Definition at line 73 of file routerFourStageVcs.cc.

References credits.

7.88.3.12 void RouterFourStageVcs::set_no_nodes (uint nodes) [virtual]

Implements **Router** (p. 329).

Definition at line 134 of file routerFourStageVcs.cc.

References decoders.

7.88.3.13 void RouterFourStageVcs::set_no_ports (uint p)

Definition at line 61 of file routerFourStageVcs.cc.

References ports.

7.88.3.14 void RouterFourStageVcs::set_no_vcs (uint v)

Definition at line 67 of file routerFourStageVcs.cc.

References vcs.

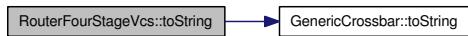
7.88.3.15 string RouterFourStageVcs::toString () const [virtual]

Reimplemented from **Router** (p. 329).

Definition at line 404 of file routerFourStageVcs.cc.

References NetworkComponent::address, decoders, in_arbiters, in_buffers, NetworkComponent::node_ip, port_arbiters, GenericCrossbar::toString(), and xbar.

Here is the call graph for this function:

**7.88.4 Member Data Documentation****7.88.4.1 uint RouterFourStageVcs::buffer_size [private]**

Reimplemented from **Router** (p. 329).

Definition at line 67 of file routerFourStageVcs.h.

Referenced by init(), and set_buffer_size().

7.88.4.2 int RouterFourStageVcs::credits [private]

Reimplemented from **Router** (p. 329).

Definition at line 66 of file routerFourStageVcs.h.

Referenced by init(), and set_no_credits().

7.88.4.3 vector<GenericAddressDecoder> RouterFourStageVcs::decoders [private]

Definition at line 72 of file routerFourStageVcs.h.

Referenced by handle_link_arrival_event(), handle_tick_event(), init(), set_grid_x_location(), set_grid_y_location(), set_no_nodes(), and toString().

7.88.4.4 uint RouterFourStageVcs::flits

Definition at line 58 of file routerFourStageVcs.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.88.4.5 vector<GenericArbiter> RouterFourStageVcs::in_arbiters [private]

Definition at line 70 of file routerFourStageVcs.h.

Referenced by handle_tick_event(), init(), and toString().

7.88.4.6 `vector<GenericOutputBuffer>` RouterFourStageVcs::in_buffers [private]

Definition at line 69 of file routerFourStageVcs.h.

Referenced by handle_link_arrival_event(), handle_tick_event(), init(), and toString().

7.88.4.7 `uint` RouterFourStageVcs::packets

Definition at line 57 of file routerFourStageVcs.h.

Referenced by handle_link_arrival_event(), init(), and print_stats().

7.88.4.8 `vector<GenericPortArbiter>` RouterFourStageVcs::port_arbiters [private]

Definition at line 71 of file routerFourStageVcs.h.

Referenced by handle_tick_event(), init(), and toString().

7.88.4.9 `uint` RouterFourStageVcs::ports [private]

Reimplemented from **Router** (p. 330).

Definition at line 64 of file routerFourStageVcs.h.

Referenced by handle_link_arrival_event(), handle_tick_event(), init(), and set_no_ports().

7.88.4.10 `bool` RouterFourStageVcs::ticking [private]

Definition at line 75 of file routerFourStageVcs.h.

Referenced by handle_link_arrival_event(), handle_tick_event(), and RouterFourStageVcs().

7.88.4.11 `double` RouterFourStageVcs::total_packet_latency

Definition at line 59 of file routerFourStageVcs.h.

Referenced by handle_tick_event(), init(), and print_stats().

7.88.4.12 `uint` RouterFourStageVcs::vcs [private]

Reimplemented from **Router** (p. 330).

Definition at line 65 of file routerFourStageVcs.h.

Referenced by handle_tick_event(), init(), and set_no_vcs().

7.88.4.13 GenericCrossbar RouterFourStageVcs::xbar [private]

Definition at line 73 of file routerFourStageVcs.h.

Referenced by handle_tick_event(), init(), and toString().

The documentation for this class was generated from the following files:

- `routerFourStageVcs.h`
- `routerFourStageVcs.cc`

7.89 SA_unit Class Reference

```
#include <genericData.h>
```

Public Member Functions

- `SA_unit ()`

Public Attributes

- `uint port`
- `uint ch`
- `ullint in_time`

7.89.1 Detailed Description

Definition at line 113 of file genericData.h.

7.89.2 Constructor & Destructor Documentation

7.89.2.1 `SA_unit::SA_unit () [inline]`

Definition at line 116 of file genericData.h.

7.89.3 Member Data Documentation

7.89.3.1 `uint SA_unit::ch`

Definition at line 118 of file genericData.h.

7.89.3.2 `ullint SA_unit::in_time`

Definition at line 119 of file genericData.h.

7.89.3.3 `uint SA_unit::port`

Definition at line 116 of file genericData.h.

Referenced by `GenericRouterVct::do_switch_allocation()`, `GenericRouterNoVcs::do_switch_allocation()`, and `GenericRouterAdaptive::do_switch_allocation()`.

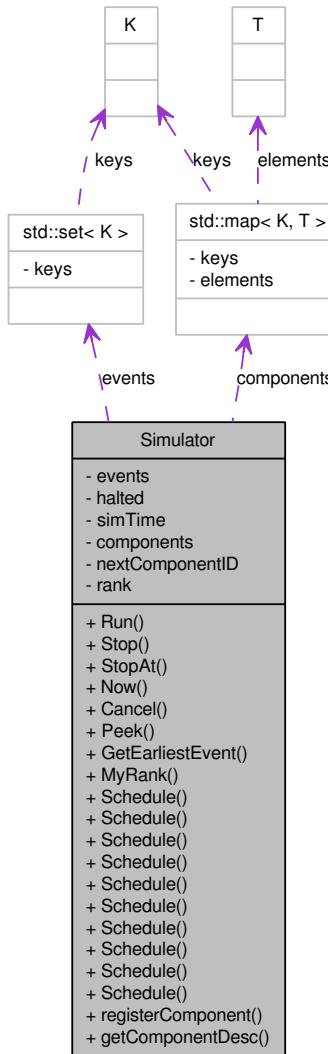
The documentation for this class was generated from the following file:

- `genericData.h`

7.90 Simulator Class Reference

```
#include <simulator.h>
```

Collaboration diagram for Simulator:



Static Public Member Functions

- static void **Run** ()
- static void **Stop** ()
- static void **StopAt** (double)
- static double **Now** ()
- static bool **Cancel** (**EventId** &)
- static **EventId** **Peek** ()
- static **EventBase** * **GetEarliestEvent** ()
- static int **MyRank** ()
- template<typename T , typename OBJ >
 static **EventId** **Schedule** (double t, void(T::*handler)(void), OBJ *obj)
- template<typename T , typename OBJ , typename U1 , typename T1 >
 static **EventId** **Schedule** (double t, void(T::*handler)(U1), OBJ *obj, T1 t1)

- template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 >
 - static **EventId Schedule** (double t, void(T::*handler)(U1, U2), OBJ *obj, T1 t1, T2 t2)
- template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 >
 - static **EventId Schedule** (double t, void(T::*handler)(U1, U2, U3), OBJ *obj, T1 t1, T2 t2, T3 t3)
- template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 , typename U4 , typename T4 >
 - static **EventId Schedule** (double t, void(T::*handler)(U1, U2, U3, U4), OBJ *obj, T1 t1, T2 t2, T3 t3, T4 t4)
- static **EventId Schedule** (double t, void(*handler)(void))
- template<typename U1 , typename T1 >
 - static **EventId Schedule** (double t, void(*handler)(U1), T1 t1)
- template<typename U1 , typename T1 , typename U2 , typename T2 >
 - static **EventId Schedule** (double t, void(*handler)(U1, U2), T1 t1, T2 t2)
- template<typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 >
 - static **EventId Schedule** (double t, void(*handler)(U1, U2, U3), T1 t1, T2 t2, T3 t3)
- template<typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 , typename U4 , typename T4 >
 - static **EventId Schedule** (double t, void(*handler)(U1, U2, U3, U4), T1 t1, T2 t2, T3 t3, T4 t4)
- static void **registerComponent** (Component *obj, int lp)
- static ComponentDescription * **getComponentDesc** (int)

Static Private Attributes

- static **EventSet_t events**
- static bool **halted** = false
- static double **simTime**
- static **ComponentMap_t components**
- static int **nextComponentID**
- static int **rank**

7.90.1 Detailed Description

Definition at line 296 of file simulator.h.

7.90.2 Member Function Documentation

7.90.2.1 **bool Simulator::Cancel (EventId & evid) [static]**

Definition at line 56 of file simulator.cc.

References events.

7.90.2.2 **ComponentDescription * Simulator::getComponentDesc (int compId) [static]**

Definition at line 101 of file simulator.cc.

References components.

Referenced by Link::Link(), and Link::Send().

Here is the caller graph for this function:



7.90.2.3 EventBase * Simulator::GetEarliestEvent () [static]

Definition at line 71 of file simulator.cc.

References events.

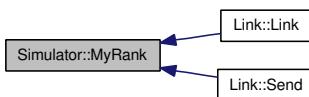
7.90.2.4 int Simulator::MyRank () [static]

Definition at line 78 of file simulator.cc.

References rank.

Referenced by Link::Link(), and Link::Send().

Here is the caller graph for this function:



7.90.2.5 double Simulator::Now () [static]

Definition at line 51 of file simulator.cc.

References simTime.

Referenced by CmdIssuer::BankNotBusy(), CmdIssuer::BusNotBusy(), Statistic::CalculateAggregateStats(), CmdIssuer::CanSchedule(), Statistic::CollectStatsPerRequest(), MSHR_H::DeleteInMSHR(), MSHR_H::demap_addr(), PToPSwitchArbiter::do_fcfs_arbitration(), GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), GenericTPG::GetNewRequest(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_in_push_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericLink::handle_link_arrival_event(), NI::handle_new_packet_event(), GenericVcsInterface::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), GenericSink::handle_new_packet_event(), GenericRPG::handle_new_packet_event(), GenericInterface::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericSink::handle_outpull_event(), NI::handle_ready_event(), MCFrontEnd::handle_ready_event(), GenericVcsInterface::handle_ready_event(), GenericTPG::handle_ready_event(), GenericSink::handle_ready_event(), GenericRPG::handle_ready_event(), GenericInterface::handle_ready_event(),

```

GenericFlatMc::handle_ready_event(),
GenericVcsInterface::handle_tick_event(),
GenericRouterNoVcs::handle_tick_event(),
event(), GenericInterface::handle_tick_event(),
main(), Statistic::PrintAggregateStats(),
RequestHandler::process_event(),
RefreshMgr::process_event(),
MSHR_H::process_event(),
DRAMChannel::process_event(),
DataBusHandler::process_event(),
CmdIssuer::process_event(),
CmdBusHandler::process_event(),
BankHandler::process_event(),
AddrMap::process_event(),
PToPSwitchArbiter::request(),
Link::Send(),
GenericRouterVct::send_credit_back(),
GenericRouterAdaptive::send_credit_back(),
GenericRouterNoVcs::send_credit_back(),
CmdIssuer::SetBusBusyTime(),
CmdIssuer::SetPrevState(),
NI::setup(),
MCFrontEnd::setup(),
GenericTPG::setup(),
GenericSink::setup(),
GenericRPG::setup(),
GenericFlatMc::setup(),
MC::StartRefresh(),
TailFlit::TailFlit(),
timed_cout(),
and IrisEvent::toString().

```

7.90.2.6 EventId Simulator::Peek () [static]

Definition at line 64 of file simulator.cc.

References events.

7.90.2.7 void Simulator::registerComponent (Component * *obj*, int *lp*) [static]

Definition at line 85 of file simulator.cc.

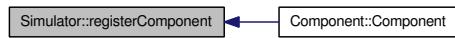
References components, nextComponentID, rank, and Component::setComponentId().

Referenced by Component::Component().

Here is the call graph for this function:



Here is the caller graph for this function:



7.90.2.8 void Simulator::Run () [static]

Definition at line 21 of file simulator.cc.

References EventBase::CallHandler(), events, halted, rank, simTime, and EventBase::time.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.90.2.9 template<typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 , typename U4 , typename T4 > static EventId Simulator::Schedule (double t, void(*)(U1, U2, U3, U4) *handler*, T1 *t1*, T2 *t2*, T3 *t3*, T4 *t4*) [inline, static]

Definition at line 397 of file simulator.h.

References events, and EventBase::uid.

7.90.2.10 template<typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 > static EventId Simulator::Schedule (double t, void(*)(U1, U2, U3) *handler*, T1 *t1*, T2 *t2*, T3 *t3*) [inline, static]

Definition at line 386 of file simulator.h.

References events, and EventBase::uid.

7.90.2.11 template<typename U1 , typename T1 , typename U2 , typename T2 > static EventId Simulator::Schedule (double t, void(*)(U1, U2) *handler*, T1 *t1*, T2 *t2*) [inline, static]

Definition at line 376 of file simulator.h.

References events, and EventBase::uid.

7.90.2.12 template<typename U1 , typename T1 > static EventId Simulator::Schedule (double t, void(*)(U1) *handler*, T1 *t1*) [inline, static]

Definition at line 367 of file simulator.h.

References events, and EventBase::uid.

7.90.2.13 static EventId Simulator::Schedule (double t, void(*)(void) *handler*) [inline, static]

Definition at line 359 of file simulator.h.

References events, and EventBase::uid.

7.90.2.14 template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 , typename U4 , typename T4 > static EventId Simulator::Schedule (double t, void(T::*)(U1, U2, U3, U4) *handler*, OBJ * *obj*, T1 *t1*, T2 *t2*, T3 *t3*, T4 *t4*) [inline, static]

Definition at line 351 of file simulator.h.

References events, and EventBase::uid.

7.90.2.15 template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 , typename U3 , typename T3 > static EventId Simulator::Schedule (double t, void(T::*)(U1, U2, U3) *handler*, OBJ * *obj*, T1 *t1*, T2 *t2*, T3 *t3*) [inline, static]

Definition at line 339 of file simulator.h.

References events, and EventBase::uid.

7.90.2.16 template<typename T , typename OBJ , typename U1 , typename T1 , typename U2 , typename T2 > static EventId Simulator::Schedule (double t, void(T::*)(U1, U2) *handler*, OBJ * *obj*, T1 *t1*, T2 *t2*) [inline, static]

Definition at line 328 of file simulator.h.

References events, and EventBase::uid.

7.90.2.17 template<typename T , typename OBJ , typename U1 , typename T1 > static EventId Simulator::Schedule (double t, void(T::*)(U1) *handler*, OBJ * *obj*, T1 *t1*) [inline, static]

Definition at line 318 of file simulator.h.

References events, and EventBase::uid.

7.90.2.18 template<typename T , typename OBJ > static EventId Simulator::Schedule (double t, void(T::*)(void) *handler*, OBJ * *obj*) [inline, static]

Definition at line 309 of file simulator.h.

References events, and EventBase::uid.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_in_push_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericLink::handle_link_arrival_event(), NI::handle_new_packet_event(), GenericVcsInterface::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), GenericSink::handle_new_packet_event(), GenericRPG::handle_new_packet_event(), GenericInterface::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), NI::handle_ready_event(), MCFrontEnd::handle_ready_event(), GenericVcsInterface::handle_ready_event(), GenericTPG::handle_ready_event(), GenericSink::handle_ready_event(), GenericRPG::handle_ready_event(), GenericInterface::handle_ready_event(), GenericFlatMc::handle_ready_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterVct::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericRouterAdaptive::handle_tick_event(), GenericInterface::handle_tick_event(), main(), ResponseHandler::process_event(), RequestHandler::process_event(), RefreshMgr::process_event(), MSHR_H::process_event(), DRAMChannel::process_event(), DataBusHandler::process_event(), CmdIssuer::process_event(), CmdBusHandler::process_event(), BankHandler::process_event(), AddrMap::process_event(), Link::Send(), GenericRouterVct::send_credit_back(), GenericRouterNoVcs::send_credit_back(), GenericRouterAdaptive::send_credit_back(), NI::setup(), MCFrontEnd::setup(), GenericTPG::setup(), GenericSink::setup(), GenericRPG::setup(), GenericFlatMc::setup(), MC::StartRefresh(), and StopAt().

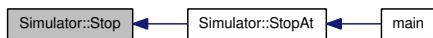
7.90.2.19 void Simulator::Stop () [static]

Definition at line 41 of file simulator.cc.

References halted.

Referenced by StopAt().

Here is the caller graph for this function:



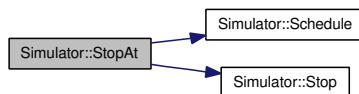
7.90.2.20 void Simulator::StopAt (double stopTime) [static]

Definition at line 46 of file simulator.cc.

References Schedule(), and Stop().

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.90.3 Member Data Documentation

7.90.3.1 ComponentMap_t Simulator::components [static, private]

Definition at line 411 of file simulator.h.

Referenced by getComponentDesc(), and registerComponent().

7.90.3.2 EventSet_t Simulator::events [static, private]

Definition at line 408 of file simulator.h.

Referenced by Cancel(), GetEarliestEvent(), Peek(), Run(), and Schedule().

7.90.3.3 bool Simulator::halted = false [static, private]

Definition at line 409 of file simulator.h.

Referenced by main(), Run(), and Stop().

7.90.3.4 int Simulator::nextComponentID [static, private]

Definition at line 412 of file simulator.h.

Referenced by registerComponent().

7.90.3.5 int Simulator::rank [static, private]

Definition at line 413 of file simulator.h.

Referenced by MyRank(), registerComponent(), and Run().

7.90.3.6 double Simulator::simTime [static, private]

Definition at line 410 of file simulator.h.

Referenced by Now(), and Run().

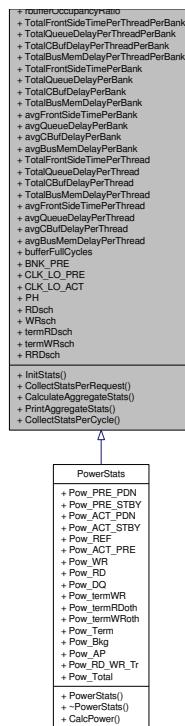
The documentation for this class was generated from the following files:

- **simulator.h**
- **simulator.cc**

7.91 Statistic Class Reference

```
#include <stats.h>
```

Inheritance diagram for Statistic:



Collaboration diagram for Statistic:



Public Member Functions

- void **InitStats** ()
 - void **CollectStatsPerRequest** (Request *req)
 - void **CalculateAggregateStats** ()
 - string **PrintAggregateStats** (uint n)
 - void **CollectStatsPerCycle** ()

Public Attributes

- **Component** * mc
 - **Component** * pwr
 - bool * doneOnce [NO_OF_THREADS]
 - double avgLatPerThread [NO_OF_THREADS]
 - unsigned long long int latPerThread [NO_OF_THREADS]
 - unsigned long long int reqPerThread [NO_OF_THREADS]
 - unsigned long long int hitsPerThread [NO_OF_THREADS]

- unsigned long long int **throttlePerThread** [NO_OF_THREADS]
- double **avgThrottlePerThread** [NO_OF_THREADS]
- double **hitRatePerThread** [NO_OF_THREADS]
- float **BLPPerThread** [NO_OF_THREADS]
- double **avgLatPerBank** [NO_OF_BANKS]
- unsigned long long int **latPerBank** [NO_OF_BANKS]
- unsigned long long int **reqPerBank** [NO_OF_BANKS]
- unsigned long long int **hitsPerBank** [NO_OF_BANKS]
- unsigned long long int **throttlePerBank** [NO_OF_BANKS]
- double **avgThrottlePerBank** [NO_OF_BANKS]
- double **hitRatePerBank** [NO_OF_BANKS]
- double **avgLatPerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **latPerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **reqPerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **hitsPerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **throttlePerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- double **avgThrottlePerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- double **hitRatePerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **rbufferOccupancy** [NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]
- unsigned long long int **emptyCycles** [NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]
- float **rbufferOccupancyRatio** [NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]
- unsigned long long int **TotalFrontSideTimePerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **TotalQueueDelayPerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **TotalCBufDelayPerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **TotalBusMemDelayPerThreadPerBank** [NO_OF_THREADS][NO_OF_BANKS]
- unsigned long long int **TotalFrontSideTimePerBank** [NO_OF_BANKS]
- unsigned long long int **TotalQueueDelayPerBank** [NO_OF_BANKS]
- unsigned long long int **TotalCBufDelayPerBank** [NO_OF_BANKS]
- unsigned long long int **TotalBusMemDelayPerBank** [NO_OF_BANKS]
- float **avgFrontSideTimePerBank** [NO_OF_BANKS]
- float **avgQueueDelayPerBank** [NO_OF_BANKS]
- float **avgCBufDelayPerBank** [NO_OF_BANKS]
- float **avgBusMemDelayPerBank** [NO_OF_BANKS]
- unsigned long long int **TotalFrontSideTimePerThread** [NO_OF_THREADS]
- unsigned long long int **TotalQueueDelayPerThread** [NO_OF_THREADS]
- unsigned long long int **TotalCBufDelayPerThread** [NO_OF_THREADS]
- unsigned long long int **TotalBusMemDelayPerThread** [NO_OF_THREADS]
- float **avgFrontSideTimePerThread** [NO_OF_THREADS]
- float **avgQueueDelayPerThread** [NO_OF_THREADS]
- float **avgCBufDelayPerThread** [NO_OF_THREADS]
- float **avgBusMemDelayPerThread** [NO_OF_THREADS]
- unsigned long long int **bufferFullCycles** [NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]
- float **BNK_PRE**
- float **CLK_LO_PRE**
- float **CLK_LO_ACT**
- float **PH**

- float **RDsch**
- float **WRsch**
- float **termRDsch**
- float **termWRsch**
- float **RRDsch**

7.91.1 Detailed Description

Definition at line 29 of file stats.h.

7.91.2 Member Function Documentation

7.91.2.1 void Statistic::CalculateAggregateStats ()

Definition at line 247 of file stats.cc.

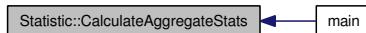
References avgBusMemDelayPerBank, avgBusMemDelayPerThread, avgCBufDelayPerBank, avgCBufDelayPerThread, avgFrontSideTimePerBank, avgFrontSideTimePerThread, avgLatPerBank, avgLatPerThread, avgLatPerThreadPerBank, avgQueueDelayPerBank, avgQueueDelayPerThread, avgThrottlePerBank, avgThrottlePerThread, avgThrottlePerThreadPerBank, BL, BNK_PRE, bufferFullCycles, CLK_LO_ACT, CLK_LO_PRE, hitRatePerBank, hitRatePerThread, hitsPerBank, hitsPerThread, hitsPerThreadPerBank, latPerBank, latPerThread, latPerThreadPerBank, mc, NO_OF_BANKS, NO_OF_CHANNELS, NO_OF_RANKS, NO_OF_THREADS, Simulator::Now(), PH, pwr, rbufferOccupancyRatio, RDsch, reqPerBank, reqPerThread, reqPerThreadPerBank, RRDsch, SysClk, termRDsch, termWRsch, throttlePerBank, throttlePerThread, throttlePerThreadPerBank, TotalBusMemDelayPerBank, TotalBusMemDelayPerThread, TotalBusMemDelayPerThreadPerBank, TotalCBufDelayPerBank, TotalCBufDelayPerThread, TotalCBufDelayPerThreadPerBank, TotalFrontSideTimePerBank, TotalFrontSideTimePerThread, TotalFrontSideTimePerThreadPerBank, TotalQueueDelayPerBank, TotalQueueDelayPerThread, TotalQueueDelayPerThreadPerBank, and WRsch.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



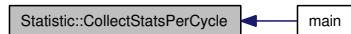
7.91.2.2 void Statistic::CollectStatsPerCycle ()

Definition at line 135 of file stats.cc.

References NO_OF_BUFFERS, NO_OF_CHANNELS, and NO_OF_RANKS.

Referenced by main().

Here is the caller graph for this function:



7.91.2.3 void Statistic::CollectStatsPerRequest (Request * req)

Definition at line 89 of file stats.cc.

References Request::address, Request::arrivalTime, avgLatPerThreadPerBank, avgThrottlePerThreadPerBank, Request::bankNo, Request::busInsertionTime, Request::cbufferInsertionTime, hitRatePerThreadPerBank, hitsPerThreadPerBank, latPerThreadPerBank, mc, Simulator::Now(), OPEN, Request::rbufferInsertionTime, reqPerThreadPerBank, Request::retireTime, Request::status, Request::threadId, Request::throttleTime, TotalBusMemDelayPerThreadPerBank, TotalCBufDelayPerThreadPerBank, TotalFrontSideTimePerThreadPerBank, and TotalQueueDelayPerThreadPerBank.

Here is the call graph for this function:



7.91.2.4 void Statistic::InitStats ()

Definition at line 152 of file stats.cc.

References avgBusMemDelayPerBank, avgBusMemDelayPerThread, avgCBufDelayPerBank, avgCBufDelayPerThread, avgFrontSideTimePerBank, avgFrontSideTimePerThread, avgLatPerBank, avgLatPerThread, avgLatPerThreadPerBank, avgQueueDelayPerBank, avgQueueDelayPerThread, avgThrottlePerBank, avgThrottlePerThread, avgThrottlePerThreadPerBank, BLPPerThread, BNK_PRE, CLK_LO_ACT, CLK_LO_PRE, emptyCycles, hitRatePerBank, hitRatePerThread, hitRatePerThreadPerBank, hitsPerBank, hitsPerThread, hitsPerThreadPerBank, latPerBank, latPerThread, latPerThreadPerBank, NO_OF_BANKS, NO_OF_BUFFERS, NO_OF_CHANNELS, NO_OF_RANKS, NO_OF_THREADS, PH, pwr, rbufferOccupancy, RDsch, reqPerBank, reqPerThread, reqPerThreadPerBank, termRDsch, termWRsch, throttlePerBank, throttlePerThread, throttlePerThreadPerBank, TotalBusMemDelayPerBank, TotalBusMemDelayPerThread, TotalBusMemDelayPerThreadPerBank, TotalCBufDelayPerBank, TotalCBufDelayPerThread, TotalCBufDelayPerThreadPerBank, TotalFrontSideTimePerBank, TotalFrontSideTimePerThread, TotalFrontSideTimePerThreadPerBank, TotalQueueDelayPerBank, TotalQueueDelayPerThread, TotalQueueDelayPerThreadPerBank, and WRsch.

Referenced by MC::Init().

Here is the caller graph for this function:



7.91.2.5 string Statistic::PrintAggregateStats (uint n)

Definition at line 348 of file stats.cc.

References avgBusMemDelayPerBank, avgBusMemDelayPerThread, avgCBufDelayPerBank, avgCBufDelayPerThread, avgFrontSideTimePerBank, avgFrontSideTimePerThread, avgLatPerBank, avgLatPerThread, avgQueueDelayPerBank, avgQueueDelayPerThread, bufferFullCycles, hitRatePerBank, hitRatePerThread, hitsPerBank, hitsPerThread, mc, NO_OF_BANKS, NO_OF_BUFFERS, NO_OF_CHANNELS, NO_OF_RANKS,

NO_OF_THREADS, Simulator::Now(), PH, pwr, rbufferOccupancyRatio, RDsch, reqPerBank, reqPerThread, and WRsch.

Referenced by main().

Here is the call graph for this function:



Here is the caller graph for this function:



7.91.3 Member Data Documentation

7.91.3.1 float Statistic::avgBusMemDelayPerBank[NO_OF_BANKS]

Definition at line 87 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.2 float Statistic::avgBusMemDelayPerThread[NO_OF_THREADS]

Definition at line 97 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.3 float Statistic::avgCBufDelayPerBank[NO_OF_BANKS]

Definition at line 86 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.4 float Statistic::avgCBufDelayPerThread[NO_OF_THREADS]

Definition at line 96 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.5 float Statistic::avgFrontSideTimePerBank[NO_OF_BANKS]

Definition at line 84 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.6 float Statistic::avgFrontSideTimePerThread[NO_OF_THREADS]

Definition at line 94 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.7 double Statistic::avgLatPerBank[NO_OF_BANKS]

Definition at line 47 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.8 double Statistic::avgLatPerThread[NO_OF_THREADS]

Definition at line 38 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.9 double Statistic::avgLatPerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 55 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.10 float Statistic::avgQueueDelayPerBank[NO_OF_BANKS]

Definition at line 85 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.11 float Statistic::avgQueueDelayPerThread[NO_OF_THREADS]

Definition at line 95 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.12 double Statistic::avgThrottlePerBank[NO_OF_BANKS]

Definition at line 52 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.13 double Statistic::avgThrottlePerThread[NO_OF_THREADS]

Definition at line 43 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.14 double Statistic::avgThrottlePerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 60 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.15 float Statistic::BLPPerThread[NO_OF_THREADS]

Definition at line 45 of file stats.h.

Referenced by InitStats().

7.91.3.16 float Statistic::BNK_PRE

Definition at line 102 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.17 unsigned long long int Statistic::bufferFullCycles[NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]

Definition at line 99 of file stats.h.

Referenced by CalculateAggregateStats(), and PrintAggregateStats().

7.91.3.18 float Statistic::CLK_LO_ACT

Definition at line 104 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.19 float Statistic::CLK_LO_PRE

Definition at line 103 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.20 bool* Statistic::doneOnce[NO_OF_THREADS]

Definition at line 37 of file stats.h.

Referenced by main().

7.91.3.21 unsigned long long int Statistic::emptyCycles[NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]

Definition at line 70 of file stats.h.

Referenced by InitStats().

7.91.3.22 double Statistic::hitRatePerBank[NO_OF_BANKS]

Definition at line 53 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.23 double Statistic::hitRatePerThread[NO_OF_THREADS]

Definition at line 44 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.24 double Statistic::hitRatePerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 61 of file stats.h.

Referenced by CollectStatsPerRequest(), and InitStats().

7.91.3.25 unsigned long long int Statistic::hitsPerBank[NO_OF_BANKS]

Definition at line 50 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.26 unsigned long long int Statistic::hitsPerThread[NO_OF_THREADS]

Definition at line 41 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.27 unsigned long long int Statistic::hitsPerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 58 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.28 unsigned long long int Statistic::latPerBank[NO_OF_BANKS]

Definition at line 48 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.29 unsigned long long int Statistic::latPerThread[NO_OF_THREADS]

Definition at line 39 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.30 unsigned long long int Statistic::latPerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 56 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.31 Component* Statistic::mc

Definition at line 35 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), MC::Init(), and PrintAggregateStats().

7.91.3.32 float Statistic::PH

Definition at line 105 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.33 Component* Statistic::pwr

Definition at line 36 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.34 unsigned long long int Statistic::rbufferOccupancy[NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]

Definition at line 69 of file stats.h.

Referenced by InitStats().

7.91.3.35 float Statistic::rbufferOccupancyRatio[NO_OF_CHANNELS][NO_OF_RANKS][NO_OF_BANKS]

Definition at line 71 of file stats.h.

Referenced by CalculateAggregateStats(), and PrintAggregateStats().

7.91.3.36 float Statistic::RDsch

Definition at line 106 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.37 unsigned long long int Statistic::reqPerBank[NO_OF_BANKS]

Definition at line 49 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.38 unsigned long long int Statistic::reqPerThread[NO_OF_THREADS]

Definition at line 40 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

7.91.3.39 unsigned long long int Statistic::reqPerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 57 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.40 float Statistic::RRDsch

Definition at line 110 of file stats.h.

Referenced by CalculateAggregateStats().

7.91.3.41 float Statistic::termRDsch

Definition at line 108 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.42 float Statistic::termWRsch

Definition at line 109 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.43 unsigned long long int Statistic::throttlePerBank[NO_OF_BANKS]

Definition at line 51 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.44 unsigned long long int Statistic::throttlePerThread[NO_OF_THREADS]

Definition at line 42 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.45 unsigned long long int Statistic::throttlePerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 59 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.46 unsigned long long int Statistic::TotalBusMemDelayPerBank[NO_OF_BANKS]

Definition at line 82 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.47 unsigned long long int Statistic::TotalBusMemDelayPerThread[NO_OF_THREADS]

Definition at line 92 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.48 unsigned long long int Statistic::TotalBusMemDelayPerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 77 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.49 unsigned long long int Statistic::TotalCBufDelayPerBank[NO_OF_BANKS]

Definition at line 81 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.50 unsigned long long int Statistic::TotalCBufDelayPerThread[NO_OF_THREADS]

Definition at line 91 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.51 unsigned long long int Statistic::TotalCBufDelayPerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 76 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.52 unsigned long long int Statistic::TotalFrontSideTimePerBank[NO_OF_BANKS]

Definition at line 79 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.53 unsigned long long int Statistic::TotalFrontSideTimePerThread[NO_OF_THREADS]

Definition at line 89 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.54 unsigned long long int Statistic::TotalFrontSideTimePerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 74 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.55 unsigned long long int Statistic::TotalQueueDelayPerBank[NO_OF_BANKS]

Definition at line 80 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.56 unsigned long long int Statistic::TotalQueueDelayPerThread[NO_OF_THREADS]

Definition at line 90 of file stats.h.

Referenced by CalculateAggregateStats(), and InitStats().

7.91.3.57 unsigned long long int Statistic::TotalQueueDelayPerThreadPerBank[NO_OF_THREADS][NO_OF_BANKS]

Definition at line 75 of file stats.h.

Referenced by CalculateAggregateStats(), CollectStatsPerRequest(), and InitStats().

7.91.3.58 float Statistic::WRsch

Definition at line 107 of file stats.h.

Referenced by CalculateAggregateStats(), InitStats(), and PrintAggregateStats().

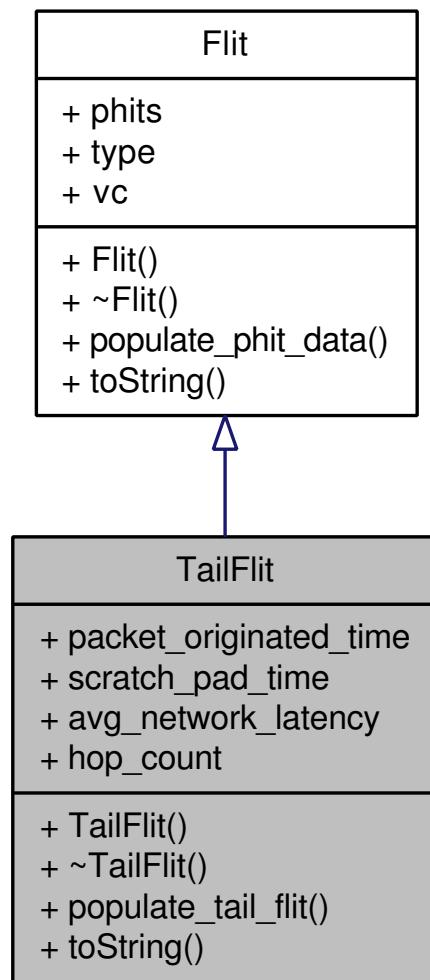
The documentation for this class was generated from the following files:

- **stats.h**
- **stats.cc**

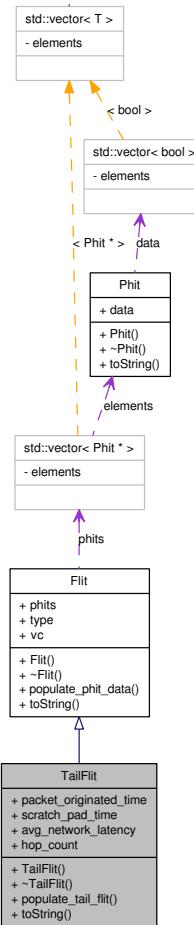
7.92 TailFlit Class Reference

```
#include <flit.h>
```

Inheritance diagram for TailFlit:



Collaboration diagram for TailFlit:



Public Member Functions

- `TailFlit()`
- `~TailFlit()`
- `void populate_tail_flit()`
- `std::string toString()` const

Public Attributes

- `simTime packet_originated_time`
- `simTime scratch_pad_time`
- `double avg_network_latency`
- `unsigned int hop_count`

7.92.1 Detailed Description

Definition at line 148 of file flit.h.

7.92.2 Constructor & Destructor Documentation

7.92.2.1 TailFlit::TailFlit ()

Definition at line 205 of file flit.cc.

References Simulator::Now(), packet_originated_time, TAIL, and Flit::type.

Here is the call graph for this function:



7.92.2.2 TailFlit::~TailFlit ()

Definition at line 175 of file flit.cc.

References Flit::phits.

7.92.3 Member Function Documentation

7.92.3.1 void TailFlit::populate_tail_flit ()

Definition at line 228 of file flit.cc.

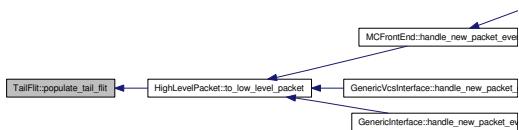
References max_tail_length_bits, packet_originated_time, and Flit::populate_phit_data().

Referenced by HighLevelPacket::to_low_level_packet().

Here is the call graph for this function:



Here is the caller graph for this function:



7.92.3.2 string TailFlit::toString () const

Reimplemented from **Flit** (p. 110).

Definition at line 211 of file flit.cc.

References avg_network_latency, hop_count, packet_originated_time, Flit::phits, and Flit::type.

7.92.4 Member Data Documentation

7.92.4.1 double TailFlit::avg_network_latency

Definition at line 159 of file flit.h.

Referenced by LowLevelPacket::add(), GenericRouterVct::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), and toString().

7.92.4.2 unsigned int TailFlit::hop_count

Definition at line 160 of file flit.h.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), and toString().

7.92.4.3 simTime TailFlit::packet_originated_time

Definition at line 154 of file flit.h.

Referenced by LowLevelPacket::add(), populate_tail_flit(), TailFlit(), HighLevelPacket::to_low_level_packet(), and toString().

7.92.4.4 simTime TailFlit::scratch_pad_time

Definition at line 155 of file flit.h.

The documentation for this class was generated from the following files:

- **flit.h**
- **flit.cc**

7.93 VirtualChannelArbiter Class Reference

```
#include <virtualChannelArbiter.h>
```

Public Member Functions

- **VirtualChannelArbiter ()**
- **virtual void request (Flit *f, uint vc)=0**
- **virtual Flit * pull_winner ()=0**
- **virtual bool ready_(uint ch)=0**
- **virtual uint size ()=0**
- **virtual void resize (uint ports)=0**
- **virtual uint pick_winner ()=0**
- **virtual void clear_()=0**
- **virtual void clear_winner ()=0**
- **virtual vector< uint > get_requests ()=0**

7.93.1 Detailed Description

Definition at line 32 of file virtualChannelArbiter.h.

7.93.2 Constructor & Destructor Documentation

7.93.2.1 VirtualChannelArbiter::VirtualChannelArbiter () [inline]

Definition at line 35 of file virtualChannelArbiter.h.

7.93.3 Member Function Documentation

7.93.3.1 virtual void VirtualChannelArbiter::clear () [pure virtual]

7.93.3.2 virtual void VirtualChannelArbiter::clear_winner () [pure virtual]

7.93.3.3 virtual vector< uint > VirtualChannelArbiter::get_requests () [pure virtual]

7.93.3.4 virtual uint VirtualChannelArbiter::pick_winner () [pure virtual]

7.93.3.5 virtual Flit* VirtualChannelArbiter::pull_winner () [pure virtual]

7.93.3.6 virtual bool VirtualChannelArbiter::ready (uint ch) [pure virtual]

7.93.3.7 virtual void VirtualChannelArbiter::request (Flit * f, uint vc) [pure virtual]

7.93.3.8 virtual void VirtualChannelArbiter::resize (uint ports) [pure virtual]

7.93.3.9 virtual uint VirtualChannelArbiter::size () [pure virtual]

The documentation for this class was generated from the following file:

- **virtualChannelArbiter.h**

7.94 VirtualChannelDescription Class Reference

```
#include <genericData.h>
```

Public Member Functions

- [VirtualChannelDescription \(\)](#)

Public Attributes

- [uint vc](#)
- [uint port](#)

7.94.1 Detailed Description

Definition at line 80 of file genericData.h.

7.94.2 Constructor & Destructor Documentation

7.94.2.1 VirtualChannelDescription::VirtualChannelDescription ()

Definition at line 33 of file genericData.cc.

7.94.3 Member Data Documentation

7.94.3.1 uint VirtualChannelDescription::port

Definition at line 85 of file genericData.h.

Referenced by GenericRouterNoVcs::send_credit_back().

7.94.3.2 uint VirtualChannelDescription::vc

Definition at line 84 of file genericData.h.

Referenced by GenericSink::handle_new_packet_event(), GenericRPG::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), GenericSink::handle_ready_event(), GenericRPG::handle_ready_event(), GenericRouterNoVcs::send_credit_back(), NI::setup(), MCFrontEnd::setup(), GenericSink::setup(), and GenericRPG::setup().

The documentation for this class was generated from the following files:

- [genericData.h](#)
- [genericData.cc](#)

Chapter 8

File Documentation

8.1 addr_map.cc File Reference

```
#include <math.h>
#include "addr_map.h"
#include "request_handler.h"
Include dependency graph for addr_map.cc:
```



Variables

- int **BANK_BITS**

8.1.1 Variable Documentation

8.1.1.1 int BANK_BITS

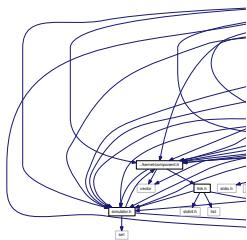
Definition at line 36 of file simMc2Mesh.cc.

Referenced by main(), and AddrMap::map_addr().

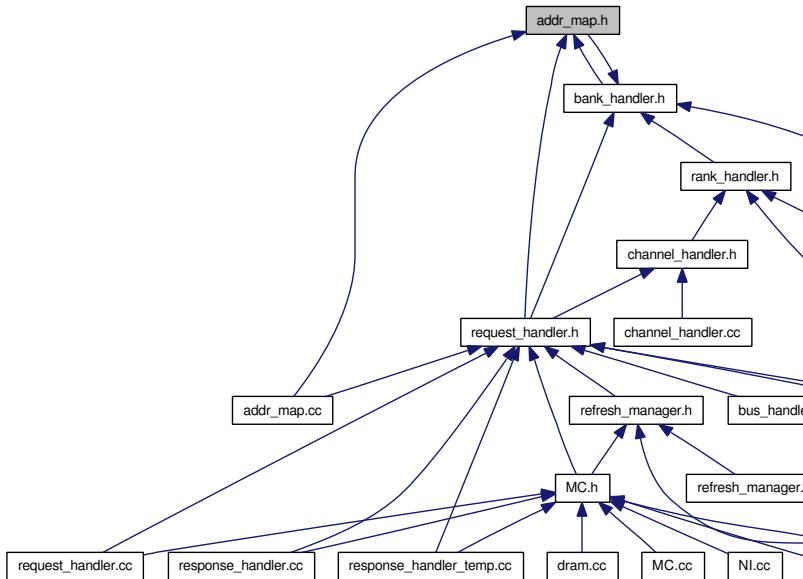
8.2 addr_map.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/bank_handler.h"
#include "../MemCtrl/constants.h"
#include "../MemCtrl/bank_handler.h"

Include dependency graph for addr_map.h:
```



This graph shows which files directly or indirectly include this file:



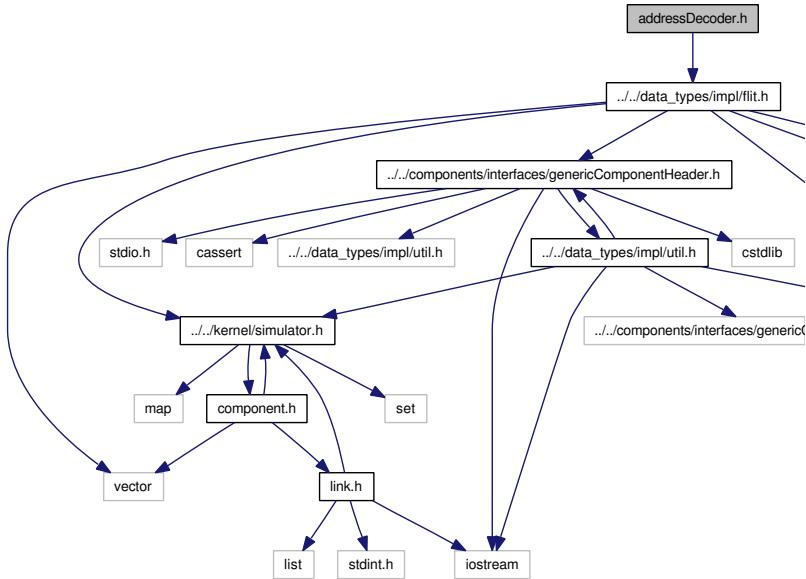
Classes

- class **AddrMap**

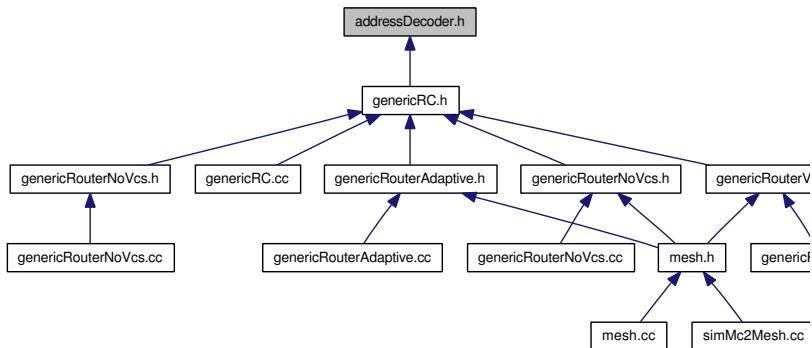
8.3 addressDecoder.h File Reference

```
#include "../../data_types/impl/flit.h"
```

Include dependency graph for addressDecoder.h:



This graph shows which files directly or indirectly include this file:



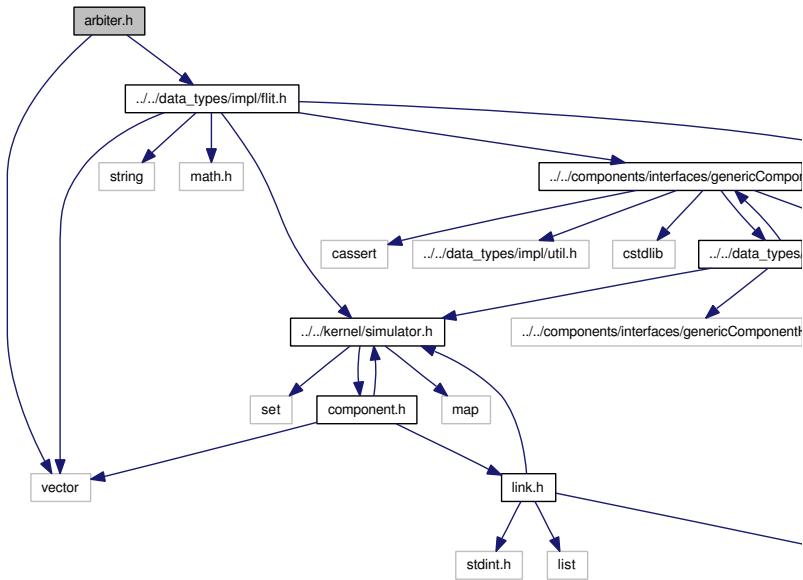
Classes

- class **AddressDecoder**

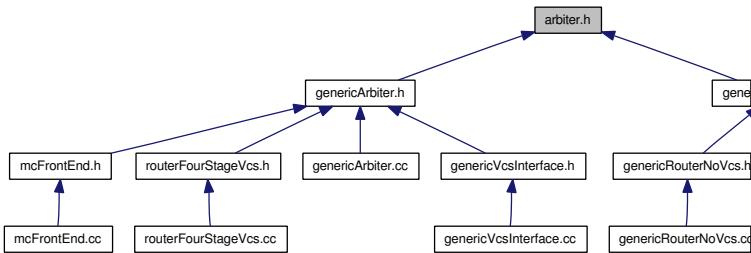
8.4 arbiter.h File Reference

```
#include "../../data_types/impl/flit.h"
#include <vector>
```

Include dependency graph for arbiter.h:



This graph shows which files directly or indirectly include this file:



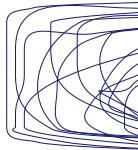
Classes

- class **Arbiter**

8.5 bank_handler.cc File Reference

```
#include <math.h>
#include "bank_handler.h"
#include "../MemCtrl/rank_handler.h"
#include "../MemCtrl/request_handler.h"
#include "../MemCtrl/cmd_issuer.h"
#include "../MemCtrl/response_handler.h"
#include "../tests/MersenneTwister.h"
#include <time.h>
```

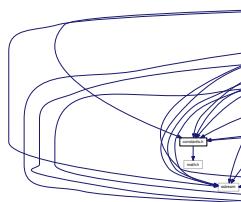
Include dependency graph for bank_handler.cc:



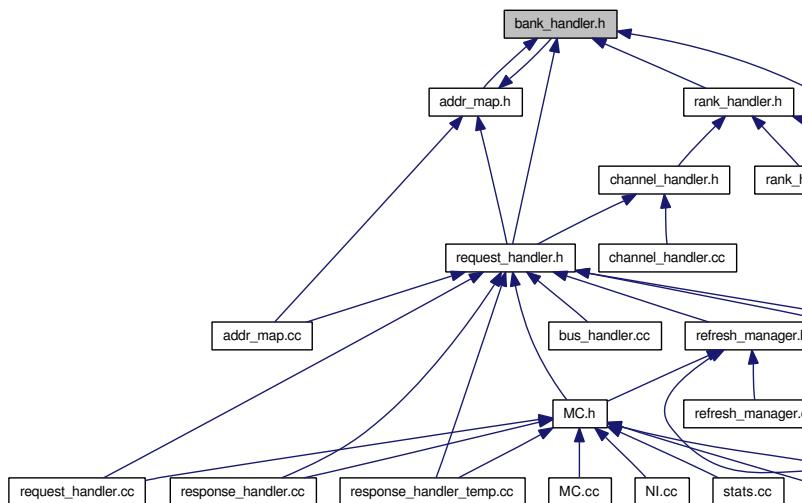
8.6 bank_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/constants.h"
#include "../MemCtrl/addr_map.h"
#include "../MemCtrl/bus_handler.h"
#include "../MemCtrl/addr_map.h"

Include dependency graph for bank_handler.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **BankHandler**

Enumerations

```
- enum CallerType {  
    READS_FIRST, HIT_FIRST, OLDEST_FIRST, UNMARKED_FIRST,  
    HIGHEST_RANKED_FIRST }
```

8.6.1 Enumeration Type Documentation

8.6.1.1 enum CallerType

Enumerator:

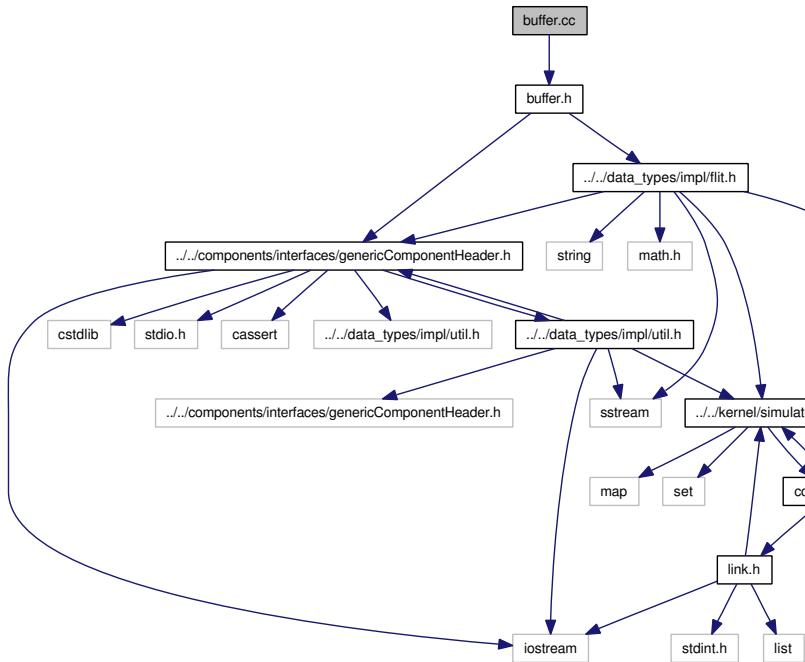
READS_FIRST
HIT_FIRST
OLDEST_FIRST
UNMARKED_FIRST
HIGHEST_RANKED_FIRST

Definition at line 46 of file bank_handler.h.

8.7 buffer.cc File Reference

```
#include "buffer.h"
```

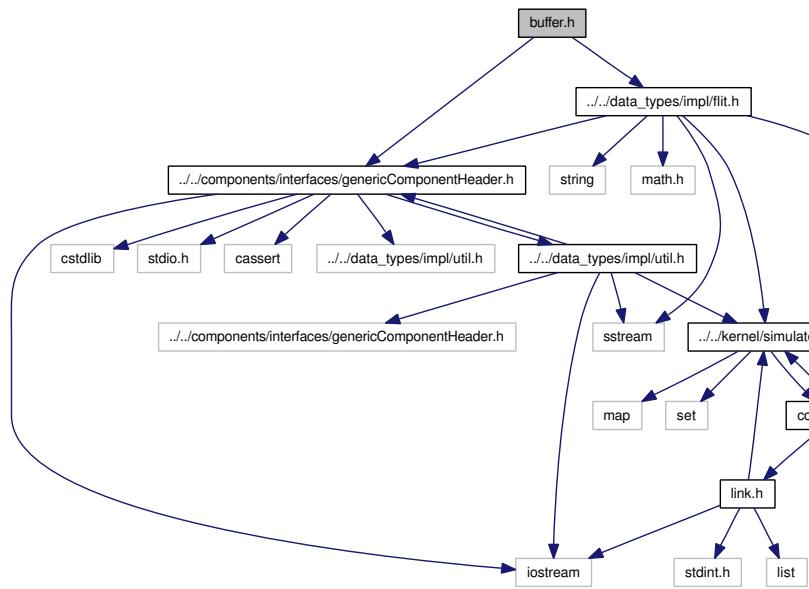
Include dependency graph for buffer.cc:



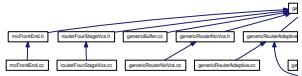
8.8 buffer.h File Reference

```
#include "../../data_types/impl/flit.h"
#include "genericComponentHeader.h"
```

Include dependency graph for buffer.h:



This graph shows which files directly or indirectly include this file:



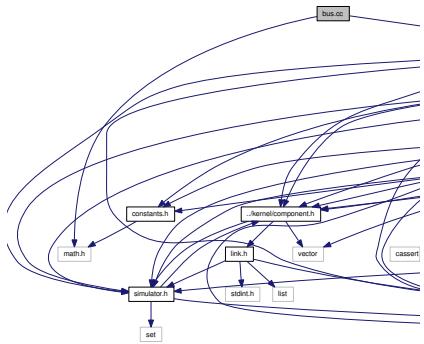
Classes

- class **Buffer**

8.9 bus.cc File Reference

```
#include <math.h>
#include "bus.h"

Include dependency graph for bus.cc:
```

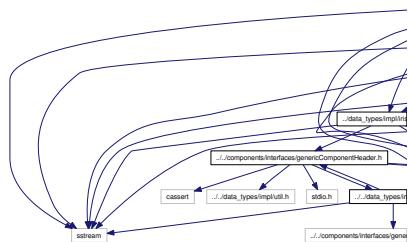


8.10 bus.h File Reference

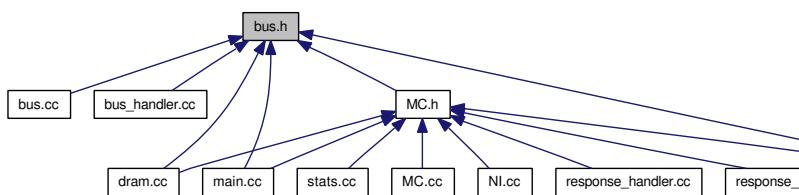
```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/data_bus_handler.h"
#include "../MemCtrl/cmd_bus_handler.h"
#include "../MemCtrl/dram.h"

Include dependency graph for bus.h:
```

Include dependency graph for bus.h:



This graph shows which files directly or indirectly include this file:



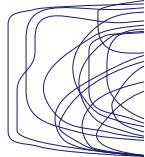
Classes

- class **Bus**

8.11 bus_handler.cc File Reference

```
#include <math.h>
#include "bus_handler.h"
#include "../MemCtrl/request_handler.h"
#include "../MemCtrl/bus.h"
```

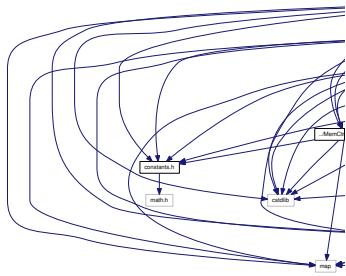
Include dependency graph for bus_handler.cc:



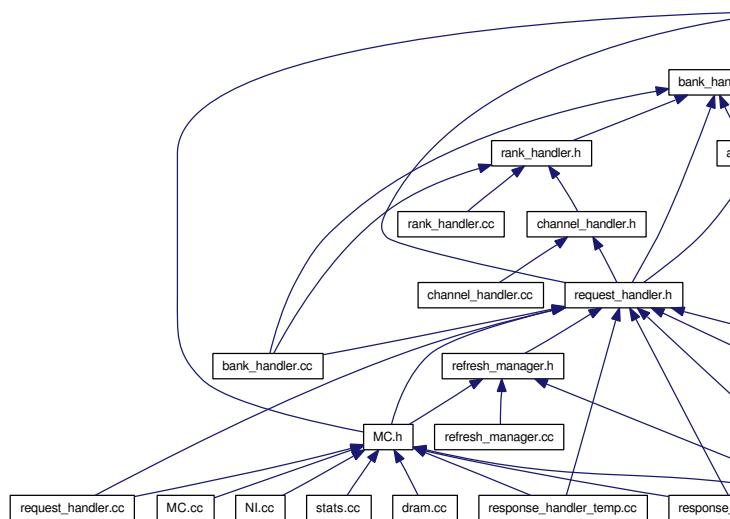
8.12 bus_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/cmd_issuer.h"
#include "../MemCtrl/constants.h"

Include dependency graph for bus_handler.h:
```



This graph shows which files directly or indirectly include this file:

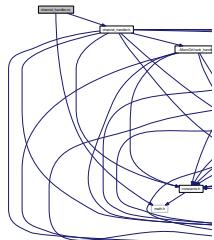


Classes

- class **BusHandler**

8.13 channel_handler.cc File Reference

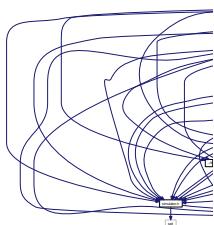
```
#include <math.h>
#include "channel_handler.h"
Include dependency graph for channel_handler.cc:
```



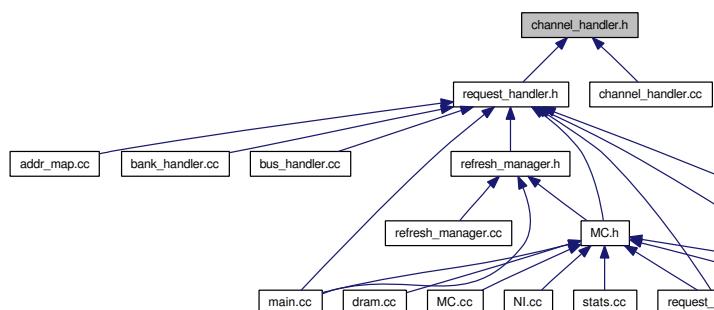
8.14 channel_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/rank_handler.h"
#include "../MemCtrl/constants.h"
```

Include dependency graph for channel_handler.h:



This graph shows which files directly or indirectly include this file:

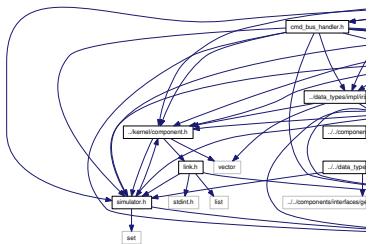


Classes

- class **ChannelHandler**

8.15 cmd_bus_handler.cc File Reference

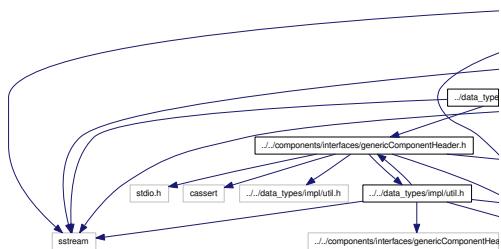
```
#include <math.h>
#include "cmd_bus_handler.h"
#include "../MemCtrl/cmd_issuer.h"
#include "../components/impl/genericEvents.h"
Include dependency graph for cmd_bus_handler.cc:
```



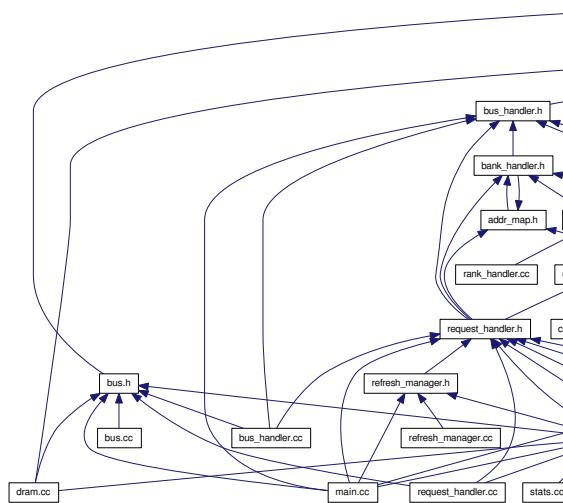
8.16 cmd_bus_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/dram.h"
```

Include dependency graph for cmd_bus_handler.h:



This graph shows which files directly or indirectly include this file:

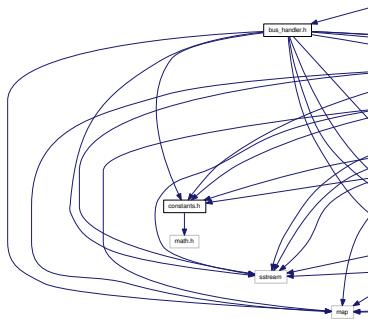


Classes

- class **CmdBusHandler**

8.17 cmd_issuer.cc File Reference

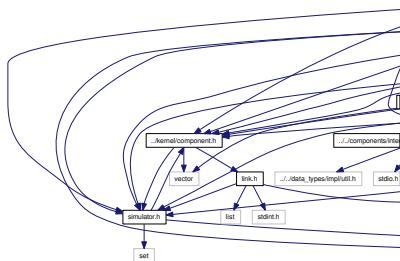
```
#include "cmd_issuer.h"
#include "bus_handler.h"
#include "../components/impl/genericEvents.h"
Include dependency graph for cmd_issuer.cc:
```



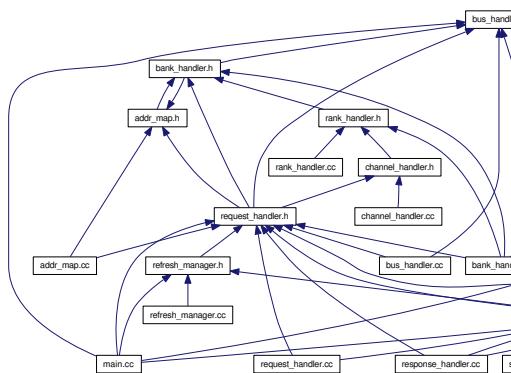
8.18 cmd issuer.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/data_bus_handler.h"
#include "../MemCtrl/cmd_bus_handler.h"
#include "../MemCtrl/constants.h"
```

Include dependency graph for cmd_issuer.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct **DRAMCmdState**
 - struct **BankState**
 - struct **RankState**
 - struct **ChannelState**
 - class **CmdIssuer**

Typedefs

- `typedef DRAMCmd DRAMCmd`
- `typedef BurstLength BurstLength`
- `typedef DRAMCmdState DRAMCmdState`
- `typedef vector< DRAMCmdState > CmdQueue`
- `typedef BankState BankState`
- `typedef RankState RankState`
- `typedef ChannelState ChannelState`

Enumerations

- `enum DRAMCmd {`
`READ, WRITE, PRECHARGE, ACTIVATE,`
`ALL_BANK_REFRESH }`
- `enum BurstLength {`
`READL, WRITEL, PREFETCHL, WRITEBACKL,`
`NORMAL }`
- `enum RankBankComb { SAME, DIFF }`

8.18.1 Typedef Documentation

8.18.1.1 `typedef BankState BankState`

Definition at line 88 of file cmd_issuer.h.

8.18.1.2 `typedef BurstLength BurstLength`

Definition at line 42 of file cmd_issuer.h.

8.18.1.3 `typedef ChannelState ChannelState`

Definition at line 104 of file cmd_issuer.h.

8.18.1.4 `typedef vector<DRAMCmdState> CmdQueue`

Definition at line 72 of file cmd_issuer.h.

8.18.1.5 `typedef DRAMCmd DRAMCmd`

Definition at line 39 of file cmd_issuer.h.

8.18.1.6 `typedef DRAMCmdState DRAMCmdState`

Definition at line 71 of file cmd_issuer.h.

8.18.1.7 `typedef RankState RankState`

Definition at line 94 of file cmd_issuer.h.

8.18.2 Enumeration Type Documentation

8.18.2.1 enum BurstLength

Enumerator:

READL
WRITEL
PREFETCHL
WRITEBACKL
NORMAL

Definition at line 41 of file cmd_issuer.h.

8.18.2.2 enum DRAMCmd

Enumerator:

READ
WRITE
PRECHARGE
ACTIVATE
ALL_BANK_REFRESH

Definition at line 38 of file cmd_issuer.h.

8.18.2.3 enum RankBankComb

Enumerator:

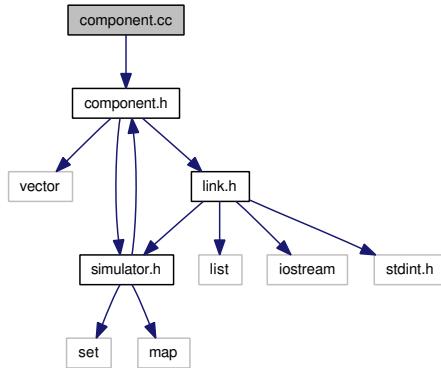
SAME
DIFF

Definition at line 74 of file cmd_issuer.h.

8.19 component.cc File Reference

```
#include "component.h"
```

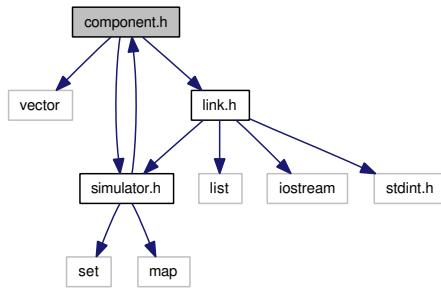
Include dependency graph for component.cc:



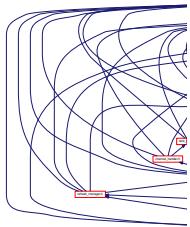
8.20 component.h File Reference

```
#include <vector>
#include "simulator.h"
#include "link.h"
```

Include dependency graph for component.h:



This graph shows which files directly or indirectly include this file:



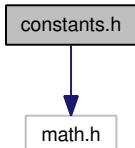
Classes

- class **Component**

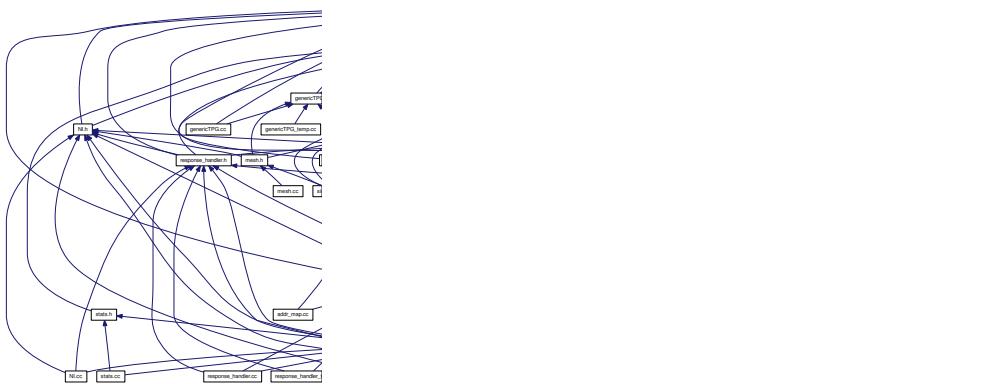
8.21 constants.h File Reference

```
#include <math.h>
```

Include dependency graph for constants.h:



This graph shows which files directly or indirectly include this file:



Defines

```
- #define OPEN_PAGE_POLICY 1
- #define FR_FCFS 1
- #define PAGE_INTERLEAVING 1
- #define NO_OF_THREADS 64
- #define NO_OF_CHANNELS 1
- #define NO_OF_RANKS 1
- #define NO_OF_BANKS 8
- #define NO_OF_BUFFERS NO_OF_BANKS
- #define NO_OF_ROWS 8192
- #define NO_OF_COLUMNS 128
- #define COLUMN_SIZE 64
- #define BLOCKS_PER_ROW 128
- #define CACHE_BLOCK_SIZE 64
- #define ROW_SIZE NO_OF_COLUMNS*COLUMN_SIZE
- #define DRAM_SIZE NO_OF_CHANNELS*NO_OF_RANKS*NO_OF_BANKS*NO_OF_ROWS*ROW_SIZE
- #define TAG_BITS 8
- #define USE_MSHR 1
- #define MSHR_SIZE 8
- #define MAX_BUFFER_SIZE 8
- #define MAX_CMD_BUFFER_SIZE 16
- #define RESPONSE_BUFFER_SIZE 56*8
- #define BATCH_FORM_TIME 2000;
- #define MAX_BATCH_SIZE 5
- #define MAX_READ_OVER_WRITE 8
- #define NETWORK_ADDRESS_BITS 32
```

```

- #define NETWORK_THREADID_BITS 6
- #define NETWORK_COMMAND_BITS 3
- #define READ_SIZE CACHE_BLOCK_SIZE
- #define WRITE_SIZE CACHE_BLOCK_SIZE
- #define PREFETCH_SIZE CACHE_BLOCK_SIZE
- #define WRITEBACK_SIZE CACHE_BLOCK_SIZE
- #define DDR_BUS_WIDTH 8
- #define BUS_SPEED 1600
- #define CORE_SPEED 2000
- #define MEM_SPEED 800
- #define MEM_CYCLE (CORE_SPEED*1.0 / MEM_SPEED)
- #define BUS_CYCLE (CORE_SPEED*1.0 / BUS_SPEED)
- #define CYCLE_2_NS (CORE_SPEED*1.0 / 1000)
- #define tREFI 7.8
- #define tRFC 160
- #define tRC 47.5
- #define tRAS 35
- #define t_CMD ceil (1.0 * BUS_CYCLE)
- #define t_RCD ceil (12.5 * CYCLE_2_NS)
- #define t_RRD ceil (6.0 * CYCLE_2_NS)
- #define t_RAS ceil (35 * CYCLE_2_NS)
- #define t_CAS ceil (10.0 * MEM_CYCLE)
- #define t_RTRS ceil (1.0 * MEM_CYCLE)
- #define t_OST ceil (1.0 * MEM_CYCLE)
- #define t_WR ceil (15 * CYCLE_2_NS)
- #define t_WTR ceil (7.5 * CYCLE_2_NS)
- #define t_RP ceil (12.5 * CYCLE_2_NS)
- #define t_CCD ceil (DDR_BUS_WIDTH/2 * MEM_CYCLE)
- #define t_AL 0
- #define t_CWD t_CAS-t_CMD
- #define t_RC ceil (47.5 * CYCLE_2_NS)
- #define t_RTP ceil (7.5 * CYCLE_2_NS)
- #define t_RFC ceil (160 * CYCLE_2_NS)
- #define REFRESH_PERIOD CORE_SPEED*64000
- #define REFRESH_INC floor(REFRESH_PERIOD/(8192)) - BUS_CYCLE

```

Typedefs

- typedef unsigned long long int **Time**
- typedef unsigned long long int **Addr_t**
- typedef unsigned int **UInt**

8.21.1 Define Documentation

8.21.1.1 #define BATCH_FORM_TIME 2000;

Definition at line 78 of file constants.h.

Referenced by RequestHandler::FormBatch(), and RequestHandler::process_event().

8.21.1.2 #define BLOCKS_PER_ROW 128

Definition at line 62 of file constants.h.

Referenced by MSHR_H::local_map_addr(), and AddrMap::map_addr().

8.21.1.3 #define BUS_CYCLE (CORE_SPEED*1.0 / BUS_SPEED)

Definition at line 144 of file constants.h.

Referenced by CmdIssuer::CalculateBurstL(), and DataBusHandler::process_event().

8.21.1.4 #define BUS_SPEED 1600

Definition at line 140 of file constants.h.

8.21.1.5 #define CACHE_BLOCK_SIZE 64

Definition at line 63 of file constants.h.

Referenced by GenericTPG::convertFromBitStream(), NI::convertToBitStream(), GenericTPG::convertToBitStream(), MSHR_H::local_map_addr(), and AddrMap::map_addr().

8.21.1.6 #define COLUMN_SIZE 64

Definition at line 61 of file constants.h.

Referenced by MSHR_H::local_map_addr(), and AddrMap::map_addr().

8.21.1.7 #define CORE_SPEED 2000

Definition at line 141 of file constants.h.

8.21.1.8 #define CYCLE_2_NS (CORE_SPEED*1.0 / 1000)

Definition at line 146 of file constants.h.

8.21.1.9 #define DDR_BUS_WIDTH 8

Definition at line 139 of file constants.h.

Referenced by CmdIssuer::CalculateBurstL(), and DataBusHandler::process_event().

8.21.1.10 #define DRAM_SIZE_NO_OF_CHANNELS*NO_OF_RANKS*NO_OF_BANKS*NO_OF_ROWS*ROW_SIZE

Definition at line 65 of file constants.h.

Referenced by MSHR_H::local_map_addr(), and AddrMap::map_addr().

8.21.1.11 #define FR_FCFS 1

Definition at line 33 of file constants.h.

8.21.1.12 #define MAX_BATCH_SIZE 5

Definition at line 80 of file constants.h.

Referenced by RequestHandler::MarkBatchOnly().

8.21.1.13 #define MAX_BUFFER_SIZE 8

Definition at line 74 of file constants.h.

Referenced by BankHandler::CanUpperStart(), BankHandler::FindHighest(), and BankHandler::IsBufferFull().

8.21.1.14 #define MAX_CMD_BUFFER_SIZE 16

Definition at line 75 of file constants.h.

Referenced by BusHandler::SetIfFull().

8.21.1.15 #define MAX_READ_OV_WRITE 8

Definition at line 81 of file constants.h.

Referenced by BankHandler::ReadsFirst().

8.21.1.16 #define MEM_CYCLE (CORE_SPEED*1.0 / MEM_SPEED)

Definition at line 143 of file constants.h.

8.21.1.17 #define MEM_SPEED 800

Definition at line 142 of file constants.h.

8.21.1.18 #define MSHR_SIZE 8

Definition at line 70 of file constants.h.

Referenced by GenericFlatMc::handle_new_packet_event(), GenericFlatMc::handle_ready_event(), and MSHR_H::process_event().

8.21.1.19 #define NETWORK_ADDRESS_BITS 32

Definition at line 83 of file constants.h.

Referenced by NI::convertFromBitStream(), MCFrontEnd::convertFromBitStream(), GenericTPG::convertFromBitStream(), GenericFlatMc::convertFromBitStream(), NI::convertToBitStream(), and GenericTPG::convertToBitStream().

8.21.1.20 #define NETWORK_COMMAND_BITS 3

Definition at line 85 of file constants.h.

Referenced by NI::convertFromBitStream(), MCFrontEnd::convertFromBitStream(), GenericFlatMc::convertFromBitStream(), and GenericTPG::convertToBitStream().

8.21.1.21 #define NETWORK_THREADID_BITS 6

Definition at line 84 of file constants.h.

Referenced by NI::convertFromBitStream(), MCFrontEnd::convertFromBitStream(), GenericFlatMc::convertFromBitStream(), and GenericTPG::convertToBitStream().

8.21.1.22 #define NO_OF_BANKS 8

Definition at line 57 of file constants.h.

Referenced by CmdIssuer::BankNotBusy(), BusHandler::BusHandler(), Statistic::CalculateAggregateStats(), CmdIssuer::CmdIssuer(), MSHR_H::countBLP(), DRAM::DRAM(), Statistic::InitStats(), MSHR_H::local_map_addr(), AddrMap::map_addr(), Statistic::PrintAggregateStats(), DRAMChannel::process_event(), Request::Request(), and RequestHandler::RequestHandler().

8.21.1.23 #define NO_OF_BUFFERS NO_OF_BANKS

Definition at line 58 of file constants.h.

Referenced by Statistic::CollectStatsPerCycle(), BankHandler::FindHighest(), Statistic::InitStats(), RequestHandler::MarkAll(), RequestHandler::MarkBatchOnly(), Statistic::PrintAggregateStats(), and RequestHandler::RequestHandler().

8.21.1.24 #define NO_OF_CHANNELS 1

Definition at line 55 of file constants.h.

Referenced by BusHandler::BusHandler(), Statistic::CalculateAggregateStats(), Statistic::CollectStatsPerCycle(), DRAM::DRAM(), Statistic::InitStats(), MSHR_H::local_map_addr(), BusHandler::LowLevelCmdGen(), AddrMap::map_addr(), RequestHandler::MarkAll(), RequestHandler::MarkBatchOnly(), Statistic::PrintAggregateStats(), RequestHandler::process_event(), RefreshMgr::process_event(), Request::Request(), RequestHandler::RequestHandler(), BusHandler::SetIfFull(), RequestHandler::SetLinks(), DRAM::SetLinks(), and Bus::SetLinks().

8.21.1.25 #define NO_OF_COLUMNS 128

Definition at line 60 of file constants.h.

Referenced by MSHR_H::local_map_addr(), AddrMap::map_addr(), and Request::Request().

8.21.1.26 #define NO_OF_RANKS 1

Definition at line 56 of file constants.h.

Referenced by BusHandler::BusHandler(), Statistic::CalculateAggregateStats(), CmdIssuer::CmdIssuer(), Statistic::CollectStatsPerCycle(), DRAM::DRAM(), Statistic::InitStats(), MSHR_H::local_map_addr(), AddrMap::map_addr(), RequestHandler::MarkAll(), RequestHandler::MarkBatchOnly(), Statistic::PrintAggregateStats(), RefreshMgr::process_event(), Request::Request(), and RequestHandler::RequestHandler().

8.21.1.27 #define NO_OF_ROWS 8192

Definition at line 59 of file constants.h.

Referenced by MSHR_H::local_map_addr(), AddrMap::map_addr(), RefreshMgr::process_event(), and Request::Request().

8.21.1.28 #define NO_OF_THREADS 64

Definition at line 54 of file constants.h.

Referenced by Statistic::CalculateAggregateStats(), BankHandler::FindHighest(), GetNextRequest(), MSHR_H::GlobalAddrMap(), Statistic::InitStats(), main(), MSHR_H::map_addr(), RequestHandler::MarkBatchOnly(), and Statistic::PrintAggregateStats().

8.21.1.29 #define OPEN_PAGE_POLICY 1

Definition at line 29 of file constants.h.

8.21.1.30 #define PAGE_INTERLEAVING 1

Definition at line 44 of file constants.h.

8.21.1.31 #define PREFETCH_SIZE CACHE_BLOCK_SIZE

Definition at line 89 of file constants.h.

Referenced by CmdIssuer::CalculateBurstL(), and DRAMCmdState::set().

8.21.1.32 #define READ_SIZE CACHE_BLOCK_SIZE

Definition at line 87 of file constants.h.

Referenced by CmdIssuer::CalculateBurstL(), and DRAMCmdState::set().

8.21.1.33 #define REFRESH_INC floor(REFRESH_PERIOD/(8192)) - BUS_CYCLE

Definition at line 279 of file constants.h.

Referenced by RefreshMgr::process_event(), and MC::StartRefresh().

8.21.1.34 #define REFRESH_PERIOD CORE_SPEED*64000

Definition at line 278 of file constants.h.

8.21.1.35 #define RESPONSE_BUFFER_SIZE 56*8

Definition at line 76 of file constants.h.

Referenced by ResponseHandler::CanStart(), ResponseHandler::IsBufferFull(), BankHandler::process_event(), and ResponseHandler::ResponseHandler().

8.21.1.36 #define ROW_SIZE NO_OF_COLUMNS*COLUMN_SIZE

Definition at line 64 of file constants.h.

Referenced by MSHR_H::local_map_addr(), AddrMap::map_addr(), and RefreshMgr::process_event().

8.21.1.37 #define t_AL 0

Definition at line 164 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.38 #define t_CAS ceil (10.0 * MEM_CYCLE)

Definition at line 157 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime(), CmdIssuer::CmdDelay(), and DRAMChannel::process_event().

8.21.1.39 #define t_CCD ceil (DDR_BUS_WIDTH/2 * MEM_CYCLE)

Definition at line 163 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.40 #define t_CMD ceil (1.0 * BUS_CYCLE)

Definition at line 153 of file constants.h.

Referenced by CmdIssuer::BusNotBusy(), CmdIssuer::CalculateBusyTime(), CmdIssuer::CalculateDataDelay(), DataBusHandler::process_event(), and CmdBusHandler::process_event().

8.21.1.41 #define t_CWD t_CAS-t_CMD

Definition at line 165 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime(), and CmdIssuer::CalculateDataDelay().

8.21.1.42 #define t_OST ceil (1.0 * MEM_CYCLE)

Definition at line 159 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.43 #define t_RAS ceil (35 * CYCLE_2_NS)

Definition at line 156 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.44 #define t_RC ceil (47.5 * CYCLE_2_NS)

Definition at line 166 of file constants.h.

8.21.1.45 #define t_RCD ceil (12.5 * CYCLE_2_NS)

Definition at line 154 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime(), CmdIssuer::CmdDelay(), and DRAMChannel::process_event().

8.21.1.46 #define t_RFC ceil (160 * CYCLE_2_NS)

Definition at line 168 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime(), CmdIssuer::CmdDelay(), and DRAMChannel::process_event().

8.21.1.47 #define t_RP ceil (12.5 * CYCLE_2_NS)

Definition at line 162 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime(), CmdIssuer::CmdDelay(), and DRAMChannel::process_event().

8.21.1.48 #define t_RRD ceil (6.0 * CYCLE_2_NS)

Definition at line 155 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.49 #define t RTP ceil (7.5 * CYCLE_2_NS)

Definition at line 167 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.50 #define t_RTRS ceil (1.0 * MEM_CYCLE)

Definition at line 158 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.51 #define t_WR ceil (15 * CYCLE_2_NS)

Definition at line 160 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime(), and DRAMChannel::process_event().

8.21.1.52 #define t_WTR ceil (7.5 * CYCLE_2_NS)

Definition at line 161 of file constants.h.

Referenced by CmdIssuer::CalculateBusyTime().

8.21.1.53 #define TAG_BITS 8

Definition at line 66 of file constants.h.

Referenced by MSHR_H::local_map_addr(), and AddrMap::map_addr().

8.21.1.54 #define tRAS 35

Definition at line 151 of file constants.h.

Referenced by PowerStats::CalcPower().

8.21.1.55 #define tRC 47.5

Definition at line 150 of file constants.h.

Referenced by PowerStats::CalcPower().

8.21.1.56 #define tREFI 7.8

Definition at line 148 of file constants.h.

Referenced by PowerStats::CalcPower().

8.21.1.57 #define tRFC 160

Definition at line 149 of file constants.h.

Referenced by PowerStats::CalcPower().

8.21.1.58 #define USE_MSHR 1

Definition at line 69 of file constants.h.

8.21.1.59 #define WRITE_SIZE CACHE_BLOCK_SIZE

Definition at line 88 of file constants.h.

Referenced by CmdIssuer::CalculateBurstL(), and DRAMCmdState::set().

8.21.1.60 #define WRITEBACK_SIZE CACHE_BLOCK_SIZE

Definition at line 90 of file constants.h.

Referenced by CmdIssuer::CalculateBurstL(), MCFrontEnd::convertFromBitStream(), and DRAMCmdState::set().

8.21.2 Typedef Documentation**8.21.2.1 typedef unsigned long long int Addr_t**

Definition at line 23 of file constants.h.

8.21.2.2 typedef unsigned long long int Time

Definition at line 22 of file constants.h.

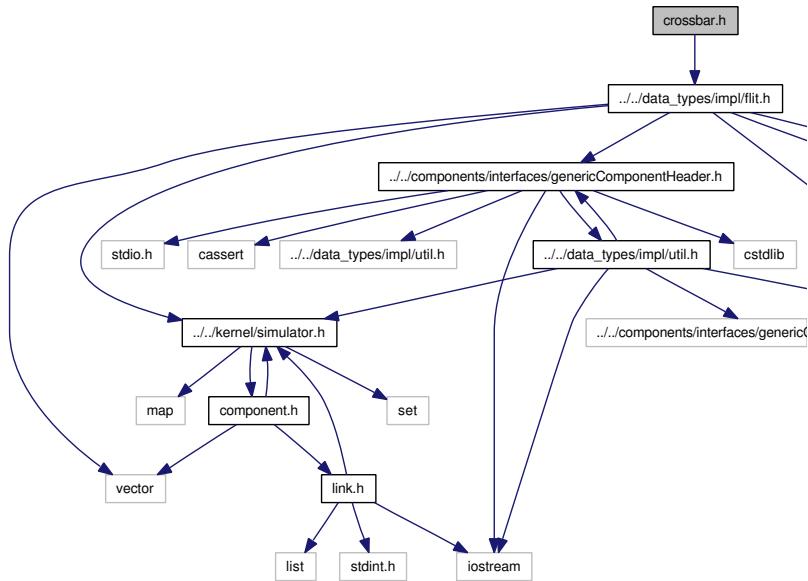
8.21.2.3 typedef unsigned int UInt

Definition at line 24 of file constants.h.

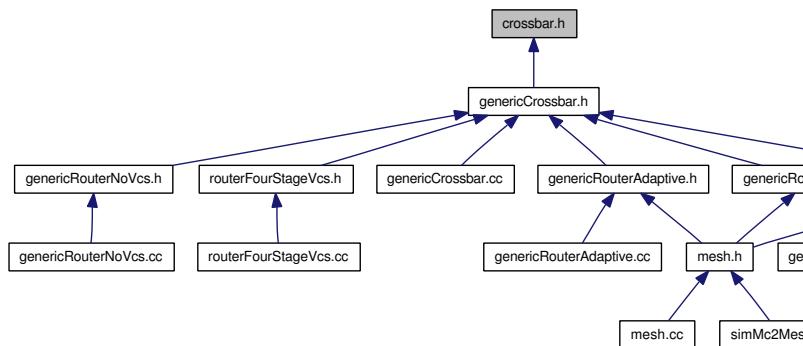
8.22 crossbar.h File Reference

```
#include "....data_types/impl/flit.h"
```

Include dependency graph for crossbar.h:



This graph shows which files directly or indirectly include this file:

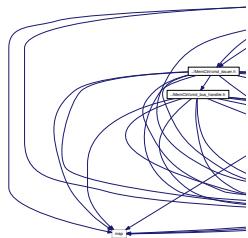


Classes

- ### – class **Crossbar**

8.23 data_bus_handler.cc File Reference

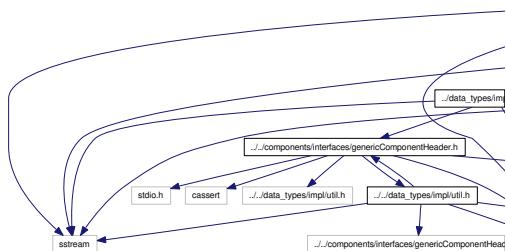
```
#include <math.h>
#include "data_bus_handler.h"
#include "../MemCtrl/cmd_issuer.h"
#include "../MemCtrl/response_handler.h"
Include dependency graph for data_bus_handler.cc:
```



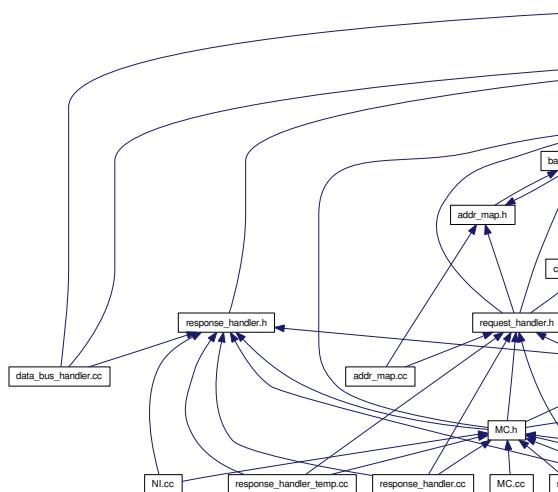
8.24 data_bus_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/dram.h"
#include "../MemCtrl/constants.h"

Include dependency graph for data_bus_handler.h:
```



This graph shows which files directly or indirectly include this file:



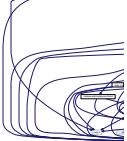
Classes

- class **DataBusHandler**

8.25 dram.cc File Reference

```
#include <math.h>
#include "dram.h"
#include "../MemCtrl/bus.h"
#include "../MemCtrl/cmd_issuer.h"
#include "../MemCtrl/data_bus_handler.h"
#include "MC.h"
```

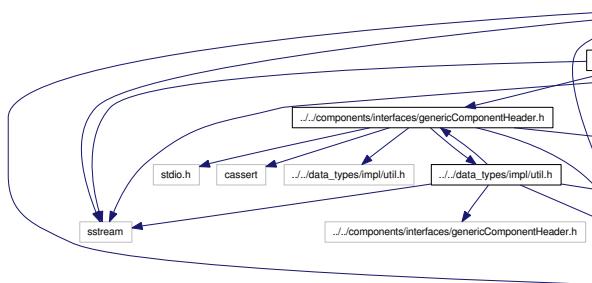
Include dependency graph for dram.cc:



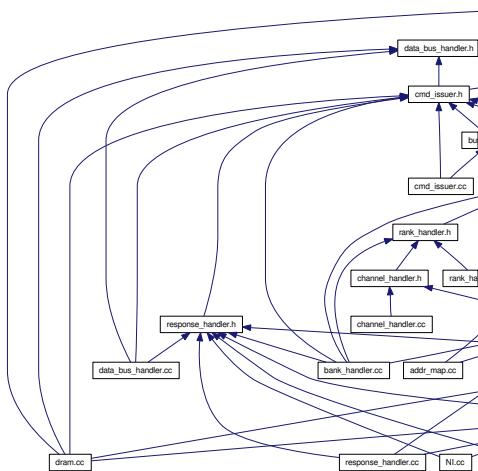
8.26 dram.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/constants.h"
```

Include dependency graph for dram.h:



This graph shows which files directly or indirectly include this file:



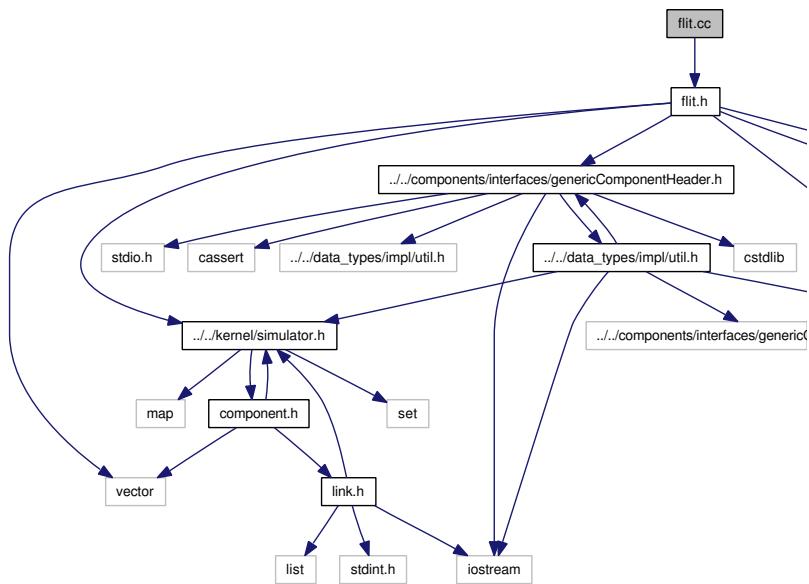
Classes

- class **DRAMChannel**
- class **DRAM**

8.27 flit.cc File Reference

```
#include "flit.h"
```

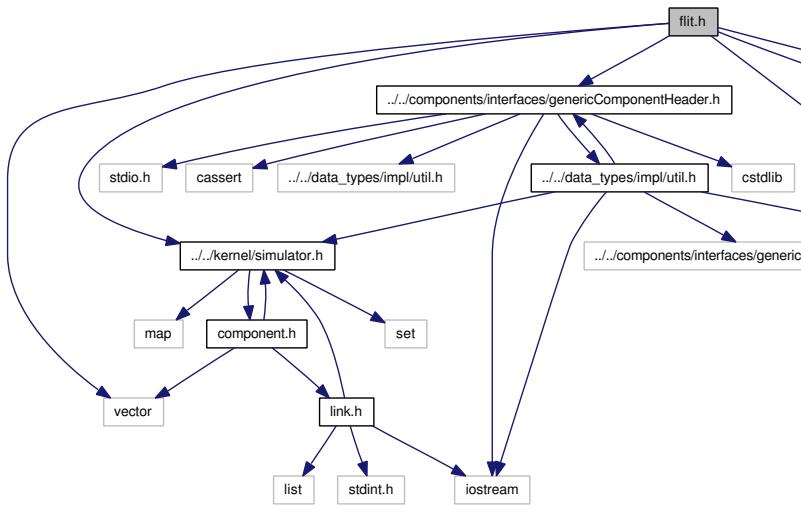
Include dependency graph for flit.cc:



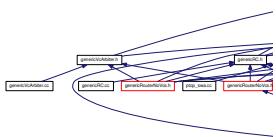
8.28 flit.h File Reference

```
#include <vector>
#include <string>
#include <sstream>
#include <math.h>
#include "../../components/interfaces/genericComponentHeader.h"
#include "../../kernel/simulator.h"

Include dependency graph for flit.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **Phit**
- class **Flit**
- class **HeadFlit**
- class **BodyFlit**
- class **TailFlit**

Enumerations

- enum **flit_type** { **HEAD**, **BODY**, **TAIL** }

Variables

- uint **max_phy_link_bits**

8.28.1 Enumeration Type Documentation

8.28.1.1 enum flit_type

Enumerator:

HEAD

BODY

TAIL

Definition at line 31 of file flit.h.

8.28.2 Variable Documentation

8.28.2.1 uint max_phy_link_bits

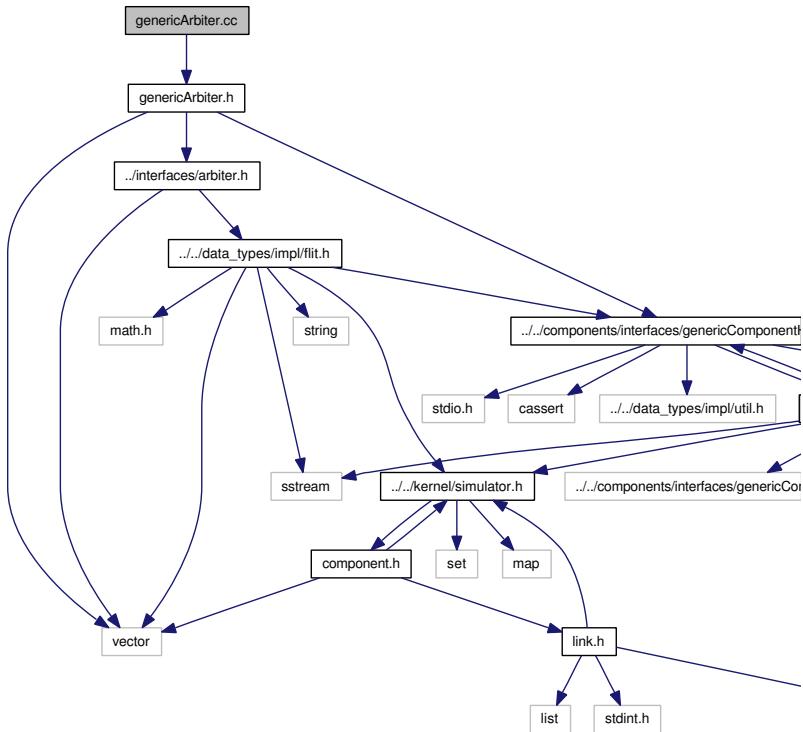
Definition at line 39 of file simMc2Mesh.cc.

Referenced by GenericTPG::convertFromBitStream(), NI::convertToBitStream(), GenericTPG::convertToBitStream(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), main(), Flit::populate_phit_data(), and HighLevelPacket::to_low_level_packet().

8.29 genericArbiter.cc File Reference

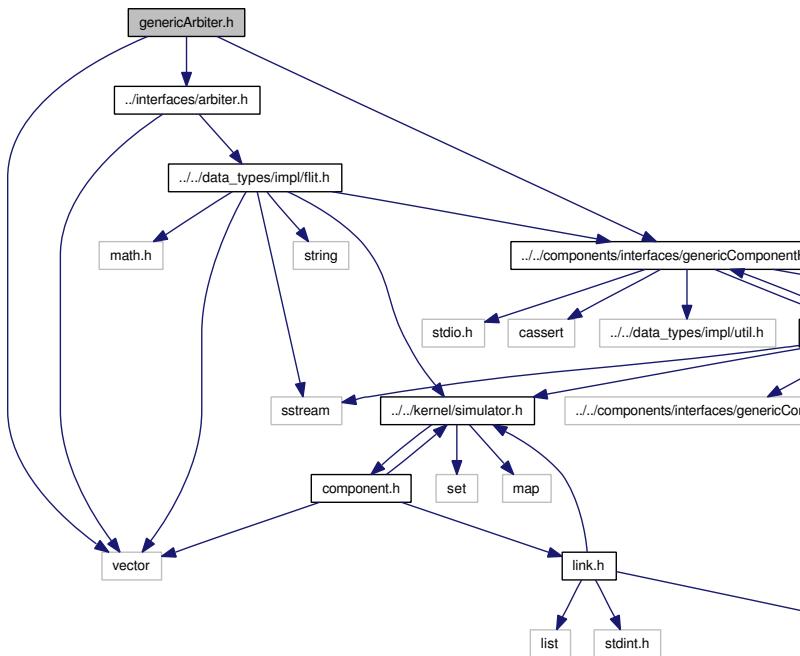
```
#include "genericArbiter.h"
```

Include dependency graph for genericArbiter.cc:

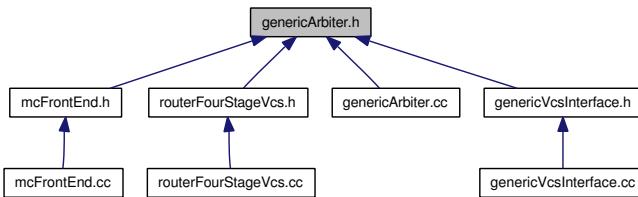


8.30 genericArbiter.h File Reference

```
#include <vector>
#include "../interfaces/arbiter.h"
#include "../interfaces/genericComponentHeader.h"
Include dependency graph for genericArbiter.h:
```



This graph shows which files directly or indirectly include this file:



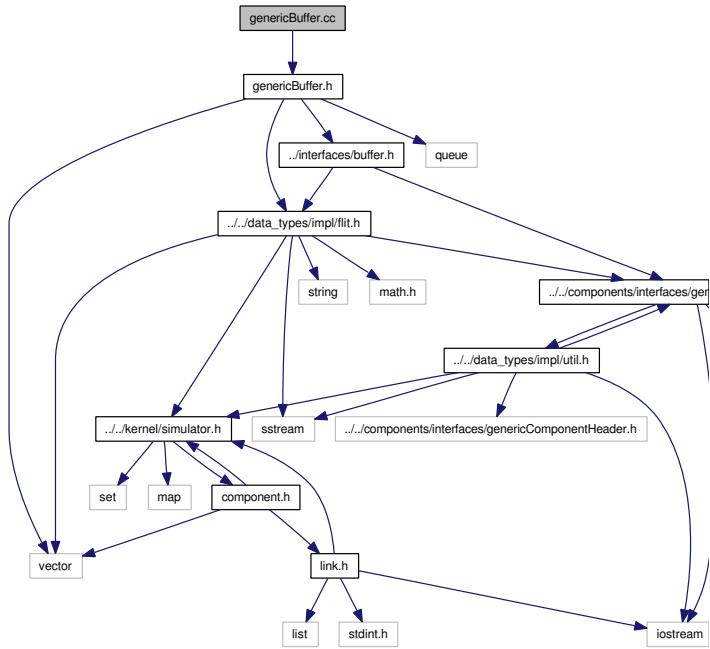
Classes

- class **GenericArbiter**

8.31 genericBuffer.cc File Reference

```
#include "genericBuffer.h"
```

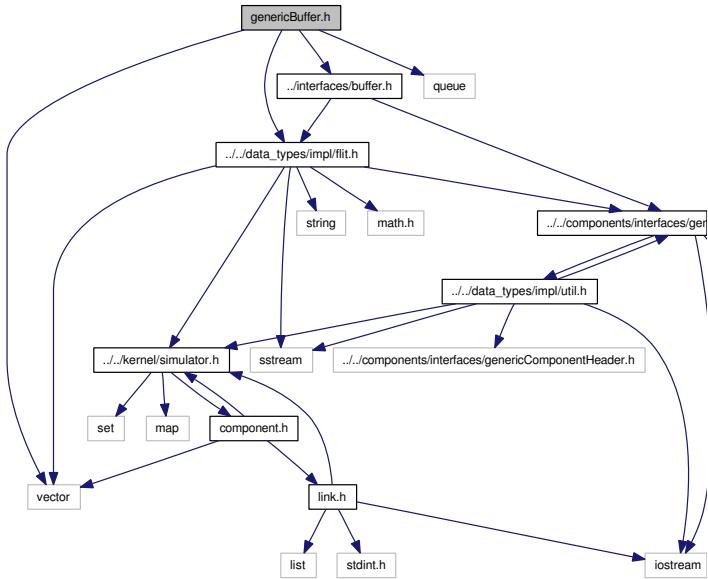
Include dependency graph for genericBuffer.cc:



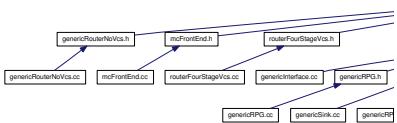
8.32 genericBuffer.h File Reference

```
#include "../interfaces/buffer.h"
#include "../../data_types/impl/flit.h"
#include <queue>
#include <vector>
```

Include dependency graph for genericBuffer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericBuffer**

Defines

- #define **BUFFER_SIZE** 20

8.32.1 Define Documentation

8.32.1.1 #define **BUFFER_SIZE** 20

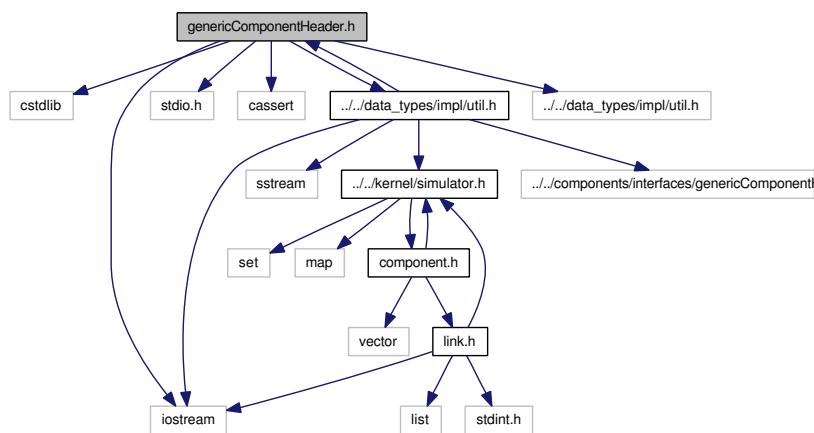
Definition at line 26 of file genericBuffer.h.

Referenced by GenericBuffer::is_channel_full().

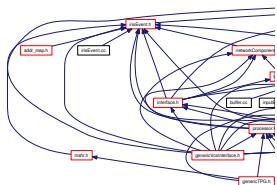
8.33 genericComponentHeader.h File Reference

```
#include <cstdlib>
#include <iostream>
#include <stdio.h>
#include <cassert>
#include "../../data_types/impl/util.h"
#include "../../data_types/impl/util.h"
```

Include dependency graph for genericComponentHeader.h:



This graph shows which files directly or indirectly include this file:



Defines

- #define **DEFAULT_ADDRESS** 0
- #define **DEFAULT_CONVERT_PACKET_CYCLES** 1
- #define **NO_DATA** true
- #define **FLIT_ID** 9800
- #define **CREDIT_ID** 9801
- #define LOC std::cout << "nTime:" << dec << Simulator::Now() << " " << name << " " << address << " " << node_ip << " ";
- #define **DBG(fmt,...)** LOC printf(fmt, __VA_ARGS__);
- #define **_DBG_NOARG(fmt)** LOC printf(fmt);

Typedefs

- typedef unsigned long int **uniqueId**
- typedef unsigned long long int **simTime**
- typedef unsigned long long int **ullint**
- typedef unsigned int **uint**

Enumerations

- enum **message_class** { **INVALID_PKT**, **REQUEST_PKT**, **RESPONSE_PKT**, **ONE_FLIT_REQ** }

Variables

- const unsigned int **max_network_node_bits** = 8
- const unsigned int **max_transaction_id_bits** = 8
- const unsigned int **max_tail_length_bits** = 8
- const unsigned int **max_control_bits** = 8

8.33.1 Define Documentation

8.33.1.1 #define _DBG(fmt, ...) LOC printf(fmt, __VA_ARGS__);

Definition at line 37 of file genericComponentHeader.h.

Referenced by `GenericRouterVct::do_switch_allocation()`, `GenericRouterNoVcs::do_switch_allocation()`, `GenericRouterAdaptive::do_switch_allocation()`, `GenericRouterVct::do_switch_traversal()`, `GenericRouterNoVcs::do_switch_traversal()`, `GenericRouterAdaptive::do_switch_traversal()`, `GenericVcArbiter::empty()`, `GenericInterface::handle_link_arrival()`, `RouterFourStageVcs::handle_link_arrival_event()`, `GenericRouterVct::handle_link_arrival_event()`, `GenericRouterNoVcs::handle_link_arrival_event()`, `GenericRouterAdaptive::handle_link_arrival_event()`, `GenericLink::handle_link_arrival_event()`, `NI::handle_new_packet_event()`, `GenericVcsInterface::handle_new_packet_event()`, `GenericTPG::handle_new_packet_event()`, `GenericRPG::handle_new_packet_event()`, `GenericInterface::handle_new_packet_event()`, `NI::handle_out_pull_event()`, `GenericTPG::handle_out_pull_event()`, `GenericSink::handle_outpull_event()`, `GenericVcsInterface::handle_ready_event()`, `GenericInterface::handle_ready_event()`, `RouterFourStageVcs::handle_tick_event()`, `GenericVcsInterface::handle_tick_event()`, `GenericRouterNoVcs::handle_tick_event()`, `GenericRouterAdaptive::handle_tick_event()`, `GenericInterface::handle_tick_event()`, `GenericRouterNoVcs::init()`, `GenericVcArbiter::is_requested()`, `GenericVcArbiter::pick_winner()`, `RouterFourStageVcs::process_event()`, `GenericRouterVct::process_event()`, `GenericRouterNoVcs::process_event()`, `GenericRouterAdaptive::process_event()`, `GenericVcArbiter::pull_winner()`, `GenericRC::push()`, `GenericRC::route_negative_first()`, `GenericRC::route_north_last()`, `GenericRC::route_north_last_non_minimal()`, `GenericRC::route_west_first()`, `GenericRC::route_x_y()`, `GenericRouterVct::send_credit_back()`, `GenericRouterNoVcs::send_credit_back()`, and `GenericRouterAdaptive::send_credit_back()`.

8.33.1.2 #define _DBG_NOARG(fmt) LOC printf(fmt);

Definition at line 38 of file genericComponentHeader.h.

Referenced by `PToPSwitchArbiter::do_fcfs_arbitration()`, `PToPSwitchArbiter::do_priority_round_robin_arbitration()`, `PToPSwitchArbiter::do_round_robin_arbitration()`, `GenericRouterVct::do_switch_traversal()`, `GenericRouterNoVcs::do_switch_traversal()`, `GenericRouterAdaptive::do_switch_traversal()`, `NI::handle_new_packet_event()`, `GenericRPG::handle_ready_event()`, `GenericFlatMc::handle_ready_event()`, `GenericRouterVct::handle_tick_event()`, `GenericRouterNoVcs::handle_tick_event()`, `GenericRouterAdaptive::handle_tick_event()`, and `GenericRC::push()`.

8.33.1.3 #define CREDIT_ID 9801

Definition at line 27 of file genericComponentHeader.h.

Referenced by MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), GenericRouterVct::send_credit_back(), GenericRouterNoVcs::send_credit_back(), and GenericRouterAdaptive::send_credit_back().

8.33.1.4 #define DEFAULT_ADDRESS 0

Definition at line 23 of file genericComponentHeader.h.

8.33.1.5 #define DEFAULT_CONVERT_PACKET_CYCLES 1

Definition at line 24 of file genericComponentHeader.h.

8.33.1.6 #define FLIT_ID 9800

Definition at line 26 of file genericComponentHeader.h.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_link_arrival(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), and GenericInterface::handle_tick_event().

8.33.1.7 #define LOC std::cout << "\nTime:" << dec << Simulator::Now() << " " << name << " " << address << " " << node_ip << " ";

Definition at line 36 of file genericComponentHeader.h.

8.33.1.8 #define NO_DATA true

Definition at line 25 of file genericComponentHeader.h.

8.33.2 Typedef Documentation**8.33.2.1 typedef unsigned long long int simTime**

Definition at line 44 of file genericComponentHeader.h.

8.33.2.2 typedef unsigned int uint

Definition at line 46 of file genericComponentHeader.h.

8.33.2.3 `typedef unsigned long long int ullint`

Definition at line 45 of file genericComponentHeader.h.

8.33.2.4 `typedef unsigned long int uniqueId`

Definition at line 43 of file genericComponentHeader.h.

8.33.3 Enumeration Type Documentation**8.33.3.1 `enum message_class`**

Enumerator:

INVALID_PKT
REQUEST_PKT
RESPONSE_PKT
ONE_FLIT_REQ

Definition at line 47 of file genericComponentHeader.h.

8.33.4 Variable Documentation**8.33.4.1 `const unsigned int max_control_bits = 8`**

Definition at line 52 of file genericComponentHeader.h.

8.33.4.2 `const unsigned int max_network_node_bits = 8`

Definition at line 49 of file genericComponentHeader.h.

Referenced by HeadFlit::populate_head_flit().

8.33.4.3 `const unsigned int max_tail_length_bits = 8`

Definition at line 51 of file genericComponentHeader.h.

Referenced by TailFlit::populate_tail_flit().

8.33.4.4 `const unsigned int max_transaction_id_bits = 8`

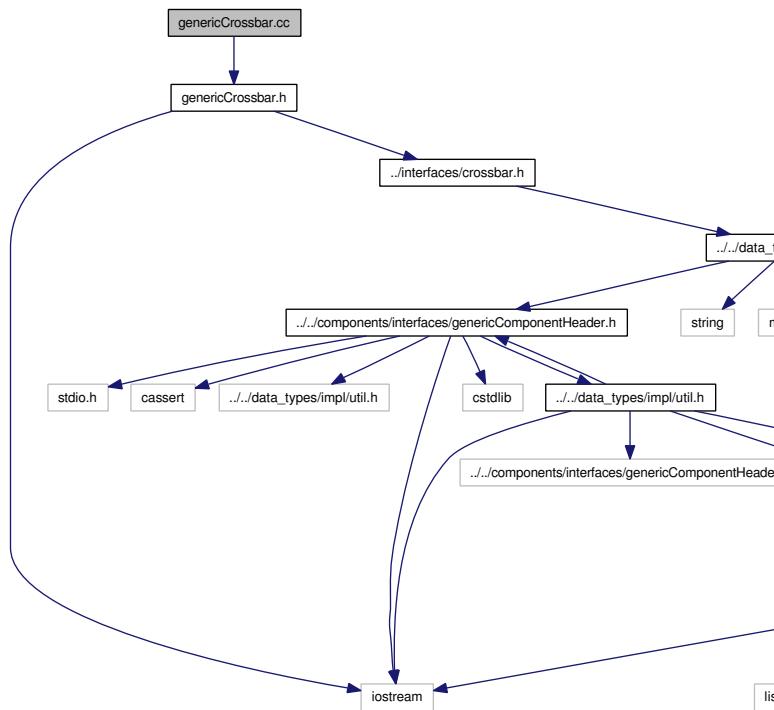
Definition at line 50 of file genericComponentHeader.h.

Referenced by HeadFlit::populate_head_flit().

8.34 genericCrossbar.cc File Reference

```
#include "genericCrossbar.h"
```

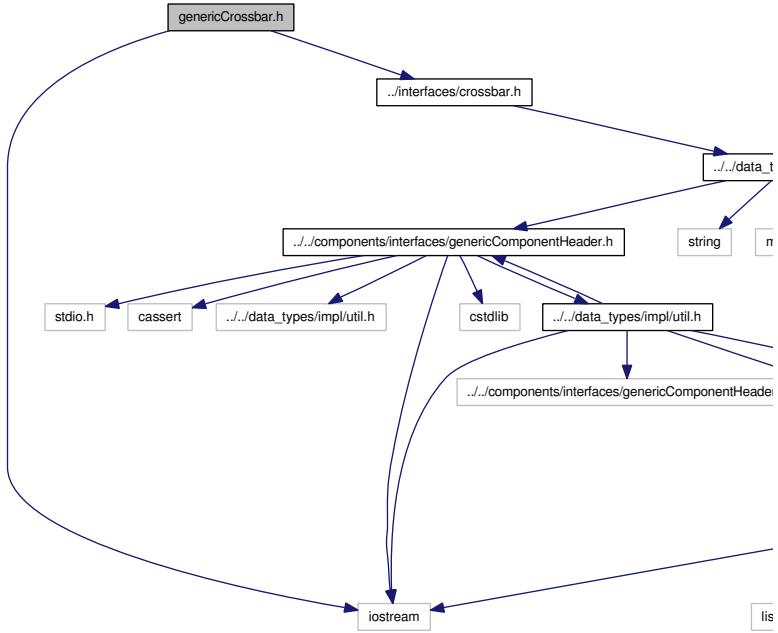
Include dependency graph for genericCrossbar.cc:



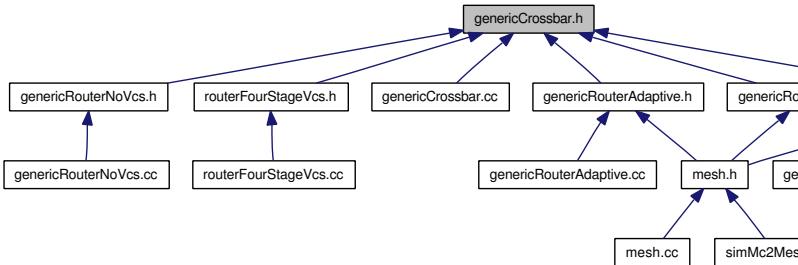
8.35 genericCrossbar.h File Reference

```
#include "../interfaces/crossbar.h"
#include <iostream>
```

Include dependency graph for genericCrossbar.h:



This graph shows which files directly or indirectly include this file:



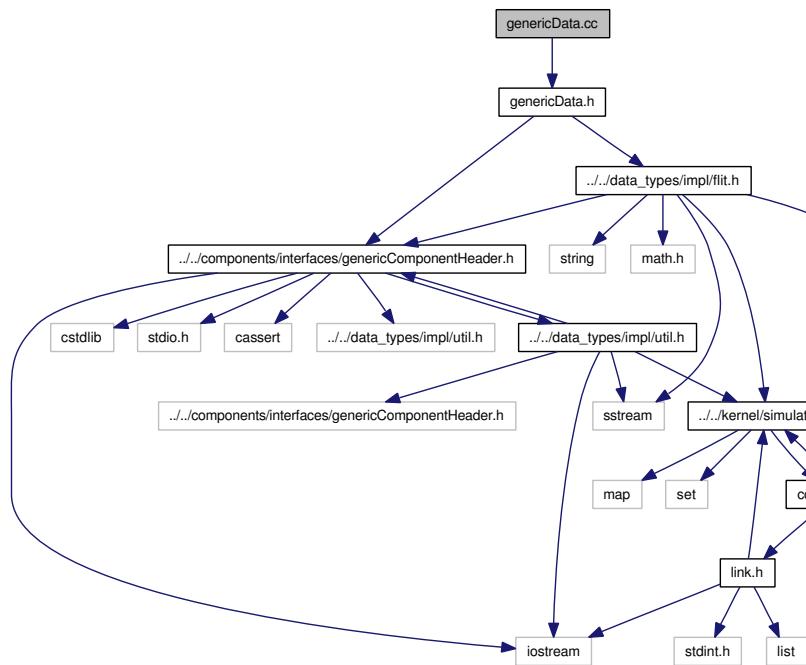
Classes

- class **GenericCrossbar**
- class **GenericCrossbar::GenericCrossbar::CrossbarUnit**

8.36 genericData.cc File Reference

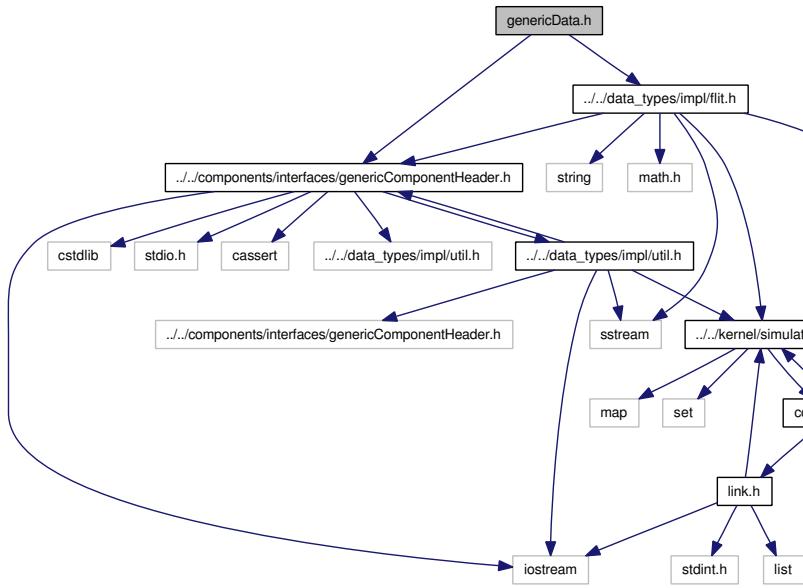
```
#include "genericData.h"
```

Include dependency graph for genericData.cc:

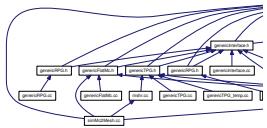


8.37 genericData.h File Reference

```
#include "../../data_types/impl/flit.h"
#include "../interfaces/genericComponentHeader.h"
Include dependency graph for genericData.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **InputBufferState**
- class **LinkArrivalData**
- class **VirtualChannelDescription**
- class **RouteEntry**
- class **SA_unit**

Enumerations

- enum **ROUTING_SCHEME** {
 XY, NEGATIVE_FIRST, VALIANT, WEST_FIRST,
 NORTH_LAST, NORTH_LAST_NON_MINIMAL }
- enum **SW_ARBITRATION** { ROUND_ROBIN, FCFS, ROUND_ROBIN_PRIORITY }
- enum **RouterPipeStage** {
 INVALID, EMPTY, IB, FULL,
 ROUTED, SWA_REQUESTED, SW_ALLOCATED, ST,
 REQ_OUTVC_ARB }

8.37.1 Enumeration Type Documentation

8.37.1.1 enum RouterPipeStage

Enumerator:

INVALID
EMPTY
IB
FULL
ROUTED
SWA_REQUESTED
SW_ALLOCATED
ST
REQ_OUTVC_ARB

Definition at line 28 of file genericData.h.

8.37.1.2 enum ROUTING_SCHEME

Enumerator:

XY
NEGATIVE_FIRST
VALIANT
WEST_FIRST
NORTH_LAST
NORTH_LAST_NON_MINIMAL

Definition at line 26 of file genericData.h.

8.37.1.3 enum SW_ARBITRATION

Enumerator:

ROUND_ROBIN
FCFS
ROUND_ROBIN_PRIORITY

Definition at line 27 of file genericData.h.

8.38 genericEvents.h File Reference

This graph shows which files directly or indirectly include this file:



Defines

- #define OLD_PACKET_EVENT 999
- #define NEW_PACKET_EVENT 1000
- #define READY_EVENT 1101
- #define CREDIT_EVENT 1102
- #define TICK_EVENT 1103
- #define LINK_ARRIVAL_EVENT 1104
- #define SWAP_VC_EVENT 1105
- #define IN_PULL_EVENT 1106
- #define OUT_PULL_EVENT 1107
- #define IN_PUSH_EVENT 1108
- #define OUT_PUSH_EVENT 1109
- #define VC_ARBITRATE_EVENT 1110
- #define IN_ARBITRATE_EVENT 1111
- #define OUT_ARBITRATE_EVENT 1112
- #define PORT_ARBITRATE_EVENT 1113
- #define ADDRESS_DECODE_EVENT 1114
- #define TRAVERSE_CROSSBAR_EVENT 1115
- #define CONFIGURE_CROSSBAR_EVENT 1116
- #define FLIT_OUT_EVENT 1117
- #define CHECK_IN_ARBITER_EVENT 1118
- #define CHECK_OUT_ARBITER_EVENT 1119
- #define START 1120
- #define STOP 1121
- #define CONTINUE 1122
- #define START_SUBCOMPONENT 1123
- #define STOP_SUBCOMPONENT 1124
- #define START_READ 1125
- #define START_WRITE 1126
- #define PUSH_BUFFER 1127
- #define REPLY 1128
- #define STOP_CMD_QUEUE 1129
- #define START_CMD_QUEUE 1130
- #define IN_BUFFER_EVENT 1131
- #define MSHR_DELETE 1132
- #define SEND_TO_NI 1133

8.38.1 Define Documentation

8.38.1.1 #define ADDRESS_DECODE_EVENT 1114

Definition at line 37 of file genericEvents.h.

8.38.1.2 #define CHECK_IN_ARBITER_EVENT 1118

Definition at line 42 of file genericEvents.h.

8.38.1.3 #define CHECK_OUT_ARBITER_EVENT 1119

Definition at line 43 of file genericEvents.h.

8.38.1.4 #define CONFIGURE_CROSSBAR_EVENT 1116

Definition at line 39 of file genericEvents.h.

8.38.1.5 #define CONTINUE 1122

Definition at line 46 of file genericEvents.h.

Referenced by RequestHandler::process_event(), and BankHandler::process_event().

8.38.1.6 #define CREDIT_EVENT 1102

Definition at line 24 of file genericEvents.h.

8.38.1.7 #define FLIT_OUT_EVENT 1117

Definition at line 40 of file genericEvents.h.

Referenced by MCFrontEnd::handle_out_arbitrate_event(), and MCFrontEnd::process_event().

8.38.1.8 #define IN_ARBITRATE_EVENT 1111

Definition at line 34 of file genericEvents.h.

Referenced by MCFrontEnd::check_input_conditions(), and MCFrontEnd::process_event().

8.38.1.9 #define IN_BUFFER_EVENT 1131

Definition at line 55 of file genericEvents.h.

8.38.1.10 #define IN_PULL_EVENT 1106

Definition at line 28 of file genericEvents.h.

8.38.1.11 #define IN_PUSH_EVENT 1108

Definition at line 30 of file genericEvents.h.

Referenced by MCFrontEnd::check_input_conditions(), and MCFrontEnd::process_event().

8.38.1.12 #define LINK_ARRIVAL_EVENT 1104

Definition at line 26 of file genericEvents.h.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterNoVcs::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericInterface::handle_tick_event(), RouterFourStageVcs::process_event(), MCFrontEnd::process_event(), GenericVcsInterface::process_event(), GenericRouterVct::process_event(), GenericRouterNoVcs::process_event(), GenericRouterAdaptive::process_event(), GenericLink::process_event(), GenericInterface::process_event(), GenericRouterVct::send_credit_back(), GenericRouterNoVcs::send_credit_back(), and GenericRouterAdaptive::send_credit_back().

8.38.1.13 #define MSHR_DELETE 1132

Definition at line 56 of file genericEvents.h.

Referenced by GenericTPG::handle_new_packet_event(), and MSHR_H::process_event().

8.38.1.14 #define NEW_PACKET_EVENT 1000

Definition at line 22 of file genericEvents.h.

Referenced by NI::handle_out_pull_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), GenericSink::handle_outpull_event(), GenericFlatMc::handle_ready_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), RequestHandler::process_event(), NI::process_event(), MCFrontEnd::process_event(), GenericVcsInterface::process_event(), GenericTPG::process_event(), GenericSink::process_event(), GenericRPG::process_event(), GenericInterface::process_event(), and GenericFlatMc::process_event().

8.38.1.15 #define OLD_PACKET_EVENT 999

Definition at line 21 of file genericEvents.h.

8.38.1.16 #define OUT_ARBITRATE_EVENT 1112

Definition at line 35 of file genericEvents.h.

Referenced by MCFrontEnd::check_tick(), MCFrontEnd::handle_link_arrival(), and MCFrontEnd::process_event().

8.38.1.17 #define OUT_PULL_EVENT 1107

Definition at line 29 of file genericEvents.h.

Referenced by GenericFlatMc::handle_new_packet_event(), GenericTPG::handle_out_pull_event(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), NI::handle_ready_event(), GenericTPG::handle_ready_event(), GenericSink::handle_ready_event(), GenericRPG::handle_ready_event(),

GenericFlatMc::handle_ready_event(), ResponseHandler::process_event(), NI::process_event(), MSHR_H::process_event(), GenericTPG::process_event(), GenericSink::process_event(), GenericRPG::process_event(), and GenericFlatMc::process_event().

8.38.1.18 #define OUT_PUSH_EVENT 1109

Definition at line 31 of file genericEvents.h.

8.38.1.19 #define PORT_ARBITRATE_EVENT 1113

Definition at line 36 of file genericEvents.h.

8.38.1.20 #define PUSH_BUFFER 1127

Definition at line 51 of file genericEvents.h.

Referenced by ResponseHandler::process_event(), and BankHandler::process_event().

8.38.1.21 #define READY_EVENT 1101

Definition at line 23 of file genericEvents.h.

Referenced by NI::handle_new_packet_event(), GenericTPG::handle_new_packet_event(), GenericSink::handle_new_packet_event(), GenericRPG::handle_new_packet_event(), GenericFlatMc::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), NI::process_event(), MCFrontEnd::process_event(), GenericVcsInterface::process_event(), GenericTPG::process_event(), GenericSink::process_event(), GenericRPG::process_event(), GenericInterface::process_event(), GenericFlatMc::process_event(), NI::setup(), MCFrontEnd::setup(), GenericTPG::setup(), GenericSink::setup(), GenericRPG::setup(), and GenericFlatMc::setup().

8.38.1.22 #define REPLY 1128

Definition at line 52 of file genericEvents.h.

Referenced by ResponseHandler::process_event(), and DataBusHandler::process_event().

8.38.1.23 #define SEND_TO_NI 1133

Definition at line 57 of file genericEvents.h.

Referenced by NI::handle_out_pull_event(), and ResponseHandler::process_event().

8.38.1.24 #define START 1120

Definition at line 44 of file genericEvents.h.

Referenced by NI::handle_new_packet_event(), main(), RequestHandler::process_event(), RefreshMgr::process_event(), CmdIssuer::process_event(), BankHandler::process_event(), and AddrMap::process_event().

8.38.1.25 #define START_CMD_QUEUE 1130

Definition at line 54 of file genericEvents.h.

Referenced by RequestHandler::process_event().

8.38.1.26 #define START_READ 1125

Definition at line 49 of file genericEvents.h.

Referenced by DRAMChannel::process_event(), DataBusHandler::process_event(), and CmdBusHandler::process_event().

8.38.1.27 #define START_SUBCOMPONENT 1123

Definition at line 47 of file genericEvents.h.

8.38.1.28 #define START_WRITE 1126

Definition at line 50 of file genericEvents.h.

Referenced by DRAMChannel::process_event(), DataBusHandler::process_event(), and CmdIssuer::process_event().

8.38.1.29 #define STOP 1121

Definition at line 45 of file genericEvents.h.

Referenced by BankHandler::process_event().

8.38.1.30 #define STOP_CMD_QUEUE 1129

Definition at line 53 of file genericEvents.h.

Referenced by RequestHandler::process_event().

8.38.1.31 #define STOP_SUBCOMPONENT 1124

Definition at line 48 of file genericEvents.h.

8.38.1.32 #define SWAP_VC_EVENT 1105

Definition at line 27 of file genericEvents.h.

8.38.1.33 #define TICK_EVENT 1103

Definition at line 25 of file genericEvents.h.

Referenced by GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), RouterFourStageVcs::handle_link_arrival_event(), GenericRouterVct::handle_link_arrival_event(), GenericRouterNoVcs::handle_link_arrival_event(), GenericRouterAdaptive::handle_link_arrival_event(), GenericVcsInterface::handle_new_packet_event(), GenericInterface::handle_new_packet_event(), GenericVcsInterface::handle_ready_event(), GenericInterface::handle_ready_event(),

RouterFourStageVcs::handle_tick_event(), GenericVcsInterface::handle_tick_event(), GenericRouterVct::handle_tick_event(), GenericRouterNoVcs::handle_tick_event(), GenericRouterAdaptive::handle_tick_event(), GenericInterface::handle_tick_event(), RouterFourStageVcs::process_event(), GenericVcsInterface::process_event(), GenericRouterVct::process_event(), GenericRouterNoVcs::process_event(), GenericRouterAdaptive::process_event(), and GenericInterface::process_event().

8.38.1.34 #define TRAVERSE_CROSSBAR_EVENT 1115

Definition at line 38 of file genericEvents.h.

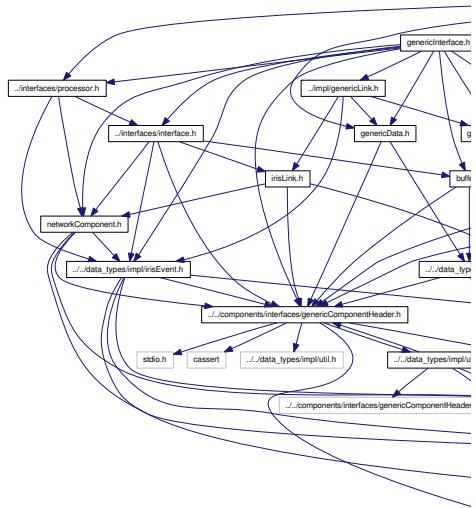
8.38.1.35 #define VC_ARBITRATE_EVENT 1110

Definition at line 32 of file genericEvents.h.

8.39 genericFlatMc.cc File Reference

```
#include "genericFlatMc.h"
```

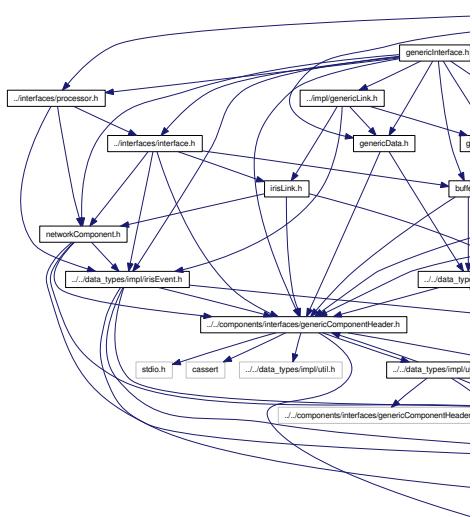
Include dependency graph for genericFlatMc.cc:



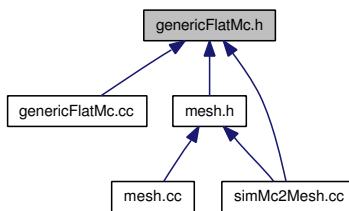
8.40 genericFlatMc.h File Reference

```
#include "genericInterface.h"
#include "genericEvents.h"
#include "genericData.h"
#include "../interfaces/processor.h"
#include "../../data_types/impl/highLevelPacket.h"
#include "../../MemCtrl/request.h"
#include "../../tests/MersenneTwister.h"
#include <fstream>
#include <deque>
```

Include dependency graph for genericFlatMc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericFlatMc**

Defines

- #define **DEFAULT_RAN_MAX_TIME** 100
- #define **MAX_ADDRESS** 3

– `#define MAX(a, b) (((a)<(b))?(b):(a))`
– `#define MIN(a, b) (((a)<(b))?(a):(b))`

8.40.1 Define Documentation

8.40.1.1 `#define DEFAULT_RAN_MAX_TIME 100`

Definition at line 15 of file genericFlatMc.h.

8.40.1.2 `#define MAX(a, b) (((a)<(b))?(b):(a))`

Definition at line 17 of file genericFlatMc.h.

8.40.1.3 `#define MAX_ADDRESS 3`

Definition at line 16 of file genericFlatMc.h.

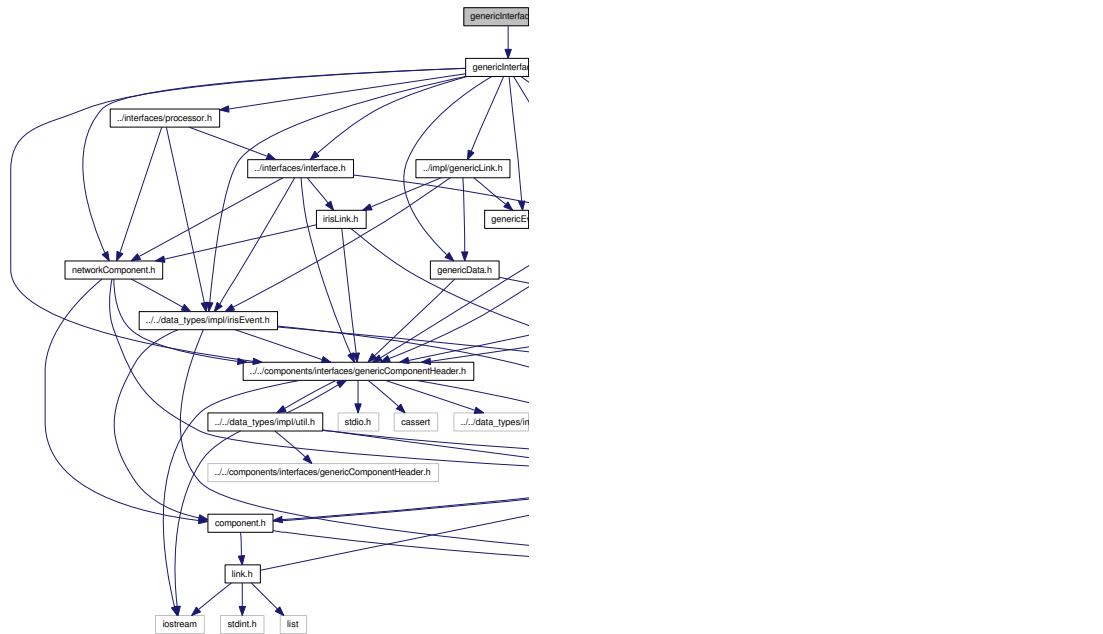
8.40.1.4 `#define MIN(a, b) (((a)<(b))?(a):(b))`

Definition at line 18 of file genericFlatMc.h.

8.41 genericInterface.cc File Reference

```
#include "genericInterface.h"
```

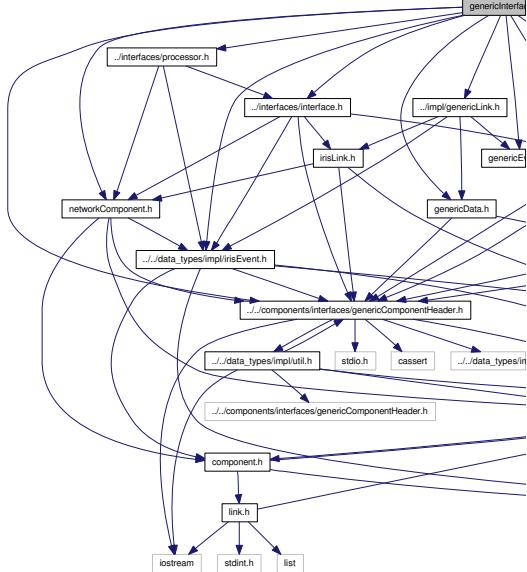
Include dependency graph for genericInterface.cc:



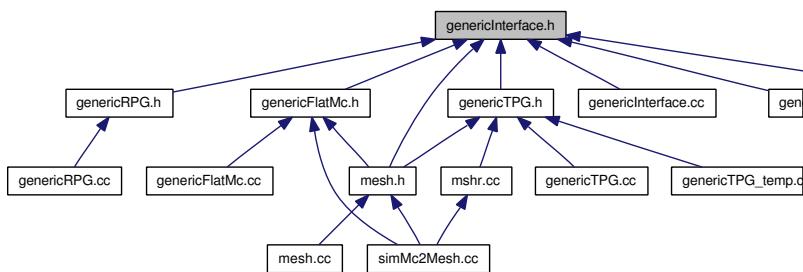
8.42 genericInterface.h File Reference

```
#include "../../data_types/impl/highLevelPacket.h"
#include "../../data_types/impl/irisEvent.h"
#include "../interfaces/interface.h"
#include "../interfaces/networkComponent.h"
#include "../interfaces/processor.h"
#include "../interfaces/buffer.h"
#include "../interfaces/genericComponentHeader.h"
#include "../impl/genericLink.h"
#include "genericEvents.h"
#include "genericBuffer.h"
#include "genericData.h"
#include <queue>
#include <vector>
#include <math.h>
```

Include dependency graph for genericInterface.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericInterface**

Defines

- `#define DEFAULT_NO_OF_CREDITS 1`

Variables

- `uint do_two_stage_router`

8.42.1 Define Documentation

8.42.1.1 `#define DEFAULT_NO_OF_CREDITS 1`

Definition at line 36 of file genericInterface.h.

8.42.2 Variable Documentation

8.42.2.1 `uint do_two_stage_router`

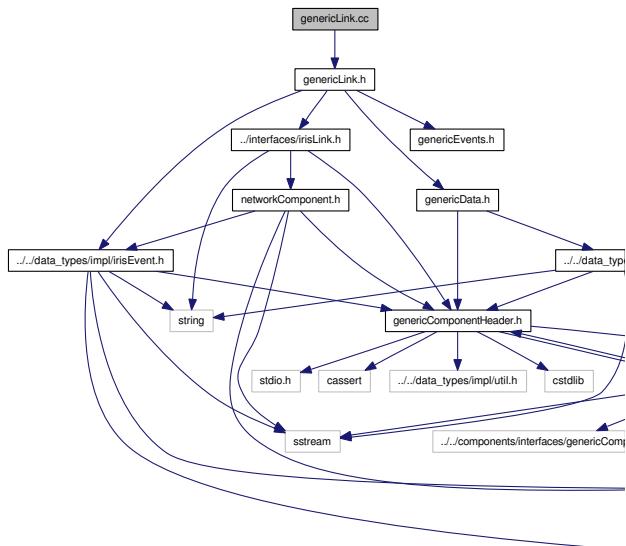
Definition at line 38 of file simMc2Mesh.cc.

Referenced by `GenericRouterVct::do_switch_traversal()`, `GenericRouterAdaptive::do_switch_traversal()`, `GenericVcsInterface::handle_link_arrival()`, `GenericInterface::handle_link_arrival()`, `GenericVcsInterface::handle_tick_event()`, `GenericInterface::handle_tick_event()`, `main()`, `GenericRouterVct::send_credit_back()`, and `GenericRouterAdaptive::send_credit_back()`.

8.43 genericLink.cc File Reference

```
#include "genericLink.h"
```

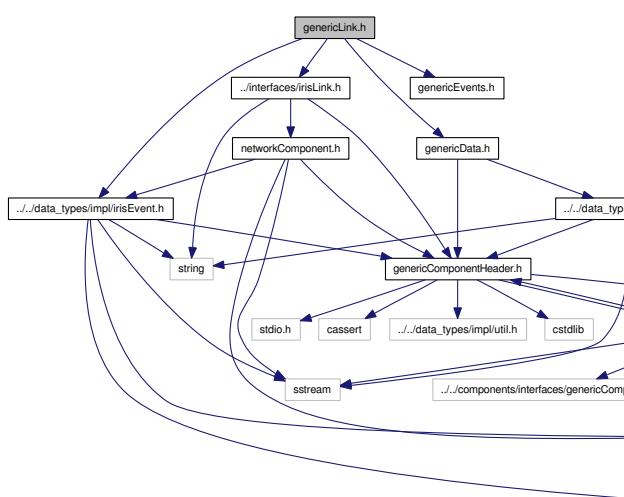
Include dependency graph for genericLink.cc:



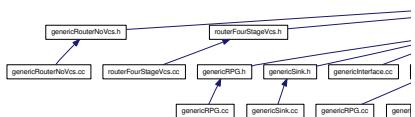
8.44 genericLink.h File Reference

```
#include "../interfaces/irisLink.h"
#include "genericData.h"
#include "genericEvents.h"
#include "../../data_types/impl/irisEvent.h"

Include dependency graph for genericLink.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericLink**

Variables

- **ullint max_sim_time**

8.44.1 Variable Documentation

8.44.1.1 **ullint max_sim_time**

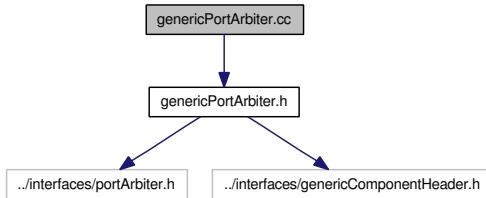
Definition at line 34 of file simMc2Mesh.cc.

Referenced by main(), GenericLink::print_stats(), and MSHR_H::process_event().

8.45 genericPortArbiter.cc File Reference

```
#include "genericPortArbiter.h"
```

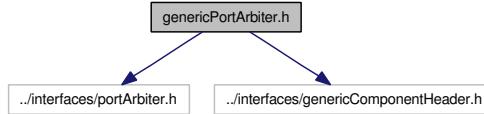
Include dependency graph for genericPortArbiter.cc:



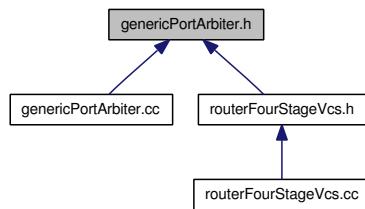
8.46 genericPortArbiter.h File Reference

```
#include "../interfaces/portArbiter.h"
#include "../interfaces/genericComponentHeader.h"
```

Include dependency graph for genericPortArbiter.h:



This graph shows which files directly or indirectly include this file:



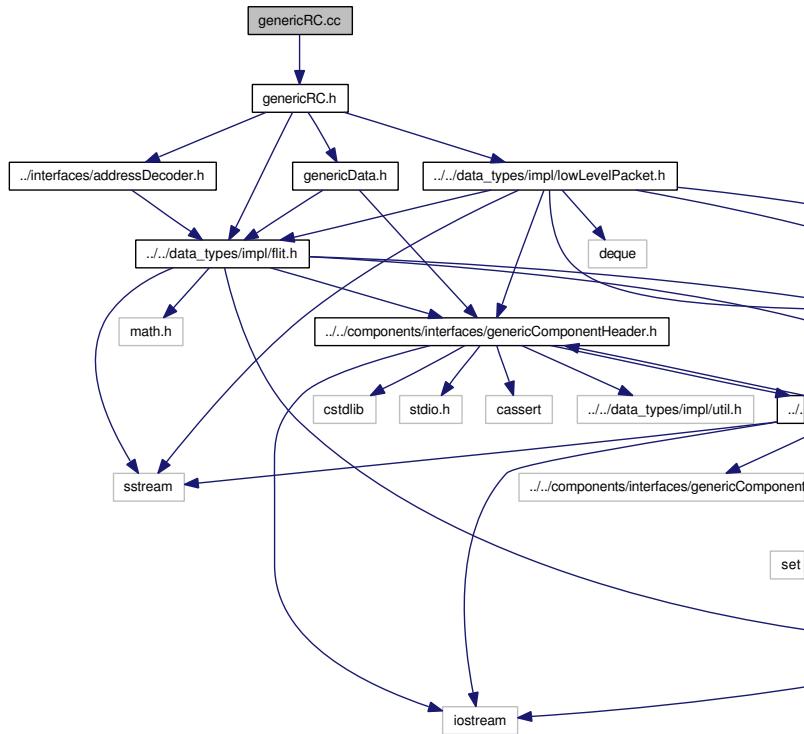
Classes

- class **GenericPortArbiter**

8.47 genericRC.cc File Reference

```
#include "genericRC.h"
```

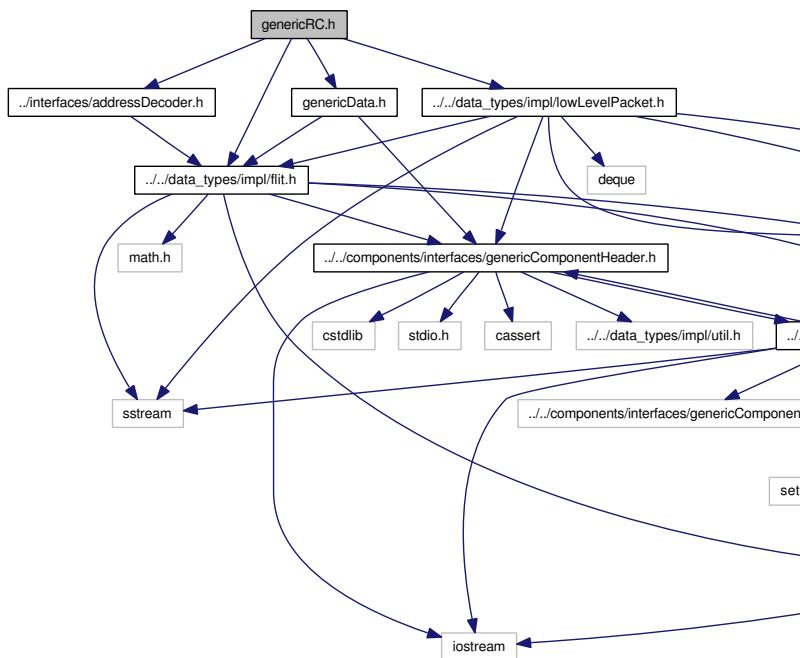
Include dependency graph for genericRC.cc:



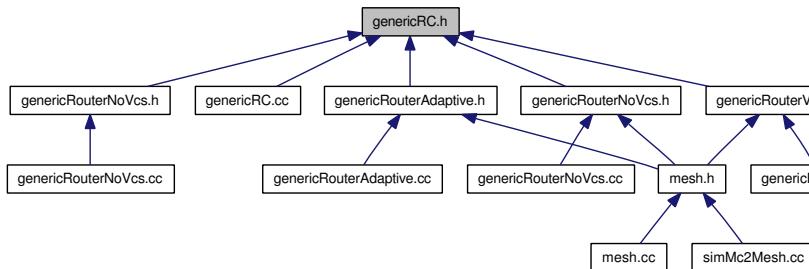
8.48 genericRC.h File Reference

```
#include "../interfaces/addressDecoder.h"
#include "genericData.h"
#include "../../data_types/impl/flit.h"
#include "../../data_types/impl/lowLevelPacket.h"

Include dependency graph for genericRC.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericRC**
- class **GenericRC::GenericRC::Address**

Variables

- **ROUTING_SCHEME rc_method**
- **uint grid_size**
- **uint no_nodes**

8.48.1 Variable Documentation

8.48.1.1 `uint grid_size`

Definition at line 43 of file simMc2Mesh.cc.

Referenced by `main()`, and `GenericRC::route_north_last_non_minimal()`.

8.48.1.2 `uint no_nodes`

Definition at line 33 of file simMc2Mesh.cc.

Referenced by `main()`, and `GenericRC::route_north_last_non_minimal()`.

8.48.1.3 `ROUTING_SCHEME rc_method`

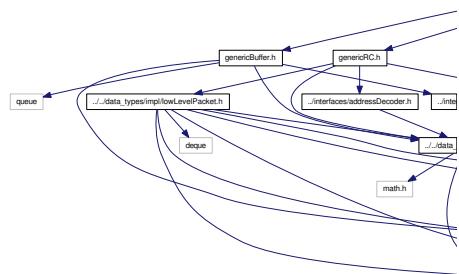
Definition at line 40 of file simMc2Mesh.cc.

Referenced by `main()`, and `GenericRC::push()`.

8.49 genericRouterAdaptive.cc File Reference

```
#include "genericRouterAdaptive.h"
```

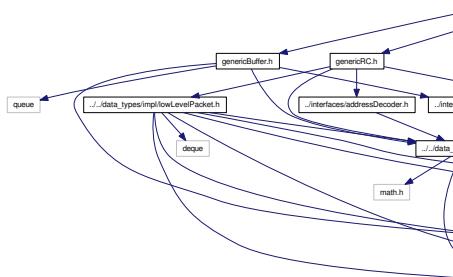
Include dependency graph for genericRouterAdaptive.cc:



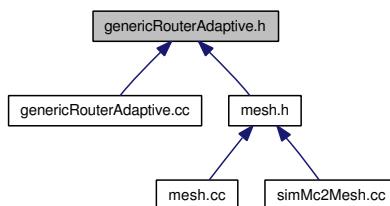
8.50 genericRouterAdaptive.h File Reference

```
#include "../interfaces/router.h"
#include "genericBuffer.h"
#include "genericRC.h"
#include "ptop_swa.h"
#include "genericCrossbar.h"
#include "genericEvents.h"
#include "genericData.h"
#include "genericLink.h"
#include <sys/time.h>
#include <algorithm>
```

Include dependency graph for genericRouterAdaptive.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericRouterAdaptive**

Variables

- uint **send_early_credit**
- uint **do_two_stage_router**

8.50.1 Variable Documentation

8.50.1.1 uint do_two_stage_router

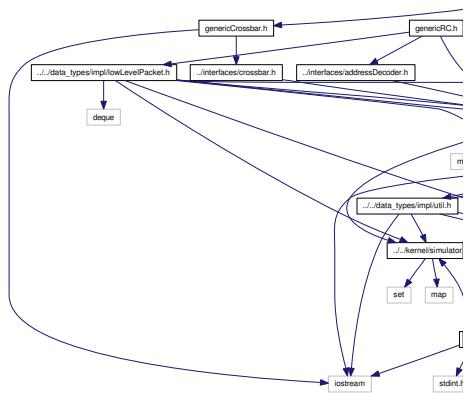
Definition at line 38 of file simMc2Mesh.cc.

8.50.1.2 uint send_early_credit

8.51 genericRouterNoVcs.cc File Reference

```
#include "genericRouterNoVcs.h"
```

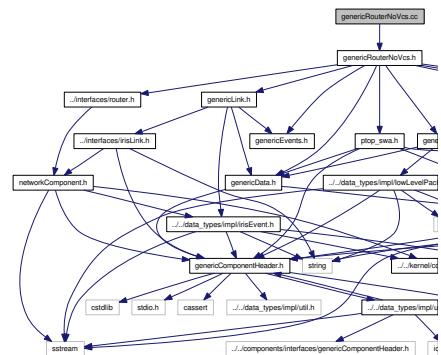
Include dependency graph for backup/genericRouterNoVcs.cc:



8.52 genericRouterNoVcs.cc File Reference

```
#include "genericRouterNoVcs.h"
```

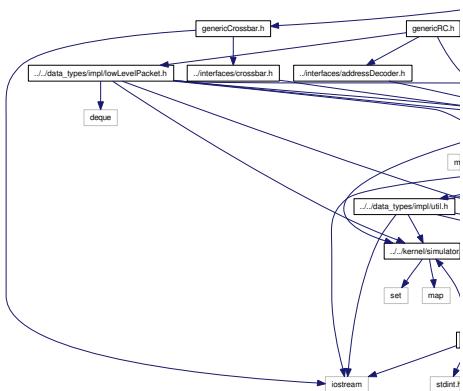
Include dependency graph for genericRouterNoVcs.cc:



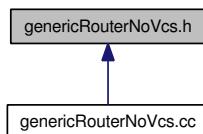
8.53 genericRouterNoVcs.h File Reference

```
#include "../interfaces/router.h"
#include "genericBuffer.h"
#include "genericRC.h"
#include "genericVcArbiter.h"
#include "myArbiter.h"
#include "genericCrossbar.h"
#include "genericEvents.h"
#include "genericData.h"
#include "genericLink.h"
```

Include dependency graph for backup/genericRouterNoVcs.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **MessageState**
- class **GenericRouterNoVcs**

Enumerations

- enum **GenericRouterNoVcsPipeStage** {
 INVALID, EMPTY, IB, FULL,
 ROUTED, SWA_REQUESTED, SW_ALLOCATED, ST,
 REQ_OUTVC_ARB
 }

8.53.1 Enumeration Type Documentation

8.53.1.1 enum GenericRouterNoVcsPipeStage

Enumerator:

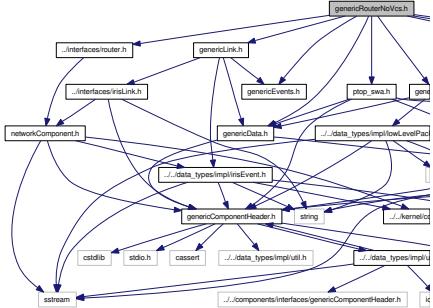
INVALID
EMPTY
IB
FULL
ROUTED
SWA_REQUESTED
SW_ALLOCATED
ST
REQ_OUTVC_ARB

Definition at line 32 of file backup/genericRouterNoVcs.h.

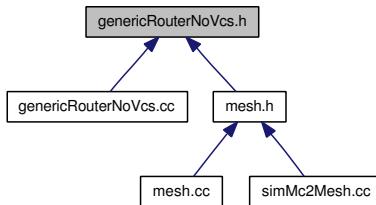
8.54 genericRouterNoVcs.h File Reference

```
#include "../interfaces/router.h"
#include "genericBuffer.h"
#include "genericRC.h"
#include "genericVcArbiter.h"
#include "ptop_swa.h"
#include "genericCrossbar.h"
#include "genericEvents.h"
#include "genericData.h"
#include "genericLink.h"
```

Include dependency graph for genericRouterNoVcs.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericRouterNoVcs**

Variables

- uint `send_early_credit`

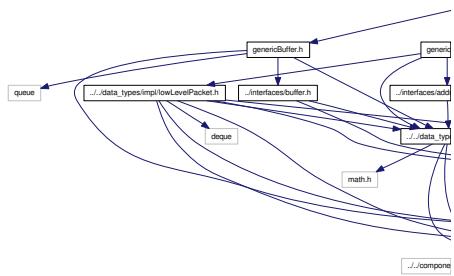
8.54.1 Variable Documentation

8.54.1.1 uint `send_early_credit`

8.55 genericRouterVct.cc File Reference

```
#include "genericRouterVct.h"
```

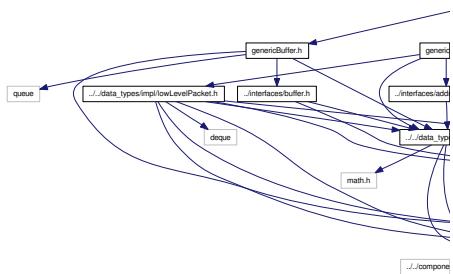
Include dependency graph for genericRouterVct.cc:



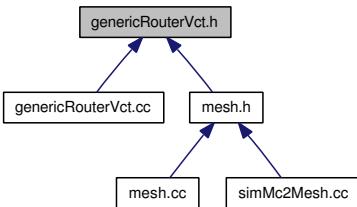
8.56 genericRouterVct.h File Reference

```
#include "../interfaces/router.h"
#include "genericBuffer.h"
#include "genericRC.h"
#include "ptop_swa.h"
#include "genericCrossbar.h"
#include "genericEvents.h"
#include "genericData.h"
#include "genericLink.h"
#include <sys/time.h>
```

Include dependency graph for genericRouterVct.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericRouterVct**

Variables

- uint send_early_credit
 - uint do_two_stage_router

8.56.1 Variable Documentation

8.56.1.1 uint do two stage router

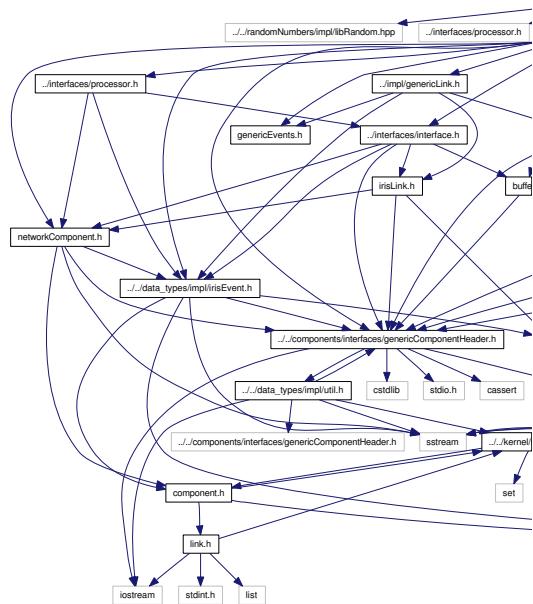
Definition at line 38 of file simMc2Mesh.cc.

8.56.1.2 uint send_early_credit

8.57 genericRPG.cc File Reference

```
#include "genericRPG.h"
```

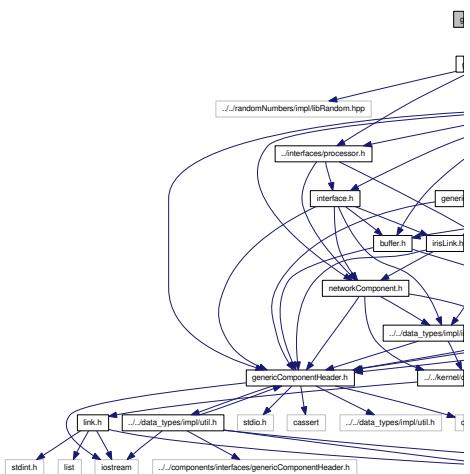
Include dependency graph for backup/genericRPG.cc:



8.58 genericRPG.cc File Reference

```
#include "genericRPG.h"
```

Include dependency graph for genericRPG.cc:



Defines

```
- #define MAX_SIM_TIME 2000000000
```

8.58.1 Define Documentation

8.58.1.1 #define MAX_SIM_TIME 2000000000

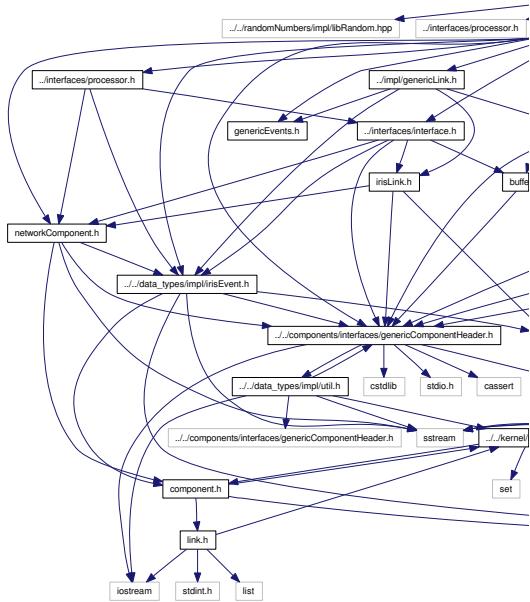
Definition at line 4 of file genericRPG.cc.

Referenced by GenericRPG::handle_out_pull_event(), and GenericRPG::handle_ready_event().

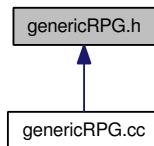
8.59 genericRPG.h File Reference

```
#include "../../randomNumbers/impl/libRandom.hpp"
#include "../interfaces/processor.h"
#include "genericInterface.h"
#include "genericData.h"
#include <fstream>
#include <deque>
```

Include dependency graph for backup/genericRPG.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericRPG**

Defines

- #define IN_OUT_MISMATCH -1
- #define SIM_SUCCESS 0
- #define REPORT_BASE 20
- #define DEFAULT_RAN_LAMDA 2.35
- #define DEFAULT_RAN_DESTINATION_TYPE "uniform"
- #define DEFAULT_RAN_LENGTH_TYPE "uniform"

```
- #define DEFAULT_RAN_ADDRESS 0
- #define MAX_ADDRESS 3
- #define MAX_LENGTH 10
- #define MIN_LENGTH 1
- #define MIN_DELAY 1
- #define MAX_DELAY 100
- #define DEFAULT_RAN_MAX_VC 1
- #define DEFAULT_RAN_SEED 324986
- #define DEFAULT_RAN_MAX_TIME 100
- #define HOT_SPOTS 3
- #define DEFAULT_RAN_TRACE_FILE_NAME "randomOut.tr"
- #define MAX(a, b) ((a)<(b))?(b):(a))
- #define MIN(a, b) (((a)<(b))?(a):(b))
```

Variables

- const string run_destination_type = "uniform"

8.59.1 Define Documentation

8.59.1.1 #define DEFAULT_RAN_ADDRESS 0

Definition at line 30 of file backup/genericRPG.h.

8.59.1.2 #define DEFAULT_RAN_DESTINATION_TYPE "uniform"

Definition at line 28 of file backup/genericRPG.h.

8.59.1.3 #define DEFAULT_RAN_LAMDA 2.35

Definition at line 27 of file backup/genericRPG.h.

8.59.1.4 #define DEFAULT_RAN_LENGTH_TYPE "uniform"

Definition at line 29 of file backup/genericRPG.h.

8.59.1.5 #define DEFAULT_RAN_MAX_TIME 100

Definition at line 38 of file backup/genericRPG.h.

8.59.1.6 #define DEFAULT_RAN_MAX_VC 1

Definition at line 36 of file backup/genericRPG.h.

8.59.1.7 #define DEFAULT_RAN_SEED 324986

Definition at line 37 of file backup/genericRPG.h.

8.59.1.8 #define DEFAULT_RAN_TRACE_FILE_NAME "randomOut.tr"

Definition at line 40 of file backup/genericRPG.h.

8.59.1.9 #define HOT_SPOTS 3

Definition at line 39 of file backup/genericRPG.h.

Referenced by GenericRPG::init_generator().

8.59.1.10 #define IN_OUT_MISMATCH -1

Definition at line 14 of file backup/genericRPG.h.

8.59.1.11 #define MAX(a, b) (((a)<(b))?(b):(a))

Definition at line 42 of file backup/genericRPG.h.

8.59.1.12 #define MAX_ADDRESS 3

Definition at line 31 of file backup/genericRPG.h.

Referenced by GenericRPG::init_generator().

8.59.1.13 #define MAX_DELAY 100

Definition at line 35 of file backup/genericRPG.h.

Referenced by GenericRPG::init_generator().

8.59.1.14 #define MAX_LENGTH 10

Definition at line 32 of file backup/genericRPG.h.

Referenced by GenericRPG::init_generator().

8.59.1.15 #define MIN(a, b) (((a)<(b))?(a):(b))

Definition at line 43 of file backup/genericRPG.h.

8.59.1.16 #define MIN_DELAY 1

Definition at line 34 of file backup/genericRPG.h.

Referenced by GenericRPG::init_generator().

8.59.1.17 #define MIN_LENGTH 1

Definition at line 33 of file backup/genericRPG.h.

Referenced by GenericRPG::init_generator().

8.59.1.18 #define REPORT_BASE 20

Definition at line 25 of file backup/genericRPG.h.

8.59.1.19 #define SIM_SUCCESS 0

Definition at line 18 of file backup/genericRPG.h.

8.59.2 Variable Documentation

8.59.2.1 const string run_destination_type = "uniform"

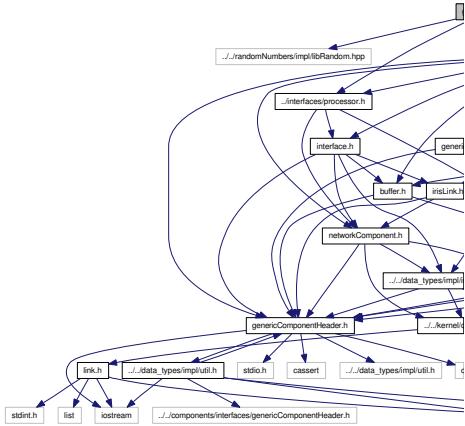
Definition at line 41 of file backup/genericRPG.h.

Referenced by GenericRPG::setup().

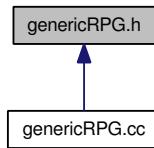
8.60 genericRPG.h File Reference

```
#include "../../randomNumbers/impl/libRandom.hpp"
#include "../interfaces/processor.h"
#include "genericInterface.h"
#include "genericData.h"
#include <fstream>
#include <deque>
```

Include dependency graph for genericRPG.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericRPG**

Defines

```
-#define IN_OUT_MISMATCH -1
#define SIM_SUCCESS 0
#define REPORT_BASE 20
#define DEFAULT_RAN_LAMDA 2.35
#define DEFAULT_RAN_DESTINATION_TYPE "uniform"
#define DEFAULT_RAN_LENGTH_TYPE "uniform"
#define DEFAULT_RAN_ADDRESS 0
#define MAX_ADDRESS 3
#define MAX_LENGTH 10
#define MIN_LENGTH 1
#define MIN_DELAY 1
```

```
- #define MAX_DELAY 100
- #define DEFAULT_RAN_MAX_VC 1
- #define DEFAULT_RAN_SEED 324986
- #define DEFAULT_RAN_MAX_TIME 100
- #define HOT_SPOTS 3
- #define DEFAULT_RAN_TRACE_FILE_NAME "randomOut.tr"
- #define MAX(a, b) (((a)<(b))?(b):(a))
- #define MIN(a, b) (((a)<(b))?(a):(b))
```

Variables

- const string **run_destination_type** = "uniform"

8.60.1 Define Documentation

8.60.1.1 #define DEFAULT_RAN_ADDRESS 0

Definition at line 30 of file genericRPG.h.

8.60.1.2 #define DEFAULT_RAN_DESTINATION_TYPE "uniform"

Definition at line 28 of file genericRPG.h.

8.60.1.3 #define DEFAULT_RAN_LAMDA 2.35

Definition at line 27 of file genericRPG.h.

8.60.1.4 #define DEFAULT_RAN_LENGTH_TYPE "uniform"

Definition at line 29 of file genericRPG.h.

8.60.1.5 #define DEFAULT_RAN_MAX_TIME 100

Definition at line 38 of file genericRPG.h.

8.60.1.6 #define DEFAULT_RAN_MAX_VC 1

Definition at line 36 of file genericRPG.h.

8.60.1.7 #define DEFAULT_RAN_SEED 324986

Definition at line 37 of file genericRPG.h.

8.60.1.8 #define DEFAULT_RAN_TRACE_FILE_NAME "randomOut.tr"

Definition at line 40 of file genericRPG.h.

8.60.1.9 #define HOT_SPOTS 3

Definition at line 39 of file genericRPG.h.

8.60.1.10 #define IN_OUT_MISMATCH -1

Definition at line 14 of file genericRPG.h.

8.60.1.11 #define MAX(a, b) (((a)<(b))?(b):(a))

Definition at line 42 of file genericRPG.h.

8.60.1.12 #define MAX_ADDRESS 3

Definition at line 31 of file genericRPG.h.

8.60.1.13 #define MAX_DELAY 100

Definition at line 35 of file genericRPG.h.

8.60.1.14 #define MAX_LENGTH 10

Definition at line 32 of file genericRPG.h.

8.60.1.15 #define MIN(a, b) (((a)<(b))?(a):(b))

Definition at line 43 of file genericRPG.h.

8.60.1.16 #define MIN_DELAY 1

Definition at line 34 of file genericRPG.h.

8.60.1.17 #define MIN_LENGTH 1

Definition at line 33 of file genericRPG.h.

8.60.1.18 #define REPORT_BASE 20

Definition at line 25 of file genericRPG.h.

8.60.1.19 #define SIM_SUCCESS 0

Definition at line 18 of file genericRPG.h.

8.60.2 Variable Documentation

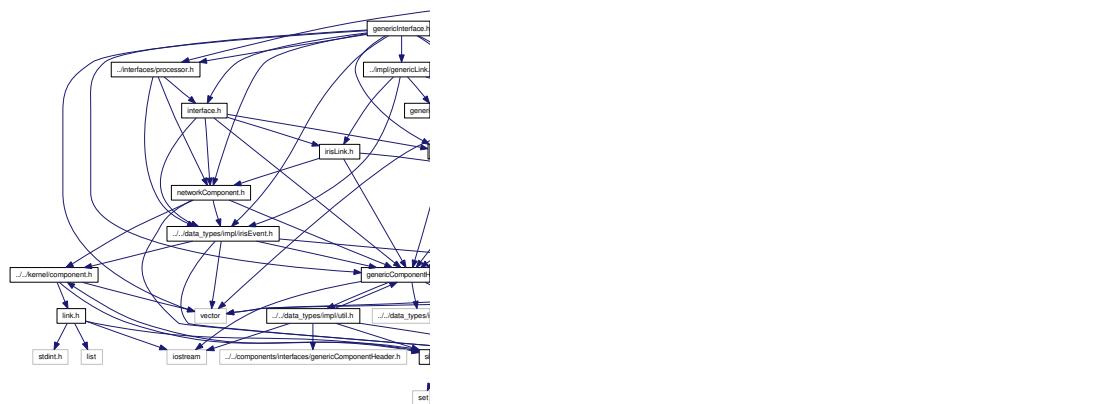
8.60.2.1 const string run_destination_type = "uniform"

Definition at line 41 of file genericRPG.h.

8.61 genericSink.cc File Reference

```
#include "genericSink.h"
```

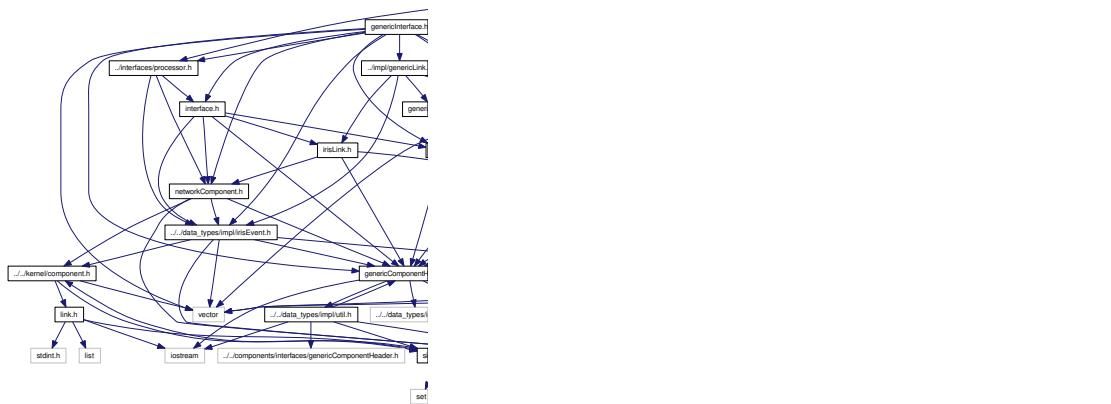
Include dependency graph for genericSink.cc:



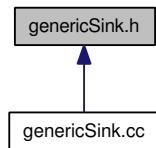
8.62 genericSink.h File Reference

```
#include "../interfaces/processor.h"
#include "../../data_types/impl/highLevelPacket.h"
#include "genericEvents.h"
#include "genericData.h"
#include "genericInterface.h"
#include "../../../tests/MersenneTwister.h"
#include <deque>
#include <fstream>
```

Include dependency graph for genericSink.h:



This graph shows which files directly or indirectly include this file:



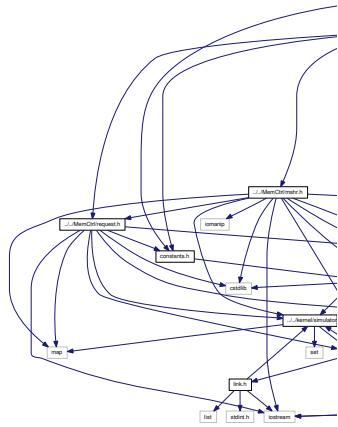
Classes

- class **GenericSink**

8.63 genericTPG.cc File Reference

```
#include "genericTPG.h"
#include "../../MemCtrl/constants.h"
#include <string.h>
```

Include dependency graph for genericTPG.cc:



Variables

- unsigned int MC_ADDR_BITS

8.63.1 Variable Documentation

8.63.1.1 unsigned int MC_ADDR_BITS

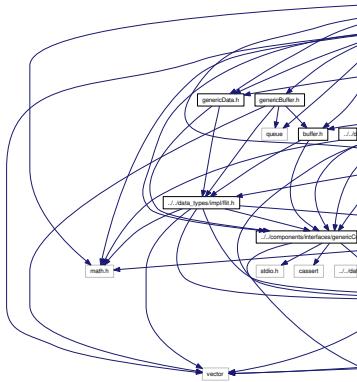
Definition at line 35 of file simMc2Mesh.cc.

Referenced by main(), and MSHR_H::map_addr().

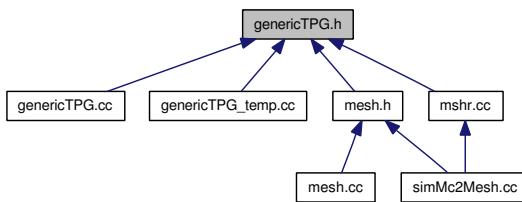
8.64 genericTPG.h File Reference

```
#include "genericInterface.h"
#include "genericEvents.h"
#include "genericData.h"
#include "../interfaces/processor.h"
#include "../../data_types/impl/highLevelPacket.h"
#include "../../MemCtrl/request.h"
#include <fstream>
#include <deque>
#include "../../MemCtrl/mshr.h"
#include "../../MemCtrl/constants.h"
#include <math.h>
```

Include dependency graph for genericTPG.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericTPG**

Defines

- #define **DEFAULT_RAN_MAX_TIME** 100
- #define **MAX_ADDRESS** 3
- #define **MAX(a, b)** (((a)<(b))?(b):(a))
- #define **MIN(a, b)** (((a)<(b))?(a):(b))

Variables

- `uint no_nodes`
- `uint no_mcs`

8.64.1 Define Documentation

8.64.1.1 `#define DEFAULT_RAN_MAX_TIME 100`

Definition at line 17 of file genericTPG.h.

8.64.1.2 `#define MAX(a, b) (((a)<(b))?(b):(a))`

Definition at line 19 of file genericTPG.h.

8.64.1.3 `#define MAX_ADDRESS 3`

Definition at line 18 of file genericTPG.h.

8.64.1.4 `#define MIN(a, b) (((a)<(b))?(a):(b))`

Definition at line 20 of file genericTPG.h.

8.64.2 Variable Documentation

8.64.2.1 `uint no_mcs`

Definition at line 33 of file simMc2Mesh.cc.

Referenced by MSHR_H::countBLP(), main(), and MSHR_H::map_addr().

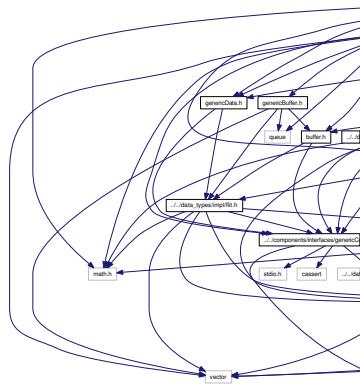
8.64.2.2 `uint no_nodes`

Definition at line 33 of file simMc2Mesh.cc.

8.65 genericTPG_temp.cc File Reference

```
#include "genericTPG.h"
```

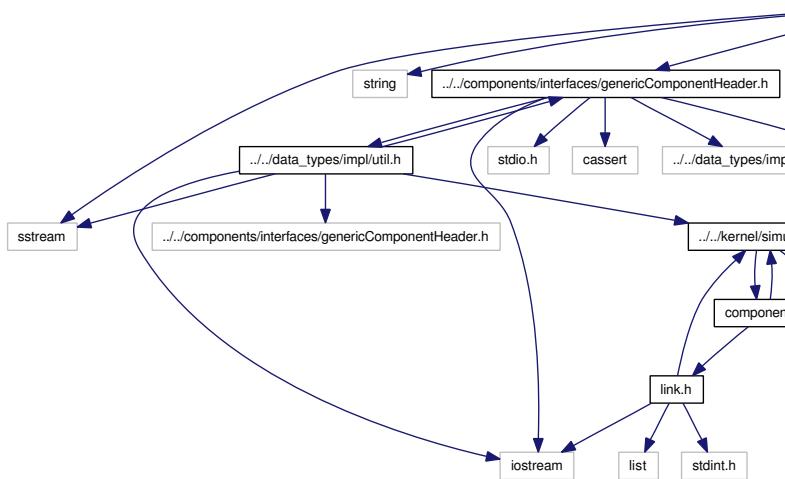
Include dependency graph for genericTPG_temp.cc:



8.66 genericVcArbiter.cc File Reference

```
#include "genericVcArbiter.h"
```

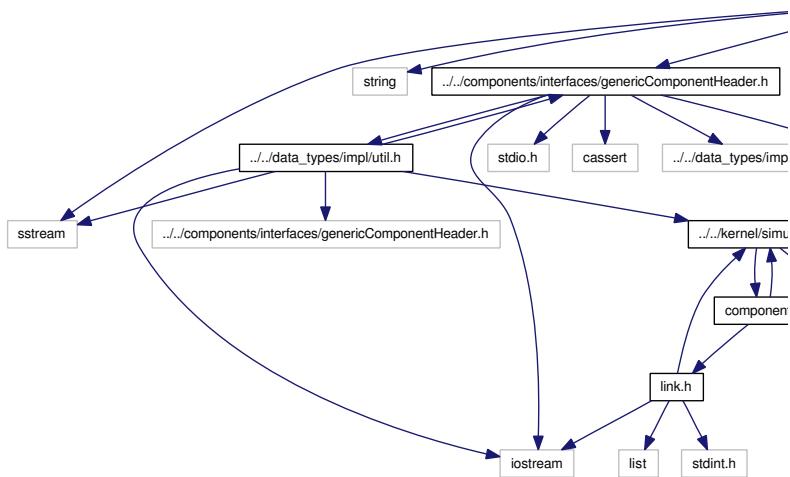
Include dependency graph for genericVcArbiter.cc:



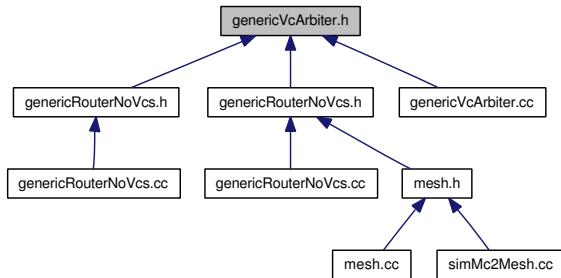
8.67 genericVcArbiter.h File Reference

```
#include "../interfaces/arbiter.h"
#include "../../data_types/impl/flit.h"
#include <vector>
```

Include dependency graph for genericVcArbiter.h:



This graph shows which files directly or indirectly include this file:



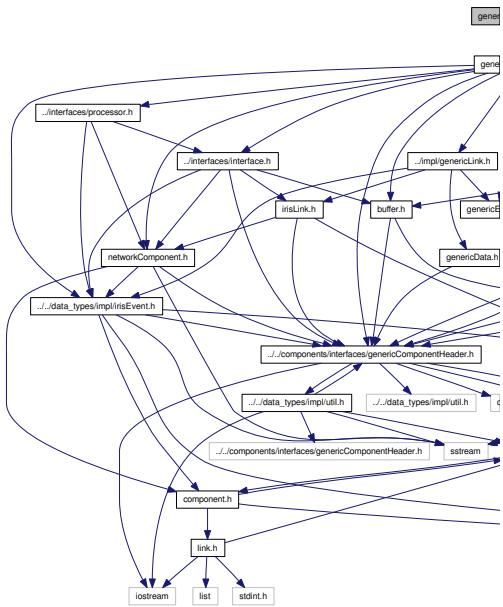
Classes

- class **GenericVcArbiter**

8.68 genericVcsInterface.cc File Reference

```
#include "genericVcsInterface.h"
```

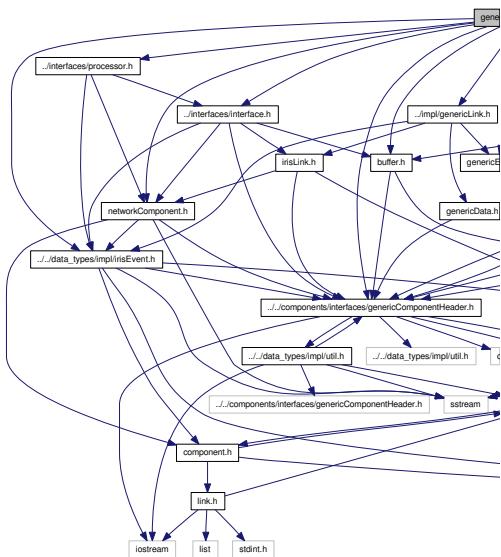
Include dependency graph for genericVcsInterface.cc:



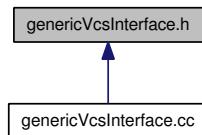
8.69 genericVcsInterface.h File Reference

```
#include "..../data_types/impl/highLevelPacket.h"
#include "..../data_types/impl/irisEvent.h"
#include "../interfaces/interface.h"
#include "../interfaces/networkComponent.h"
#include "../interfaces/processor.h"
#include "../interfaces/buffer.h"
#include "../interfaces/genericComponentHeader.h"
#include "../impl/genericLink.h"
#include "genericEvents.h"
#include "genericBuffer.h"
#include "genericData.h"
#include "genericArbiter.h"
#include <queue>
#include <vector>
#include <math.h>
```

Include dependency graph for genericVcsInterface.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **GenericVcsInterface**

Defines

- `#define DEFAULT_NO_OF_CREDITS 1`

Variables

- `uint do_two_stage_router`

8.69.1 Define Documentation

8.69.1.1 `#define DEFAULT_NO_OF_CREDITS 1`

Definition at line 37 of file genericVcsInterface.h.

8.69.2 Variable Documentation

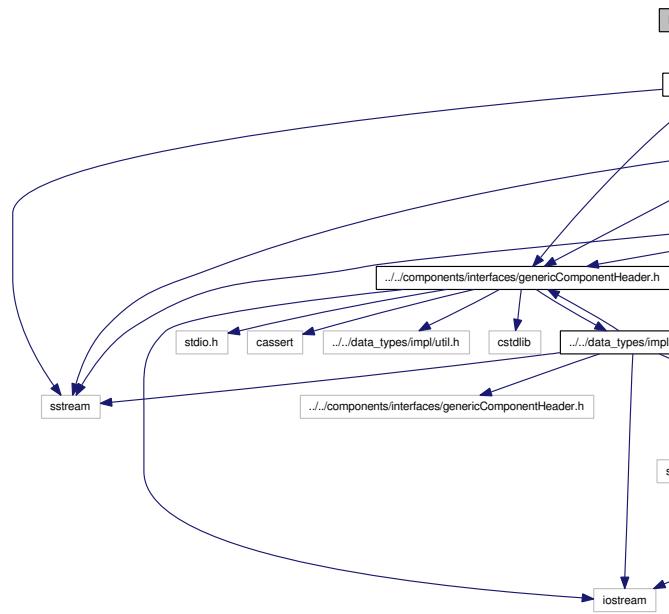
8.69.2.1 `uint do_two_stage_router`

Definition at line 38 of file simMc2Mesh.cc.

8.70 highLevelPacket.cc File Reference

```
#include "highLevelPacket.h"
```

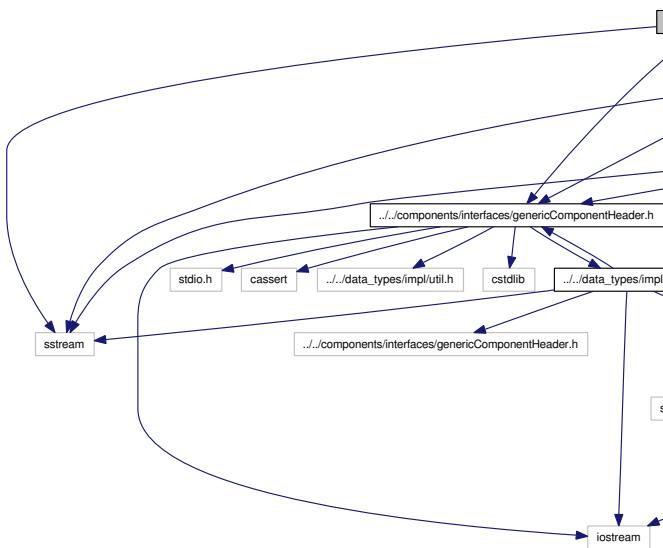
Include dependency graph for highLevelPacket.cc:



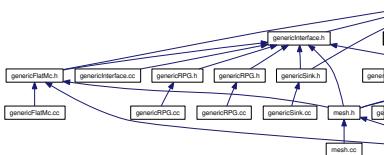
8.71 highLevelPacket.h File Reference

```
#include "../../components/interfaces/genericComponentHeader.h"
#include "lowLevelPacket.h"
#include "../../kernel/simulator.h"
#include <sstream>
#include <string>
#include <math.h>
```

Include dependency graph for highLevelPacket.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **HighLevelPacket**

Defines

- `#define CONTROL_VECTOR_SIZE 16`

Enumerations

- enum `virtual_network` { `VN0`, `VN1`, `VN2` }

8.71.1 Define Documentation

8.71.1.1 `#define CONTROL_VECTOR_SIZE 16`

Definition at line 28 of file highLevelPacket.h.

Referenced by `HighLevelPacket::to_low_level_packet()`.

8.71.2 Enumeration Type Documentation

8.71.2.1 `enum virtual_network`

Enumerator:

`VN0`

`VN1`

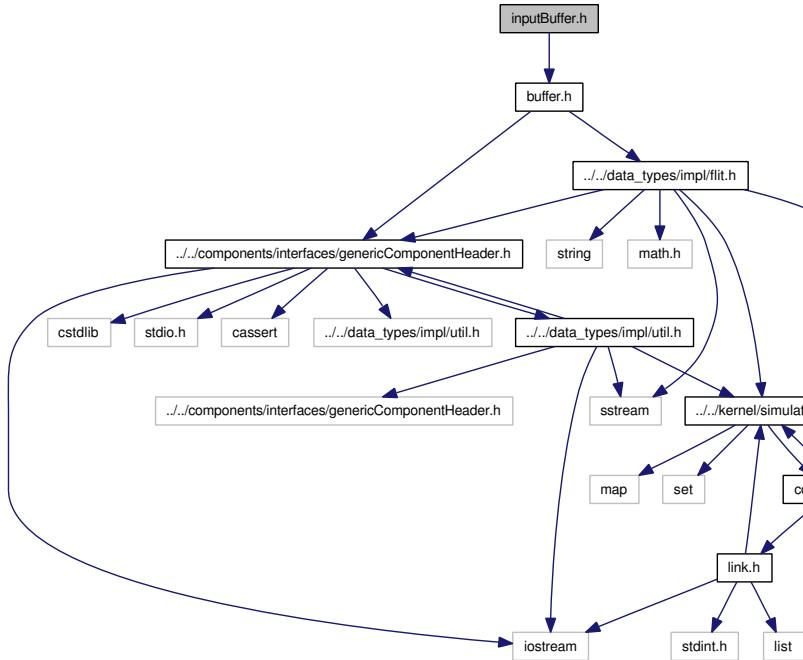
`VN2`

Definition at line 31 of file highLevelPacket.h.

8.72 inputBuffer.h File Reference

```
#include "buffer.h"
```

Include dependency graph for inputBuffer.h:



Classes

- class **InputBuffer**

Typedefs

- typedef unsigned int **uint**

8.72.1 Typedef Documentation

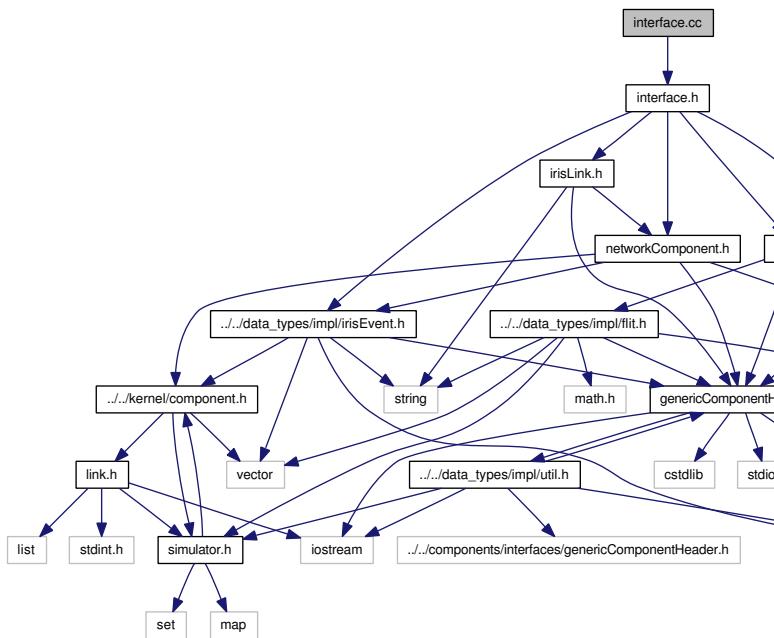
8.72.1.1 typedef unsigned int uint

Definition at line 24 of file `inputBuffer.h`.

8.73 interface.cc File Reference

```
#include "interface.h"
```

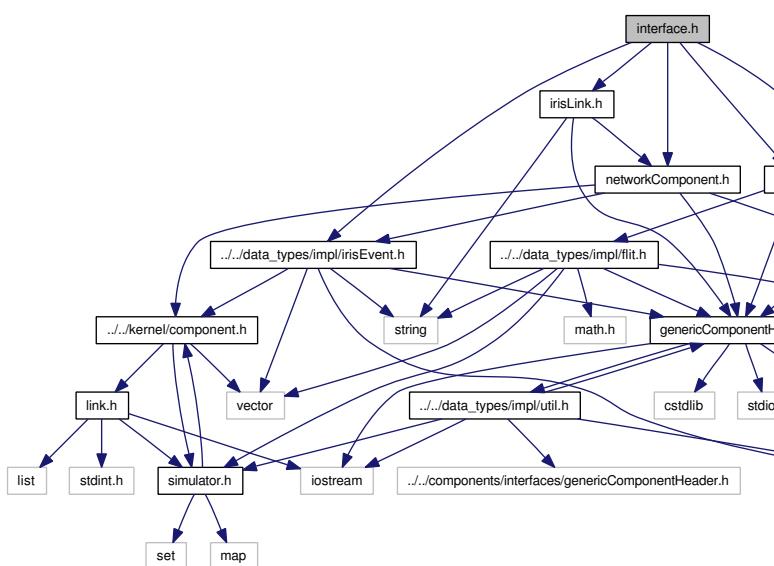
Include dependency graph for interface.cc:



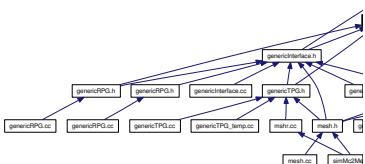
8.74 interface.h File Reference

```
#include "networkComponent.h"
#include "genericComponentHeader.h"
#include "buffer.h"
#include "irisLink.h"
#include "../../data_types/impl/irisEvent.h"
```

Include dependency graph for interface.h:



This graph shows which files directly or indirectly include this file:



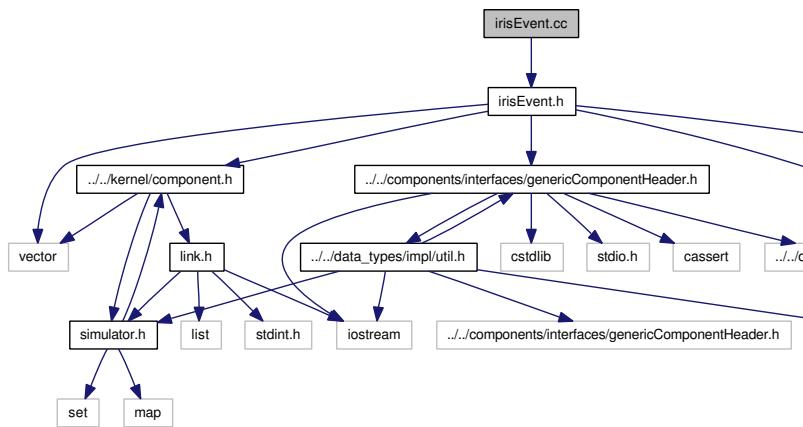
Classes

- class **Interface**

8.75 irisEvent.cc File Reference

```
#include "irisEvent.h"
```

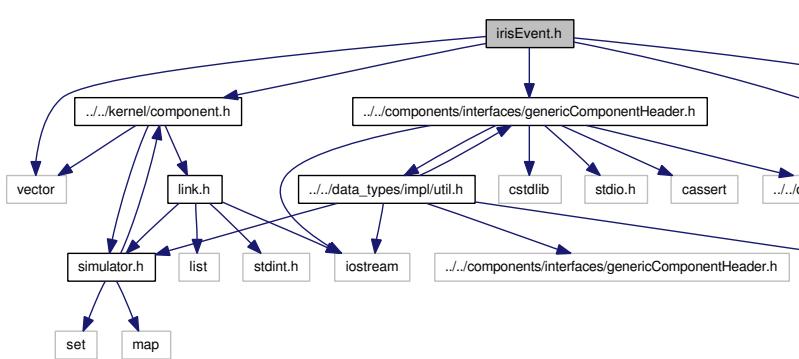
Include dependency graph for irisEvent.cc:



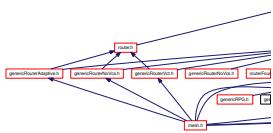
8.76 irisEvent.h File Reference

```
#include <vector>
#include <sstream>
#include <string>
#include "../../kernel/component.h"
#include "../../components/interfaces/genericComponentHeader.h"

Include dependency graph for irisEvent.h:
```



This graph shows which files directly or indirectly include this file:



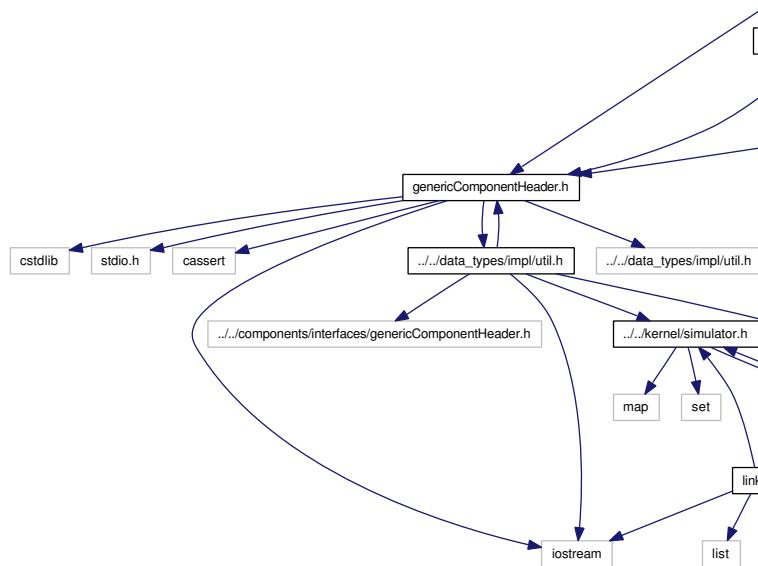
Classes

- class **IrisEvent**

8.77 irisLink.cc File Reference

```
#include "irisLink.h"
```

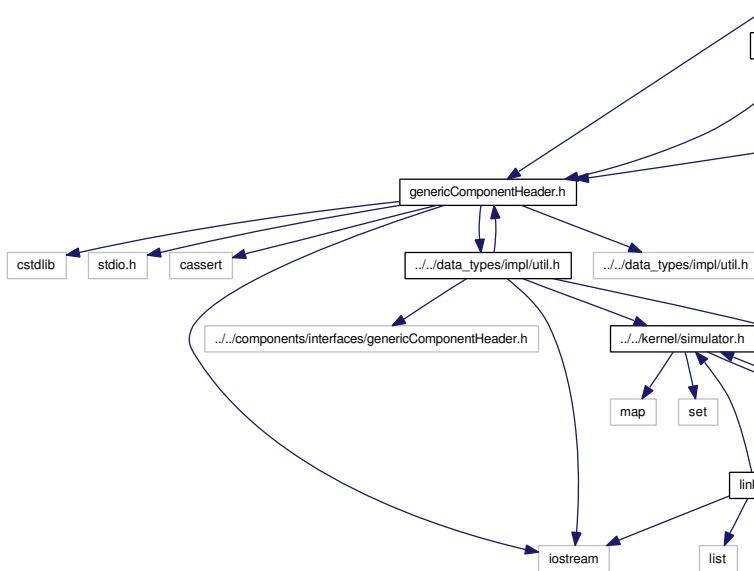
Include dependency graph for irisLink.cc:



8.78 irisLink.h File Reference

```
#include "genericComponentHeader.h"
#include "networkComponent.h"
#include <string>
```

Include dependency graph for irisLink.h:



This graph shows which files directly or indirectly include this file:



Classes

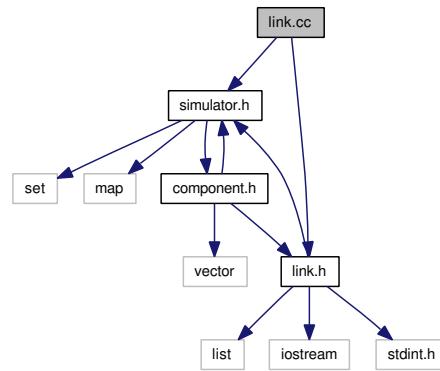
- class **IrisLink**

8.79 link.cc File Reference

```
#include "simulator.h"
```

```
#include "link.h"
```

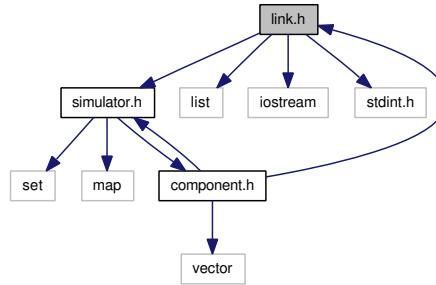
Include dependency graph for link.cc:



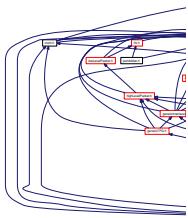
8.80 link.h File Reference

```
#include "simulator.h"
#include <list>
#include <iostream>
#include <stdint.h>
```

Include dependency graph for link.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **OutputBase**
- class **Output0< OBJ >**
- class **Link**

Functions

- template<typename OBJ >
void **addOutput** (**Link** **l*, int *outComponent*, double *latency*, void(OBJ::*)(uint64_t, int) *f*, OBJ **obj0*)

8.80.1 Function Documentation

8.80.1.1 template<typename OBJ > void addOutput (Link * *l*, int *outComponent*, double *latency*, void(OBJ::*)(uint64_t, int) *f*, OBJ **obj0*) [inline]

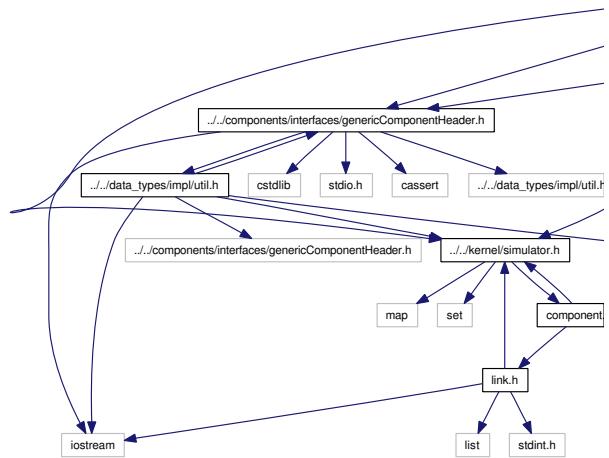
Definition at line 49 of file link.h.

References Link::outputs.

8.81 lowLevelPacket.cc File Reference

```
#include "lowLevelPacket.h"
```

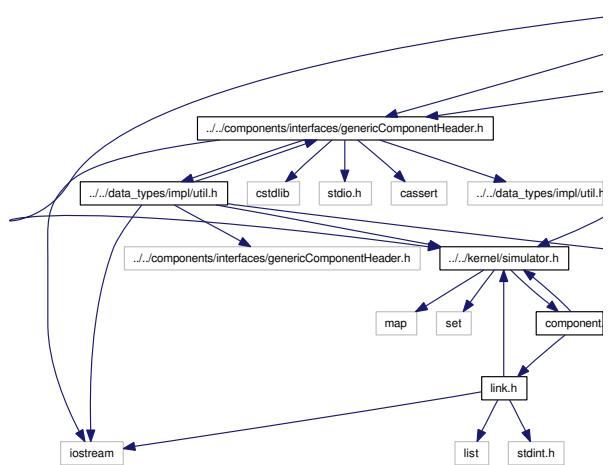
Include dependency graph for lowLevelPacket.cc:



8.82 lowLevelPacket.h File Reference

```
#include "../../components/interfaces/genericComponentHeader.h"
#include "../../kernel/simulator.h"
#include "../../data_types/impl/flit.h"
#include <vector>
#include <string>
#include <sstream>
#include <deque>
```

Include dependency graph for lowLevelPacket.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **LowLevelPacket**

8.83 main.cc File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <fstream>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/request_handler.h"
#include "../MemCtrl/bus_handler.h"
#include "../MemCtrl/bus.h"
#include "../MemCtrl/dram.h"
#include "../MemCtrl/NI.h"
#include "../MemCtrl/refresh_manager.h"
#include "../MemCtrl/response_handler.h"
#include "../MemCtrl/MC.h"
#include "../MemCtrl/mshr.h"
```

Include dependency graph for main.cc:



Functions

- bool **GetNextRequest** (**MSHR_H** mshrHandler[], **Request** *req, unsigned int *index)
- int **main** (int argc, char **argv)

8.83.1 Function Documentation

8.83.1.1 bool GetNextRequest (**MSHR_H** *mshrHandler*[], **Request** * *req*, unsigned int * *index*)

Definition at line 43 of file main.cc.

References **MSHR_H**::lastScheduledIndex, **MSHR_H**::mshr, **NO_OF_THREADS**, and **MSHR_H**::writeQueue.

Referenced by **main()**.

Here is the caller graph for this function:

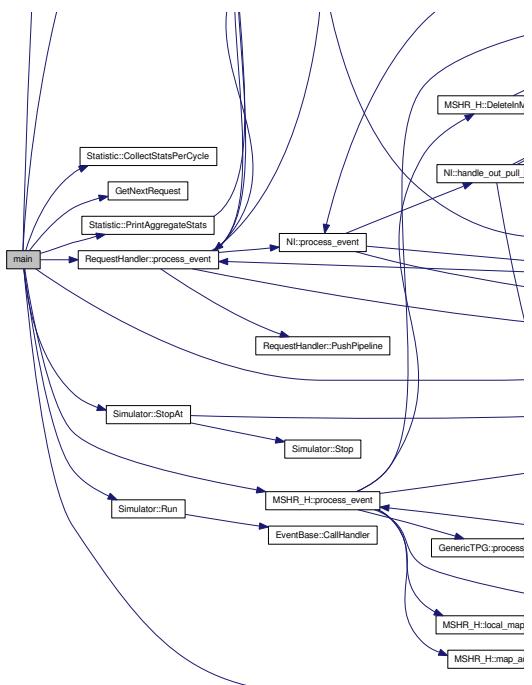


8.83.1.2 int main (int *argc*, char ** *argv*)

Definition at line 100 of file main.cc.

References Request::address, Request::arrivalTime, Statistic::CalculateAggregateStats(), Request::cmdType, Statistic::CollectStatsPerCycle(), MSHR_H::done, Statistic::doneOnce, IrisEvent::dst, IrisEvent::event_data, MSHR_H::filename, GetNextRequest(), MSHR_H::GlobalAddrMap(), MSHR_H::globalUnSink, Simulator::halted, MSHR_H::id, MC::ni, NO_OF_THREADS, Simulator::Now(), RequestHandler::oneBufferFull, Statistic::PrintAggregateStats(), RequestHandler::process_event(), MSHR_H::process_event(), MC::reqH, Simulator::Run(), Simulator::Schedule(), IrisEvent::src, START, MC::stats, Simulator::StopAt(), Request::threadId, MSHR_H::trace_filename, IrisEvent::type, and MSHR_H::unsink.

Here is the call graph for this function:



8.84 main.dox File Reference

8.85 MC.cc File Reference

```
#include "MC.h"
```

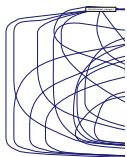
Include dependency graph for MC.cc:



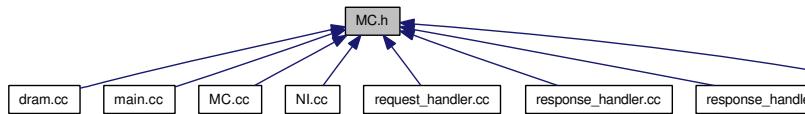
8.86 MC.h File Reference

```
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/request_handler.h"
#include "../MemCtrl/bus_handler.h"
#include "../MemCtrl/bus.h"
#include "../MemCtrl/dram.h"
#include "../MemCtrl/NI.h"
#include "../MemCtrl/refresh_manager.h"
#include "../MemCtrl/response_handler.h"
#include "../MemCtrl/stats.h"
```

Include dependency graph for MC.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **MC**

8.87 mcFrontEnd.cc File Reference

```
#include "mcFrontEnd.h"
```

Include dependency graph for mcFrontEnd.cc:



```
graph LR; A["data_types/impl/highLevelPacket.h"] --> B["data_types/impl/Device.h"]; B --> C["Interface/interface.h"]; C --> D["Interface.h"]
```

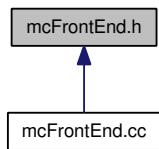
8.88 mcFrontEnd.h File Reference

```
#include "../../data_types/impl/highLevelPacket.h"
#include "../../data_types/impl/irisEvent.h"
#include "../interfaces/interface.h"
#include "../interfaces/networkComponent.h"
#include "../interfaces/processor.h"
#include "../interfaces/buffer.h"
#include "../../memctrl/constants.h"
#include "../../memctrl/request.h"
#include "genericEvents.h"
#include "genericBuffer.h"
#include "genericArbiter.h"
#include "genericData.h"
#include <queue>
#include <vector>
#include <math.h>
```

Include dependency graph for mcFrontEnd.h:



This graph shows which files directly or indirectly include this file:



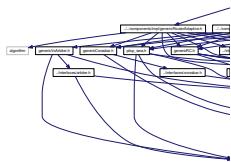
Classes

- class **MCFrontEnd**

8.89 mesh.cc File Reference

```
#include "mesh.h"
```

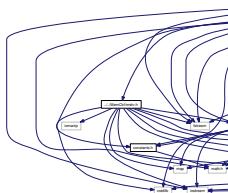
Include dependency graph for mesh.cc:



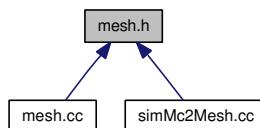
8.90 mesh.h File Reference

```
#include <iostream>
#include <fstream>
#include "../../components/impl/genericTPG.h"
#include "../../components/impl/genericRouterNoVcs.h"
#include "../../components/impl/genericRouterVct.h"
#include "../../components/impl/genericRouterAdaptive.h"
#include "../../components/impl/genericInterface.h"
#include "../../components/impl/genericFlatMc.h"
#include "../../components/impl/genericLink.h"
#include "../../MemCtrl/NI.h"
```

Include dependency graph for mesh.h:



This graph shows which files directly or indirectly include this file:



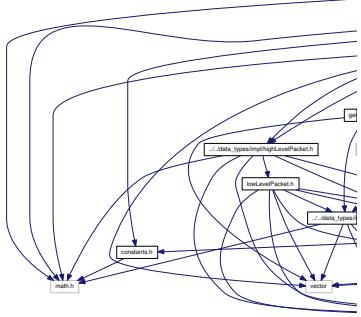
Classes

- class Mesh

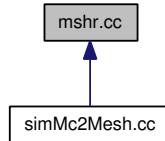
8.91 mshr.cc File Reference

```
#include <math.h>
#include "mshr.h"
#include "../components/impl/genericEvents.h"
#include "../components/impl/genericTPG.h"
```

Include dependency graph for mshr.cc:



This graph shows which files directly or indirectly include this file:



Variables

- unsigned int **THREAD_BITS_POSITION**

8.91.1 Variable Documentation

8.91.1.1 unsigned int **THREAD_BITS_POSITION**

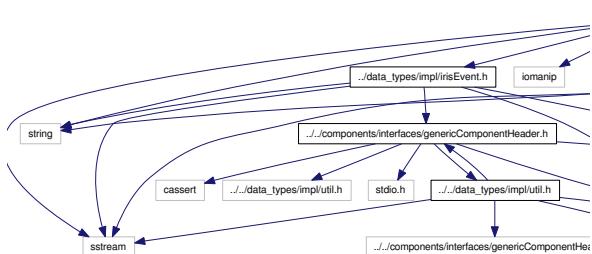
Definition at line 37 of file simMc2Mesh.cc.

Referenced by MSHR_H::GlobalAddrMap(), and main().

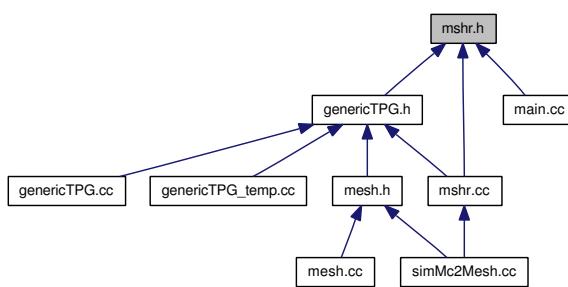
8.92 mshr.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <iomanip>
#include <cstdlib>
#include <fstream>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"

Include dependency graph for mshr.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **MSHR_H**

TypeDefs

- typedef bool **BoolVector** [8]

Variables

- `unsigned long long int max_sim_time`
- `uint no_mcs`
- `unsigned int MC_ADDR_BITS`

8.92.1 Typedef Documentation

8.92.1.1 `typedef bool BoolVector[8]`

Definition at line 48 of file mshr.h.

8.92.2 Variable Documentation

8.92.2.1 `unsigned long long int max_sim_time`

Definition at line 34 of file simMc2Mesh.cc.

8.92.2.2 `unsigned int MC_ADDR_BITS`

Definition at line 35 of file simMc2Mesh.cc.

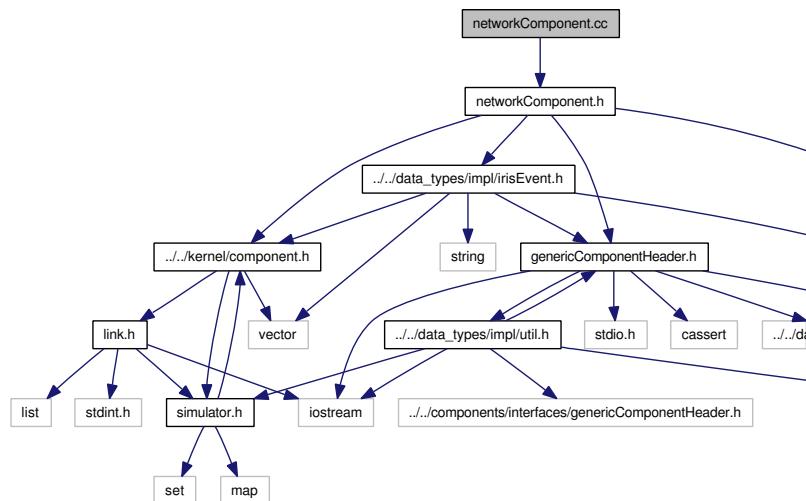
8.92.2.3 `uint no_mcs`

Definition at line 33 of file simMc2Mesh.cc.

8.93 networkComponent.cc File Reference

```
#include "networkComponent.h"
```

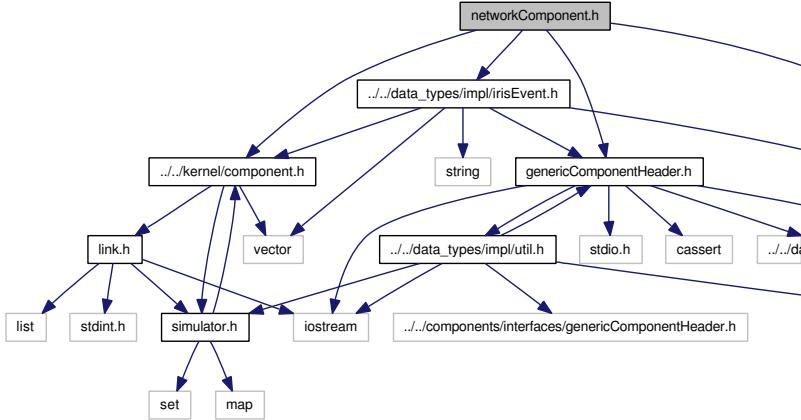
Include dependency graph for networkComponent.cc:



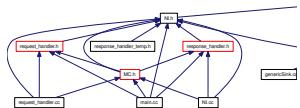
8.94 networkComponent.h File Reference

```
#include "../../kernel/component.h"
#include "genericComponentHeader.h"
#include "../../data_types/impl/irisEvent.h"
#include <sstream>
```

Include dependency graph for networkComponent.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **NetworkComponent**

8.95 NI.cc File Reference

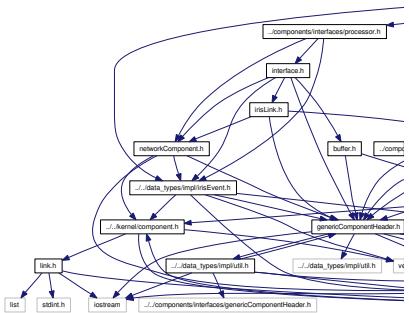
```
#include "NI.h"
#include "MC.h"
#include "response_handler.h"
Include dependency graph for NI.cc:
```



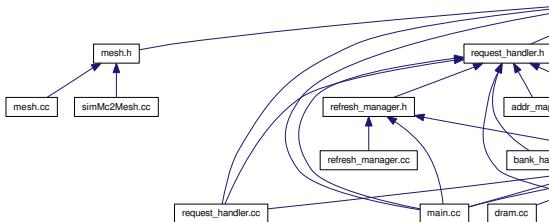
8.96 NI.h File Reference

```
#include "../components/interfaces/processor.h"
#include "../components/impl/genericEvents.h"
#include "../data_types/impl/highLevelPacket.h"
#include "../data_types/impl/irisEvent.h"
#include "../components/impl/genericData.h"
#include "../MemCtrl/constants.h"
#include "../MemCtrl/request.h"
#include <fstream>
```

Include dependency graph for NI.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **NI**

Defines

- #define **DEFAULT_RAN_MAX_TIME** 100

8.96.1 Define Documentation

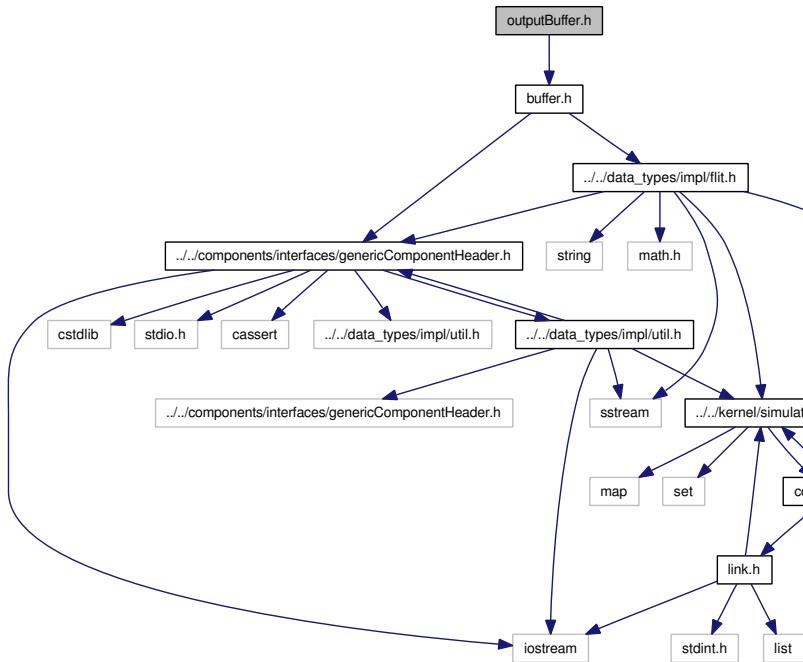
8.96.1.1 #define **DEFAULT_RAN_MAX_TIME** 100

Definition at line 32 of file NI.h.

8.97 outputBuffer.h File Reference

```
#include "buffer.h"
```

Include dependency graph for outputBuffer.h:



Classes

- class `OutputBuffer`

Typedefs

- typedef unsigned int `uint`

8.97.1 Typedef Documentation

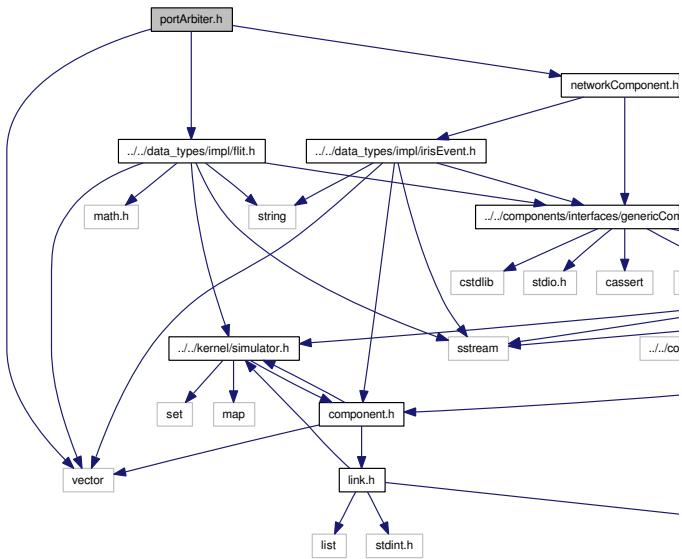
8.97.1.1 typedef unsigned int uint

Definition at line 24 of file `outputBuffer.h`.

8.98 portArbiter.h File Reference

```
#include <vector>
#include "../../data_types/impl/flit.h"
#include "networkComponent.h"
```

Include dependency graph for portArbiter.h:



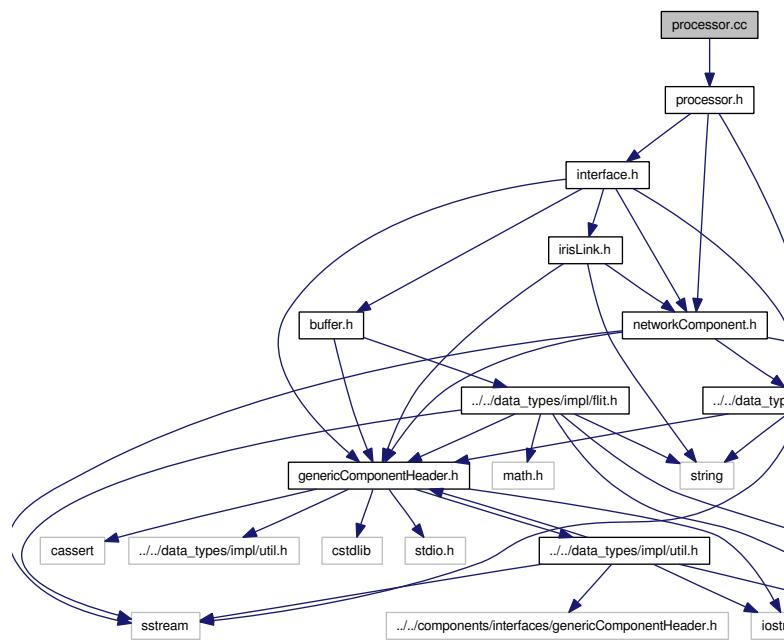
Classes

- class **PortArbiter**

8.99 processor.cc File Reference

```
#include "processor.h"
```

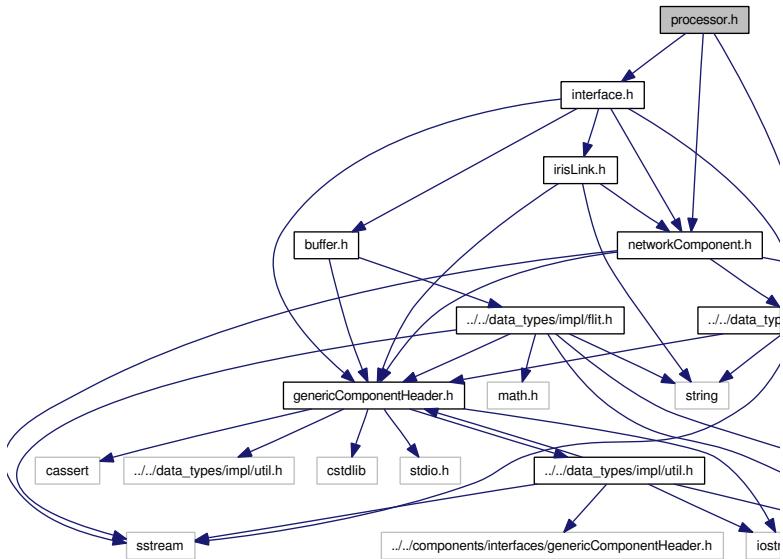
Include dependency graph for processor.cc:



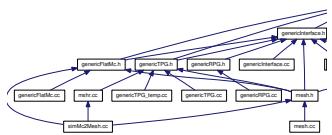
8.100 processor.h File Reference

```
#include "networkComponent.h"
#include "../../data_types/impl/irisEvent.h"
#include "interface.h"
```

Include dependency graph for processor.h:



This graph shows which files directly or indirectly include this file:



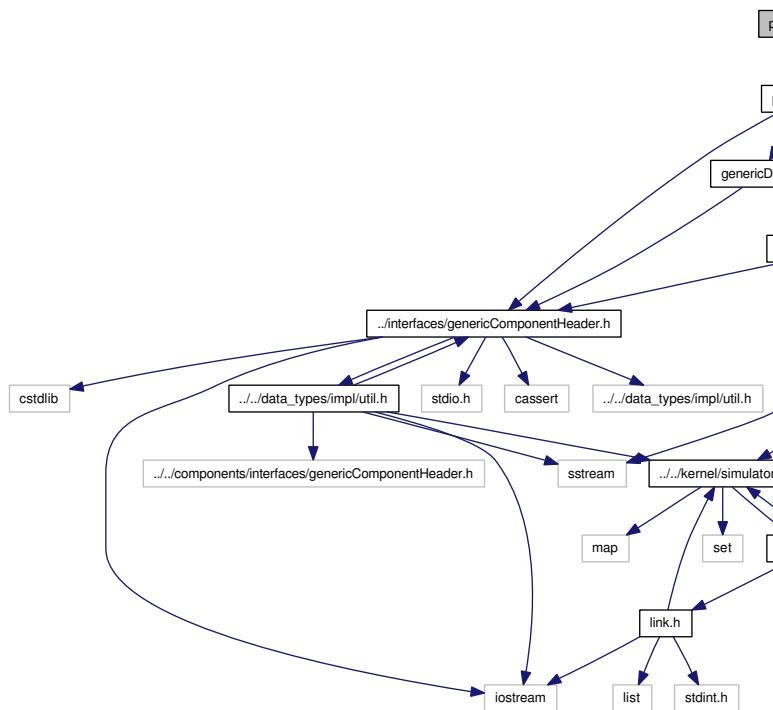
Classes

- class **Processor**

8.101 ptop_swa.cc File Reference

```
#include "ptop_swa.h"
```

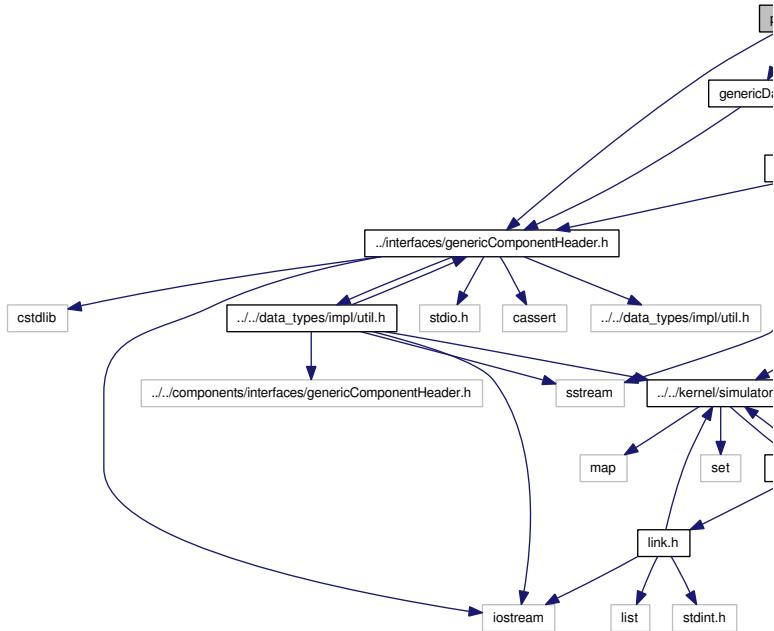
Include dependency graph for ptop_swa.cc:



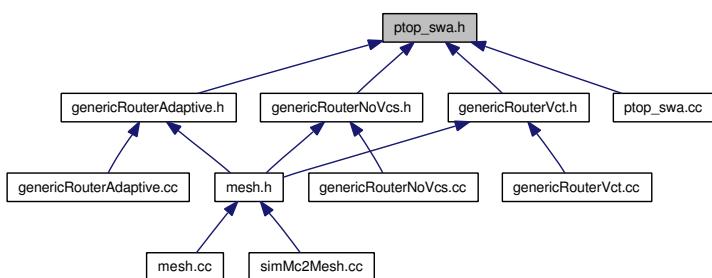
8.102 ptop_swa.h File Reference

```
#include "../interfaces/genericComponentHeader.h"
#include "../../data_types/impl/flit.h"
#include "genericData.h"
#include <vector>
```

Include dependency graph for ptop_swa.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **PToPSwitchArbiter**

Variables

- **SW_ARBITRATION sw_arbitration**

8.102.1 Variable Documentation

8.102.1.1 SW_ARBITRATION sw_arbitration

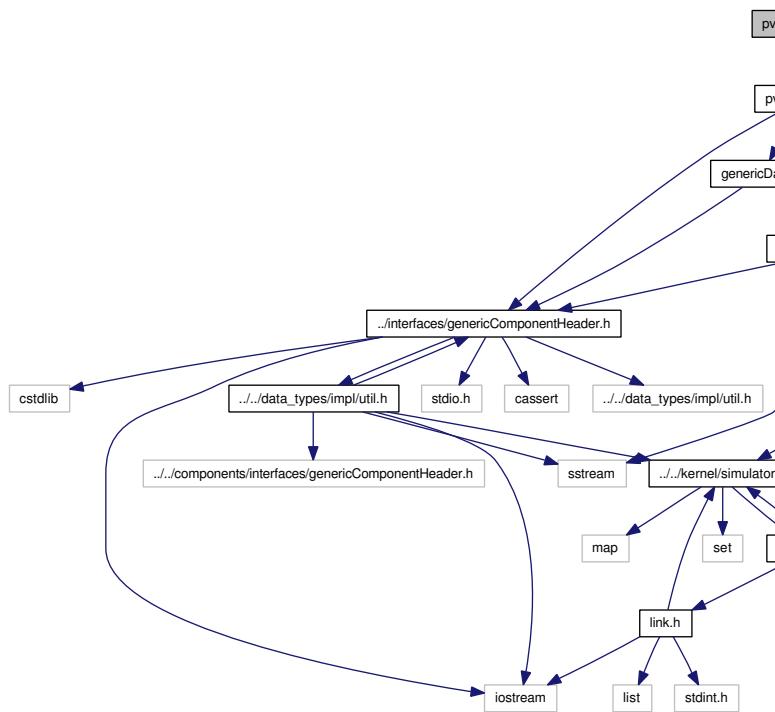
Definition at line 41 of file simMc2Mesh.cc.

Referenced by main(), and PToPSwitchArbiter::pick_winner().

8.103 pvttopv_swa.cc File Reference

```
#include "pvttopv_swa.h"
```

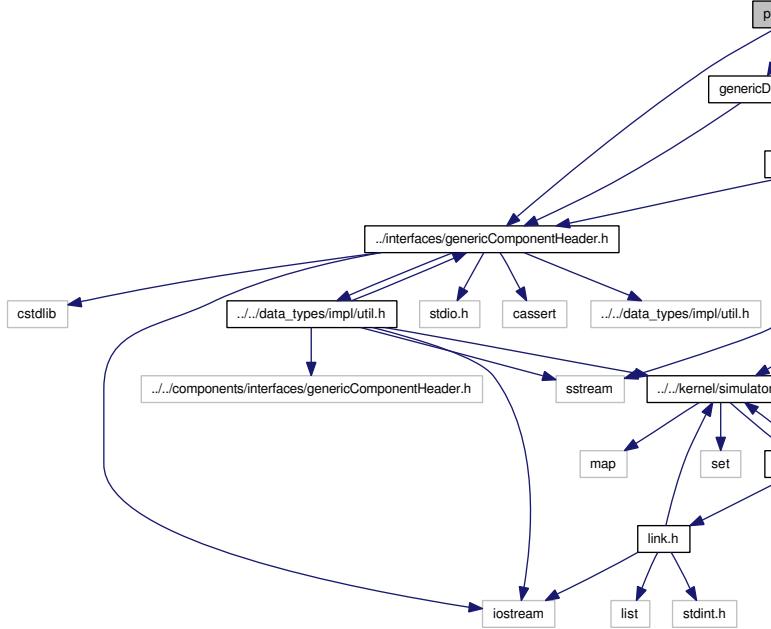
Include dependency graph for pvttopv_swa.cc:



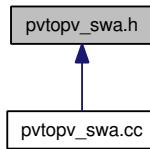
8.104 pvttopv_swa.h File Reference

```
#include "../interfaces/genericComponentHeader.h"
#include "../../data_types/impl/flit.h"
#include "genericData.h"
#include <vector>
```

Include dependency graph for pvttopv_swa.h:



This graph shows which files directly or indirectly include this file:



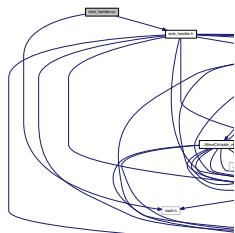
Classes

- class **PVToPV_swa**

8.105 rank_handler.cc File Reference

```
#include <math.h>
#include "rank_handler.h"
```

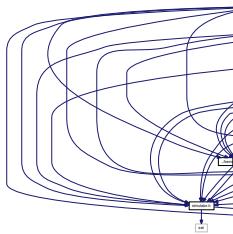
Include dependency graph for rank_handler.cc:



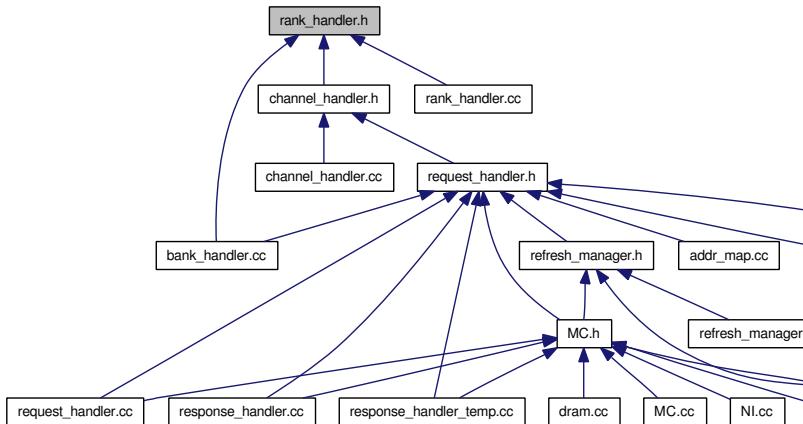
8.106 rank_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/bank_handler.h"
#include "../MemCtrl/constants.h"

Include dependency graph for rank_handler.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class **RankHandler**

Typedefs

- typedef vector< Request > ReqBuffer

8.106.1 Typedef Documentation

8.106.1.1 `typedef vector<Request> ReqBuffer`

Definition at line 44 of file rank_handler.h.

8.107 refresh_manager.cc File Reference

```
#include <math.h>
```

```
#include "refresh_manager.h"
```

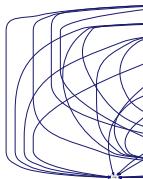
Include dependency graph for refresh_manager.cc:



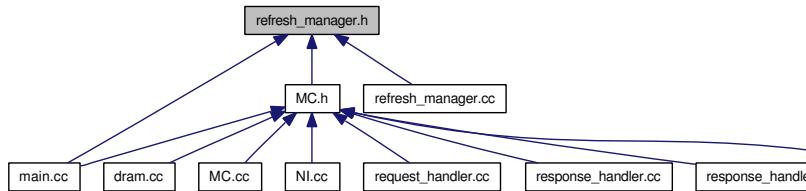
8.108 refresh_manager.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/request_handler.h"

Include dependency graph for refresh_manager.h:
```



This graph shows which files directly or indirectly include this file:



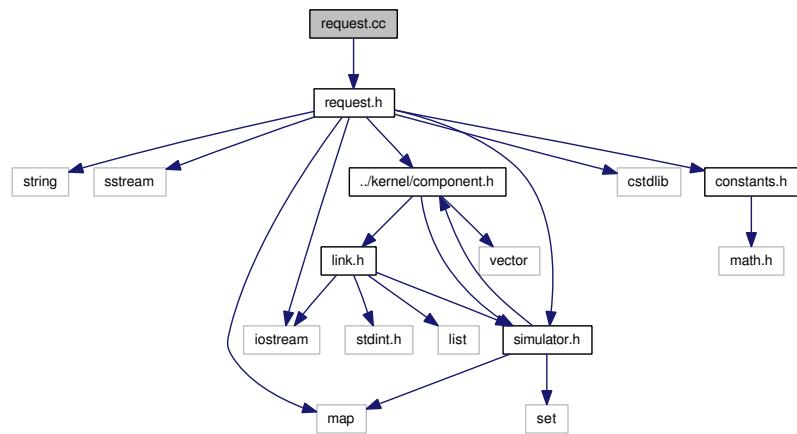
Classes

- class **RefreshMgr**

8.109 request.cc File Reference

```
#include "request.h"
```

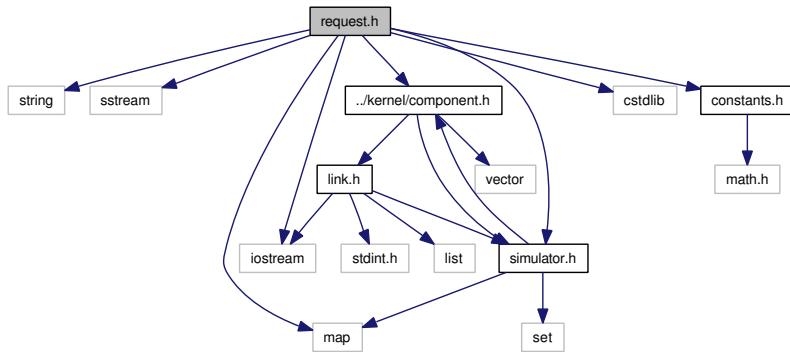
Include dependency graph for request.cc:



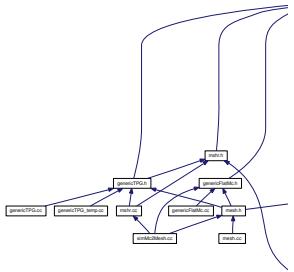
8.110 request.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "constants.h"
```

Include dependency graph for request.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct **Data**
- class **Request**

TypeDefs

- typedef **Data Data**

Enumerations

```
- enum CStatus { OPEN, CLOSED, CONFLICT, IDLE }
- enum Command_t {
    CACHE_NOP,      CACHE_READ,      CACHE_WRITE,      CACHE_-
    WRITEBACK,
    CACHE_PREFETCH, REFRESH }
```

8.110.1 Typedef Documentation

8.110.1.1 `typedef Data Data`

Definition at line 50 of file request.h.

8.110.2 Enumeration Type Documentation

8.110.2.1 `enum Command_t`

Enumerator:

```
CACHE_NOP
CACHE_READ
CACHE_WRITE
CACHE_WRITEBACK
CACHE_PREFETCH
REFRESH
```

Definition at line 43 of file request.h.

8.110.2.2 `enum CStatus`

Enumerator:

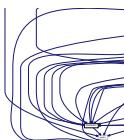
```
OPEN
CLOSED
CONFLICT
IDLE
```

Definition at line 42 of file request.h.

8.111 request_handler.cc File Reference

```
#include <math.h>
#include "request_handler.h"
#include "bus.h"
#include "NI.h"
#include "MC.h"
```

Include dependency graph for request_handler.cc:



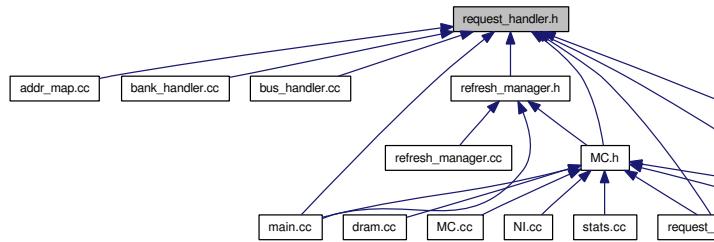
8.112 request_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/channel_handler.h"
#include "../MemCtrl/bus_handler.h"
#include "../MemCtrl/bank_handler.h"
#include "../MemCtrl/addr_map.h"
#include "../MemCtrl/constants.h"
#include "../MemCtrl/NI.h"
```

Include dependency graph for request_handler.h:



This graph shows which files directly or indirectly include this file:



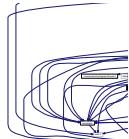
Classes

- class **RequestHandler**

8.113 response_handler.cc File Reference

```
#include <math.h>
#include "response_handler.h"
#include "request_handler.h"
#include "MC.h"
```

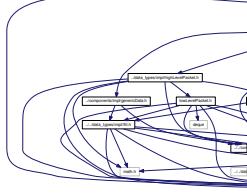
Include dependency graph for response_handler.cc:



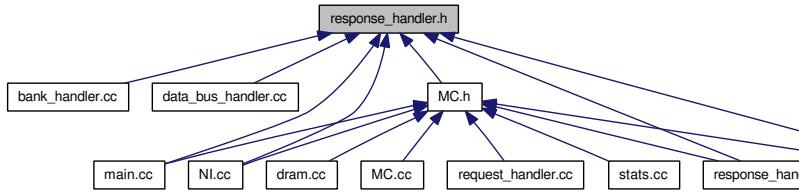
8.114 response_handler.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/constants.h"
#include "../MemCtrl/NI.h"
#include "../MemCtrl/cmd_issuer.h"
```

Include dependency graph for response_handler.h:



This graph shows which files directly or indirectly include this file:



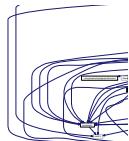
Classes

- class **ResponseHandler**

8.115 response_handler_temp.cc File Reference

```
#include <math.h>
#include "response_handler.h"
#include "request_handler.h"
#include "MC.h"
```

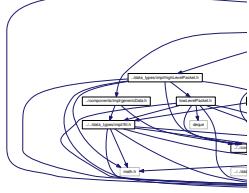
Include dependency graph for response_handler_temp.cc:



8.116 response_handler_temp.h File Reference

```
#include <string>
#include <sstream>
#include <iostream>
#include <cstdlib>
#include <map>
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "../MemCtrl/request.h"
#include "../MemCtrl/constants.h"
#include "../MemCtrl/NI.h"
#include "../MemCtrl/cmd_issuer.h"
```

Include dependency graph for response_handler_temp.h:



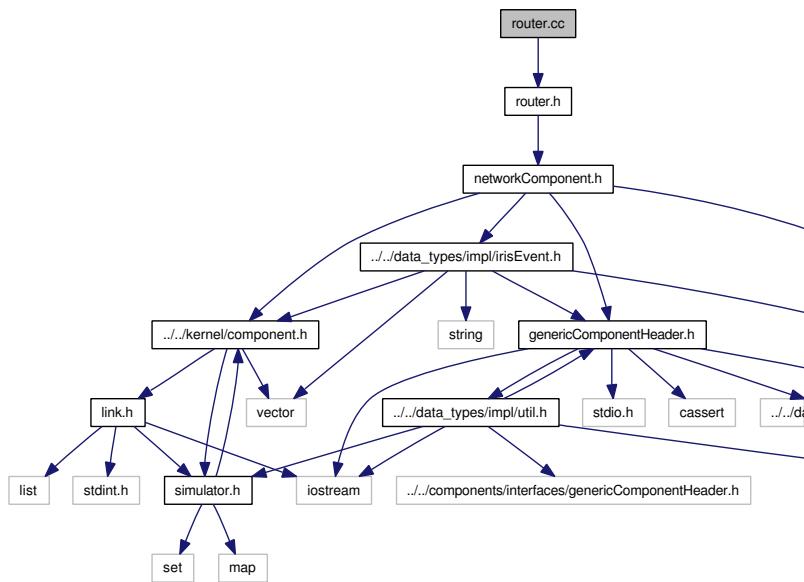
Classes

- class **ResponseHandler**

8.117 router.cc File Reference

```
#include "router.h"
```

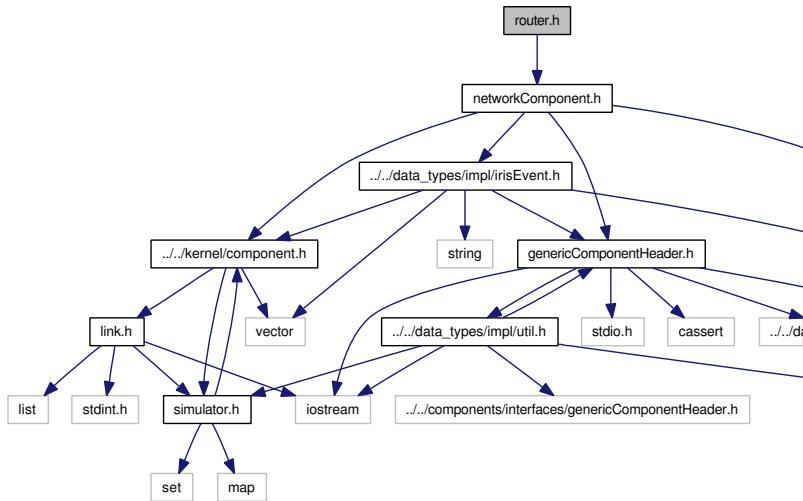
Include dependency graph for router.cc:



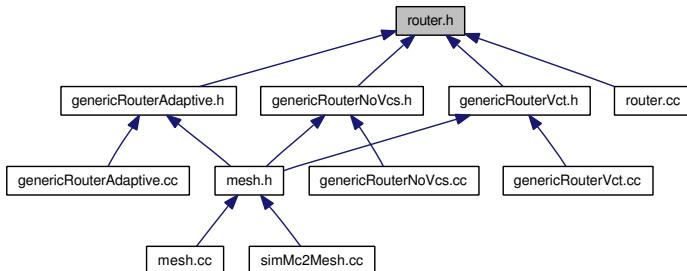
8.118 router.h File Reference

```
#include "networkComponent.h"
```

Include dependency graph for router.h:



This graph shows which files directly or indirectly include this file:



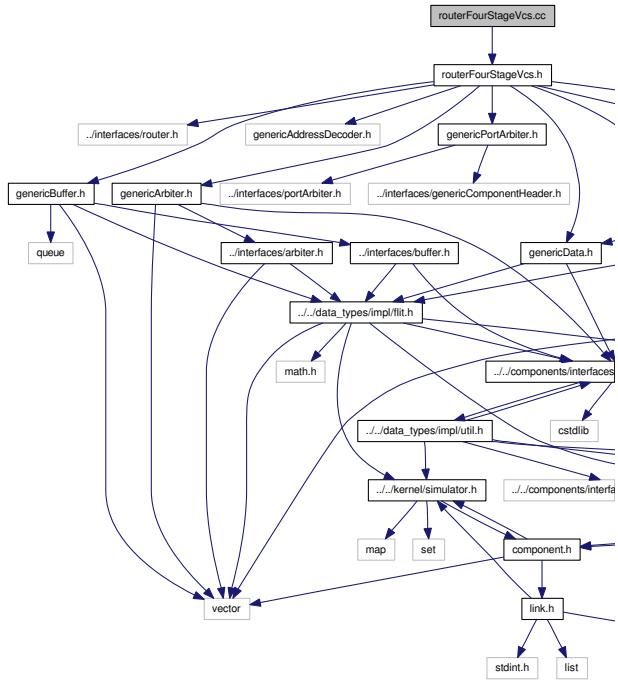
Classes

- class **Router**

8.119 routerFourStageVcs.cc File Reference

```
#include "routerFourStageVcs.h"
```

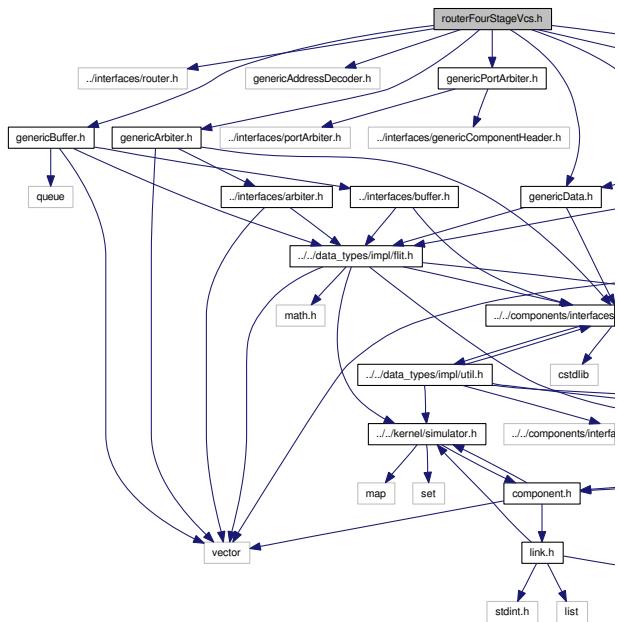
Include dependency graph for routerFourStageVcs.cc:



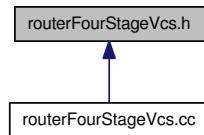
8.120 routerFourStageVcs.h File Reference

```
#include "../interfaces/router.h"
#include "genericBuffer.h"
#include "genericAddressDecoder.h"
#include "genericArbiter.h"
#include "genericPortArbiter.h"
#include "genericCrossbar.h"
#include "genericEvents.h"
#include "genericData.h"
#include "genericLink.h"
```

Include dependency graph for routerFourStageVcs.h:



This graph shows which files directly or indirectly include this file:



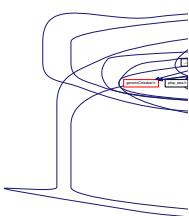
Classes

- class **RouterFourStageVcs**

8.121 simMc2Mesh.cc File Reference

```
#include "mesh.h"
#include "../../data_types/impl/flit.h"
#include "../../data_types/impl/highLevelPacket.h"
#include "../../components/impl/genericFlatMc.h"
#include "../../components/impl/genericData.h"
#include "../../../MemCtrl/mshr.cc"
#include <string.h>
#include <sys/time.h>
#include <algorithm>
```

Include dependency graph for simMc2Mesh.cc:



Functions

- int **main** (int argc, char *argv[])

Variables

- unsigned int **no_nodes** = 0
- unsigned int **no_mcs** = 0
- unsigned long long int **max_sim_time** = 10000
- unsigned int **MC_ADDR_BITS** = 10
- unsigned int **BANK_BITS** = 13
- unsigned int **THREAD_BITS_POSITION** = 25
- unsigned int **do_two_stage_router** = 0
- unsigned int **max_phy_link_bits** = 128
- **ROUTING_SCHEME** **rc_method** = XY
- **SW_ARBITRATION** **sw_arbitration** = ROUND_ROBIN
- uint **print_setup** = 0
- uint **grid_size** = 0

8.121.1 Function Documentation

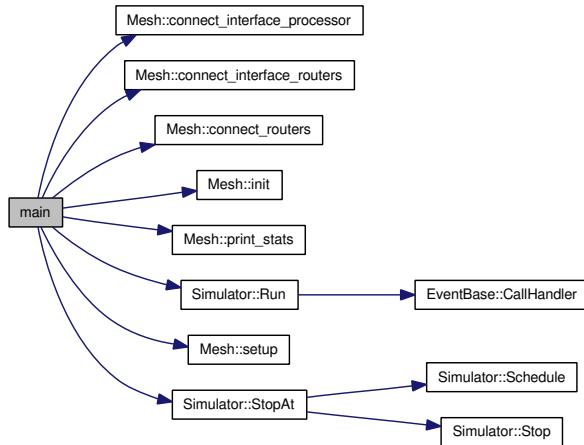
8.121.1.1 int main (int *argc*, char * *argv*[])

Definition at line 46 of file simMc2Mesh.cc.

References **BANK_BITS**, **Mesh::connect_interface_processor()**, **Mesh::connect_interface_routers()**, **Mesh::connect_routers()**, **do_two_stage_router**, **FCFS**, **grid_size**, **Mesh::init()**, **Mesh::interfaces**, **Mesh::link_a**, **Mesh::link_b**, **max_phy_link_bits**, **Mesh::max_sim_time**, **max_sim_time**, **MC_ADDR_BITS**, **NEGATIVE_FIRST**, **no_mcs**, **no_nodes**, **NORTH_LAST**, **NORTH_LAST_NON_MINIMAL**, **print_setup**,

Mesh::print_stats(), Mesh::processors, rc_method, ROUND_ROBIN, ROUND_ROBIN_PRIORITY, Mesh::routers, Simulator::Run(), Mesh::setup(), Simulator::StopAt(), sw_arbitration, THREAD_BITS_POSITION, VALIANT, WEST_FIRST, and XY.

Here is the call graph for this function:



8.121.2 Variable Documentation

8.121.2.1 unsigned int BANK_BITS = 13

Definition at line 36 of file simMc2Mesh.cc.

Referenced by main(), and AddrMap::map_addr().

8.121.2.2 unsigned int do_two_stage_router = 0

Definition at line 38 of file simMc2Mesh.cc.

Referenced by GenericRouterVct::do_switch_traversal(), GenericRouterAdaptive::do_switch_traversal(), GenericVcsInterface::handle_link_arrival(), GenericInterface::handle_link_arrival(), GenericVcsInterface::handle_tick_event(), GenericInterface::handle_tick_event(), main(), GenericRouterVct::send_credit_back(), and GenericRouterAdaptive::send_credit_back().

8.121.2.3 uint grid_size = 0

Definition at line 43 of file simMc2Mesh.cc.

Referenced by main(), and GenericRC::route_north_last_non_minimal().

8.121.2.4 unsigned int max_phy_link_bits = 128

Definition at line 39 of file simMc2Mesh.cc.

Referenced by GenericTPG::convertFromBitStream(), NI::convertToBitStream(), GenericTPG::convertToBitStream(), GenericRPG::handle_out_pull_event(), GenericFlatMc::handle_out_pull_event(), main(), Flit::populate_phit_data(), and HighLevelPacket::to_low_level_packet().

8.121.2.5 unsigned long long int max_sim_time = 10000

Definition at line 34 of file simMc2Mesh.cc.

Referenced by main(), GenericLink::print_stats(), and MSHR_H::process_event().

8.121.2.6 unsigned int MC_ADDR_BITS = 10

Definition at line 35 of file simMc2Mesh.cc.

Referenced by main(), and MSHR_H::map_addr().

8.121.2.7 unsigned int no_mcs = 0

Definition at line 33 of file simMc2Mesh.cc.

Referenced by MSHR_H::countBLP(), main(), and MSHR_H::map_addr().

8.121.2.8 unsigned int no_nodes = 0

Definition at line 33 of file simMc2Mesh.cc.

Referenced by main(), and GenericRC::route_north_last_non_minimal().

8.121.2.9 uint print_setup = 0

Definition at line 42 of file simMc2Mesh.cc.

Referenced by main().

8.121.2.10 ROUTING_SCHEME rc_method = XY

Definition at line 40 of file simMc2Mesh.cc.

Referenced by main(), and GenericRC::push().

8.121.2.11 SW_ARBITRATION sw_arbitration = ROUND_ROBIN

Definition at line 41 of file simMc2Mesh.cc.

Referenced by main(), and PToPSwitchArbiter::pick_winner().

8.121.2.12 unsigned int THREAD_BITS_POSITION = 25

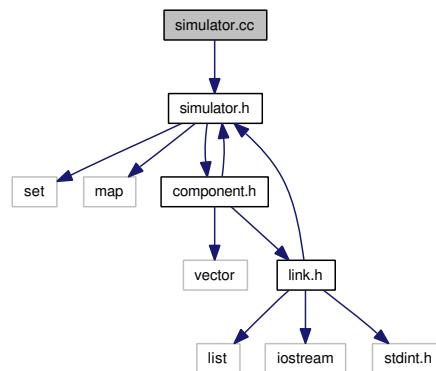
Definition at line 37 of file simMc2Mesh.cc.

Referenced by MSHR_H::GlobalAddrMap(), and main().

8.122 simulator.cc File Reference

```
#include "simulator.h"
```

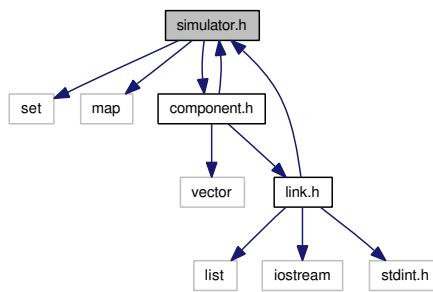
Include dependency graph for simulator.cc:



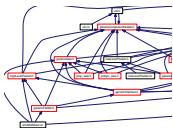
8.123 simulator.h File Reference

```
#include <set>
#include <map>
#include "component.h"

Include dependency graph for simulator.h:
```



This graph shows which files directly or indirectly include this file:



Classes

- class `EventBase`
- class `EventId`
- class `Event0< T, OBJ >`
- class `Event1< T, OBJ, U1, T1 >`
- class `Event2< T, OBJ, U1, T1, U2, T2 >`
- class `Event3< T, OBJ, U1, T1, U2, T2, U3, T3 >`
- class `Event4< T, OBJ, U1, T1, U2, T2, U3, T3, U4, T4 >`
- class `Event0Stat`
- class `Event1Stat< U1, T1 >`
- class `Event2Stat< U1, T1, U2, T2 >`
- class `Event3Stat< U1, T1, U2, T2, U3, T3 >`
- class `Event4Stat< U1, T1, U2, T2, U3, T3, U4, T4 >`
- class `event_less`
- class `ComponentDescription`
- class `Simulator`

Typedefs

- typedef `std::set< EventBase *, event_less > EventSet_t`
- typedef `std::map< int, ComponentDescription * > ComponentMap_t`

8.123.1 Typedef Documentation

8.123.1.1 `typedef std::map<int, ComponentDescription*> ComponentMap_t`

Definition at line 293 of file simulator.h.

8.123.1.2 `typedef std::set<EventBase*, event_less> EventSet_t`

Definition at line 281 of file simulator.h.

8.124 stats.cc File Reference

```
#include "MC.h"
#include "stats.h"
```

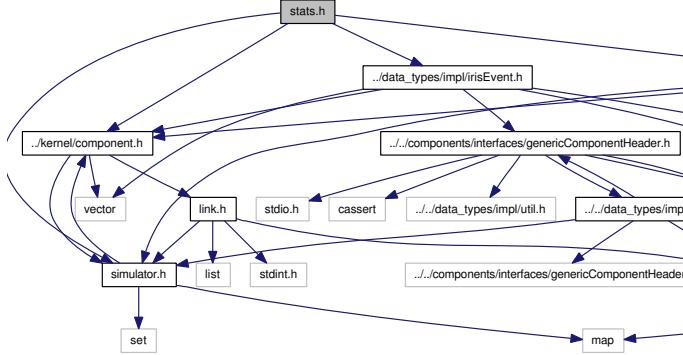
Include dependency graph for stats.cc:



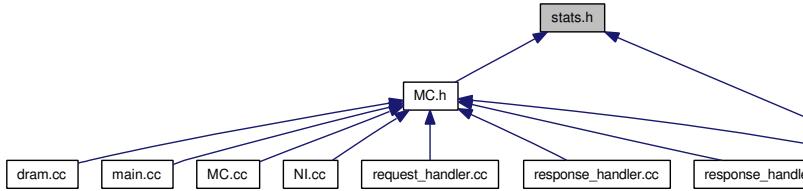
8.125 stats.h File Reference

```
#include "../kernel/component.h"
#include "../kernel/simulator.h"
#include "../data_types/impl/irisEvent.h"
#include "request.h"
```

Include dependency graph for stats.h:



This graph shows which files directly or indirectly include this file:



Classes

- class **Statistic**
- class **PowerStats**

Defines

- #define **MaxVcc** 1.58
- #define **MinVcc** 1.43
- #define **DQS** 2
- #define **DM** 1
- #define **IDD0** 120
- #define **IDD2PS** 10
- #define **IDD2PF** 25
- #define **IDD2N** 60
- #define **IDD3P** 40
- #define **IDD3N** 65
- #define **IDD4R** 250
- #define **IDD4W** 225
- #define **IDD5A** 260
- #define **t_CK** 1.25
- #define **SysVdd** 1.5
- #define **SysClk** 800

- `#define BL 8`
- `#define PdqRD 5.3`
- `#define PdqWR 13.2`
- `#define PdqRDoth 0`
- `#define PdqWRoth 0`

8.125.1 Define Documentation

8.125.1.1 `#define BL 8`

Definition at line 131 of file stats.h.

Referenced by PowerStats::CalcPower(), and Statistic::CalculateAggregateStats().

8.125.1.2 `#define DM 1`

Definition at line 118 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.3 `#define DQS 2`

Definition at line 117 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.4 `#define IDD0 120`

Definition at line 119 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.5 `#define IDD2N 60`

Definition at line 122 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.6 `#define IDD2PF 25`

Definition at line 121 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.7 `#define IDD2PS 10`

Definition at line 120 of file stats.h.

8.125.1.8 `#define IDD3N 65`

Definition at line 124 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.9 #define IDD3P 40

Definition at line 123 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.10 #define IDD4R 250

Definition at line 125 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.11 #define IDD4W 225

Definition at line 126 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.12 #define IDD5A 260

Definition at line 127 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.13 #define MaxVcc 1.58

Definition at line 115 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.14 #define MinVcc 1.43

Definition at line 116 of file stats.h.

8.125.1.15 #define PdqRD 5.3

Definition at line 132 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.16 #define PdqRDoth 0

Definition at line 134 of file stats.h.

8.125.1.17 #define PdqWR 13.2

Definition at line 133 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.18 #define PdqWRoth 0

Definition at line 135 of file stats.h.

8.125.1.19 #define SysClk 800

Definition at line 130 of file stats.h.

Referenced by PowerStats::CalcPower(), and Statistic::CalculateAggregateStats().

8.125.1.20 #define SysVdd 1.5

Definition at line 129 of file stats.h.

Referenced by PowerStats::CalcPower().

8.125.1.21 #define t_CK 1.25

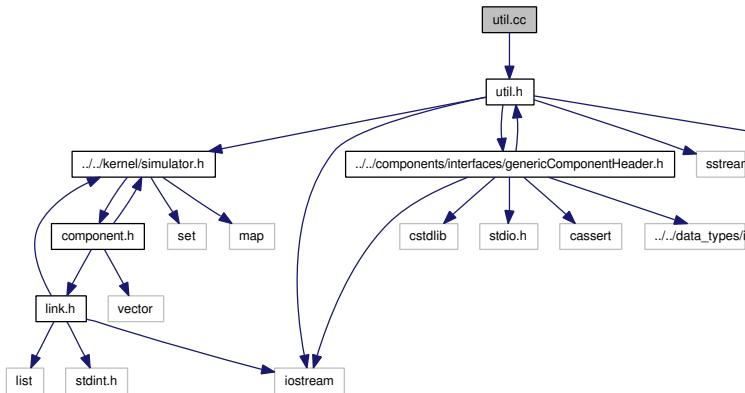
Definition at line 128 of file stats.h.

Referenced by PowerStats::CalcPower().

8.126 util.cc File Reference

```
#include "util.h"
```

Include dependency graph for util.cc:



Functions

- void **timed_cout** (string str)

8.126.1 Function Documentation

8.126.1.1 void **timed_cout** (string str)

Definition at line 23 of file util.cc.

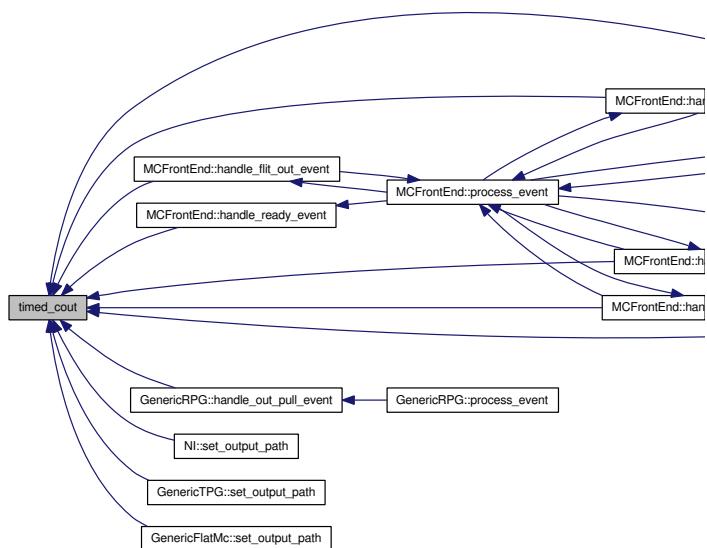
References Simulator::Now().

Referenced by MCFrontEnd::handle_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_in_push_event(), MCFrontEnd::handle_link_arrival(), MCFrontEnd::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), GenericRPG::handle_out_pull_event(), MCFrontEnd::handle_ready_event(), NI::set_output_path(), GenericTPG::set_output_path(), and GenericFlatMc::set_output_path().

Here is the call graph for this function:



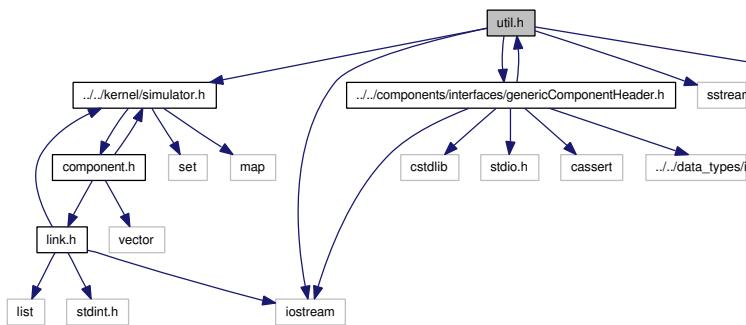
Here is the caller graph for this function:



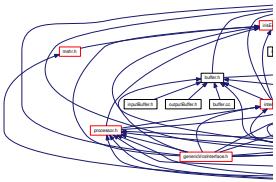
8.127 util.h File Reference

```
#include <iostream>
#include <sstream>
#include "../../kernel/simulator.h"
#include "../../../../components/interfaces/genericComponentHeader.h"
#include "../../../../components/interfaces/genericComponentHeader.h"

Include dependency graph for util.h:
```



This graph shows which files directly or indirectly include this file:



Functions

- void **timed_cout** (string str)

8.127.1 Function Documentation

8.127.1.1 void **timed_cout** (string str)

Definition at line 23 of file util.cc.

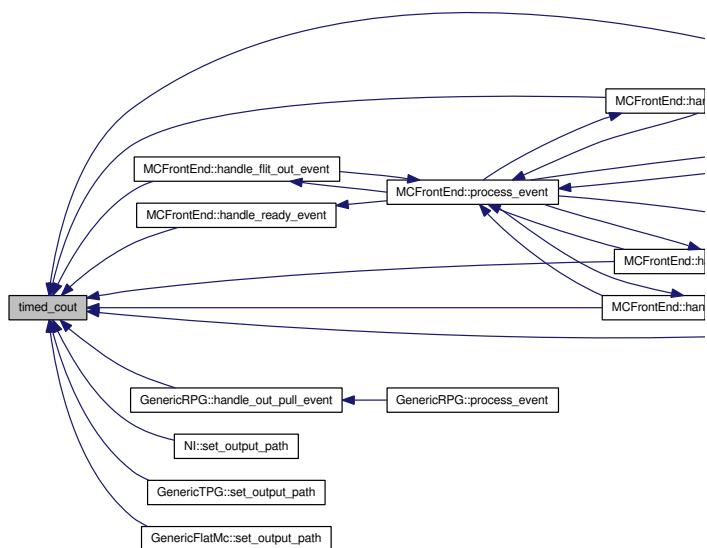
References Simulator::Now().

Referenced by MCFrontEnd::handle_flit_out_event(), MCFrontEnd::handle_in_arbitrate_event(), MCFrontEnd::handle_in_push_event(), MCFrontEnd::handle_link_arrival(), MCFrontEnd::handle_new_packet_event(), MCFrontEnd::handle_out_arbitrate_event(), GenericRPG::handle_out_pull_event(), MCFrontEnd::handle_ready_event(), NI::set_output_path(), GenericTPG::set_output_path(), and GenericFlatMc::set_output_path().

Here is the call graph for this function:



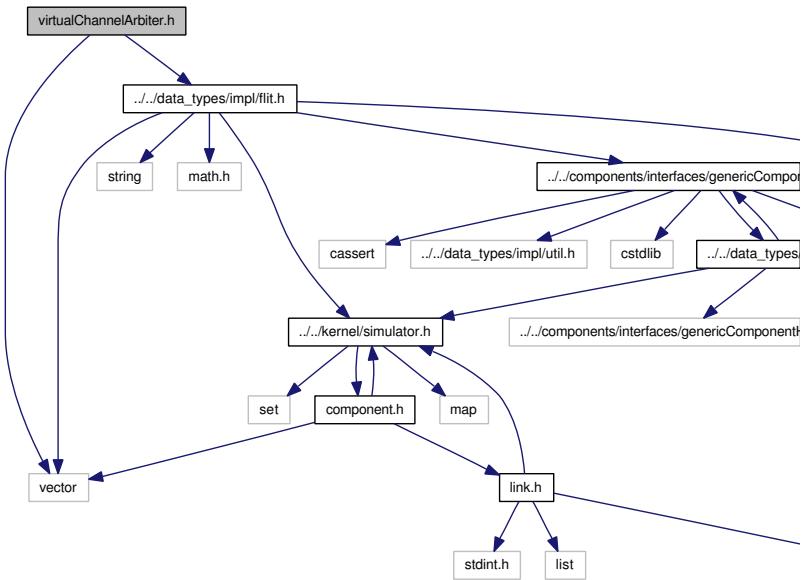
Here is the caller graph for this function:



8.128 virtualChannelArbiter.h File Reference

```
#include "../../data_types/impl/flit.h"
#include <vector>
```

Include dependency graph for virtualChannelArbiter.h:



Classes

- class **VirtualChannelArbiter**