TimingAnalysis

Generated by Doxygen 1.8.5

Wed Oct 23 2013 15:22:25

Contents

1 Main Page **1**

2 Namespace Index **3**

2.1 Namespace List 3

3 Hierarchical Index **5**

3.1 Class Hierarchy 5

4 Class Index **7**

4.1 Class List 7

5 File Index **11**

5.1 File List 11

6 Namespace Documentation **13**

6.1 std Namespace Reference 13

6.2 Timing_Analysis Namespace Reference 13

6.2.1 Detailed Description 14

6.2.2 Enumeration Type Documentation 14

6.2.2.1 Timing_Point_Type 14

7 Class Documentation **15**

7.1 SpefNetlSPD2013::Capacitor Struct Reference 15

7.1.1 Detailed Description 15

7.1.2 Constructor & Destructor Documentation 15

7.1.2.1 Capacitor 16

7.1.3 Member Data Documentation 17

7.1.3.1 node 17

7.1.3.2 value 17

7.2 Circuit_Netlist Class Reference 17

7.2.1 Detailed Description 19

7.2.2 Constructor & Destructor Documentation 19

7.2.2.1 Circuit_Netlist 19

7.2.2.2 ∼Circuit_Netlist 19

7.2.3 Member Function Documentation . . . 19
7.2.3.1 addCellInst . . . 19
7.2.3.2 get_net_topologic_index . . . 19
7.2.3.3 getGate . . . 19
7.2.3.4 getGatesSize . . . 20
7.2.3.5 getGateT . . . 20
7.2.3.6 getNet . . . 20
7.2.3.7 getNetsSize . . . 20
7.2.3.8 getNetT . . . 20
7.2.3.9 getTopologicIndex . . . 20
7.2.3.10 timing_arcs . . . 21
7.2.3.11 timing_points . . . 21
7.2.3.12 updateTopology . . . 21
7.2.3.13 verilog . . . 21
7.2.4 Friends And Related Function Documentation . . . 21
7.2.4.1 operator<< . . . 21
7.2.4.2 operator<< . . . 21
7.2.4.3 operator<< . . . 22
7.2.4.4 operator<< . . . 22
7.3 Design_Constraints Class Reference . . . 22
7.3.1 Detailed Description . . . 24
7.3.2 Member Function Documentation . . . 24
7.3.2.1 clock . . . 24
7.3.2.2 clock . . . 24
7.3.2.3 driving_cell . . . 24
7.3.2.4 driving_cell . . . 24
7.3.2.5 input_delay . . . 24
7.3.2.6 input_delay . . . 25
7.3.2.7 input_transition . . . 25
7.3.2.8 input_transition . . . 25
7.3.2.9 output_delay . . . 25
7.3.2.10 output_delay . . . 26
7.3.2.11 output_load . . . 27
7.3.2.12 output_load . . . 27
7.3.2.13 output_loads_size . . . 27
7.4 Timing_Analysis::Edge< T > Class Template Reference . . . 27
7.4.1 Detailed Description . . . 30
7.4.2 Constructor & Destructor Documentation . . . 30
7.4.2.1 Edge . . . 30
7.4.3 Member Function Documentation . . . 30

7.4.3.1 add_fanout . . . 30
7.4.3.2 fanouts_size . . . 30
7.4.3.3 from . . . 30
7.4.3.4 set_fanout . . . 31
7.4.4 Member Data Documentation . . . 32
7.4.4.1 _from . . . 32
7.4.4.2 _to . . . 32
7.5 IF< cond, ThenType, ElseType > Struct Template Reference . . . 32
7.5.1 Detailed Description . . . 33
7.5.2 Member Typedef Documentation . . . 33
7.5.2.1 RET . . . 33
7.6 IF< false, ThenType, ElseType > Struct Template Reference . . . 33
7.6.1 Detailed Description . . . 33
7.6.2 Member Typedef Documentation . . . 34
7.6.2.1 RET . . . 34
7.7 LibertyCellInfo Struct Reference . . . 34
7.7.1 Detailed Description . . . 35
7.7.2 Constructor & Destructor Documentation . . . 35
7.7.2.1 LibertyCellInfo . . . 35
7.7.3 Member Data Documentation . . . 35
7.7.3.1 area . . . 35
7.7.3.2 dontTouch . . . 35
7.7.3.3 footprint . . . 35
7.7.3.4 isSequential . . . 35
7.7.3.5 leakagePower . . . 35
7.7.3.6 name . . . 35
7.7.3.7 pins . . . 35
7.7.3.8 primaryOutput . . . 35
7.7.3.9 timingArcs . . . 35
7.8 LibertyLibrary Class Reference . . . 35
7.8.1 Detailed Description . . . 36
7.8.2 Constructor & Destructor Documentation . . . 36
7.8.2.1 LibertyLibrary . . . 36
7.8.2.2 ~LibertyLibrary . . . 37
7.8.3 Member Function Documentation . . . 37
7.8.3.1 addCellInfo . . . 37
7.8.3.2 getCellIndex . . . 37
7.8.3.3 getCellInfo . . . 37
7.8.3.4 getCellInfo . . . 37
7.8.3.5 getCellInfo . . . 38

7.8.3.6 getMaxTransition . . . 38
7.8.3.7 number_of_options . . . 38
7.9 LibertyLookupTable Struct Reference . . . 38
7.9.1 Detailed Description . . . 39
7.9.2 Member Data Documentation . . . 39
7.9.2.1 loadIndices . . . 39
7.9.2.2 tableVals . . . 39
7.9.2.3 transitionIndices . . . 40
7.10 LibertyLookupTableInterpolator Class Reference . . . 40
7.10.1 Detailed Description . . . 41
7.10.2 Member Function Documentation . . . 41
7.10.2.1 interpolate . . . 41
7.10.2.2 interpolate . . . 41
7.10.2.3 round . . . 42
7.10.3 Member Data Documentation . . . 42
7.10.3.1 DEFAULT_DECIMAL_PLACES . . . 42
7.11 LibertyParser Class Reference . . . 42
7.11.1 Detailed Description . . . 43
7.11.2 Member Function Documentation . . . 43
7.11.2.1 readFile . . . 44
7.12 LibertyPinInfo Struct Reference . . . 45
7.12.1 Detailed Description . . . 46
7.12.2 Constructor & Destructor Documentation . . . 46
7.12.2.1 LibertyPinInfo . . . 46
7.12.3 Member Data Documentation . . . 46
7.12.3.1 capacitance . . . 46
7.12.3.2 isClock . . . 46
7.12.3.3 isInput . . . 46
7.12.3.4 maxCapacitance . . . 46
7.12.3.5 name . . . 46
7.13 LibertyTimingInfo Struct Reference . . . 46
7.13.1 Detailed Description . . . 48
7.13.2 Member Data Documentation . . . 48
7.13.2.1 fallDelay . . . 48
7.13.2.2 fallTransition . . . 48
7.13.2.3 fromPin . . . 48
7.13.2.4 riseDelay . . . 48
7.13.2.5 riseTransition . . . 48
7.13.2.6 timingSense . . . 48
7.13.2.7 toPin . . . 48

7.14 LinearLibertyLookupTableInterpolator Class Reference . . . 48
7.14.1 Detailed Description . . . 50
7.14.2 Member Function Documentation . . . 50
7.14.2.1 interpolate . . . 50
7.14.2.2 interpolate . . . 50
7.15 Circuit_Netlist::Logic_Gate Struct Reference . . . 51
7.15.1 Detailed Description . . . 52
7.15.2 Constructor & Destructor Documentation . . . 52
7.15.2.1 Logic_Gate . . . 52
7.15.3 Member Data Documentation . . . 52
7.15.3.1 cellType . . . 52
7.15.3.2 fanoutNetIndex . . . 52
7.15.3.3 inNets . . . 52
7.15.3.4 inputDriver . . . 52
7.15.3.5 name . . . 52
7.15.3.6 primary_output . . . 52
7.15.3.7 sequential . . . 52
7.16 LumpedCapacitanceWireDelayModel Class Reference . . . 52
7.16.1 Detailed Description . . . 55
7.16.2 Constructor & Destructor Documentation . . . 55
7.16.2.1 LumpedCapacitanceWireDelayModel . . . 55
7.16.3 Member Function Documentation . . . 55
7.16.3.1 clear . . . 55
7.16.3.2 delay_at_fanout_node . . . 55
7.16.3.3 root_delay . . . 56
7.16.3.4 root_slew . . . 56
7.16.3.5 setFanoutPinCapacitance . . . 56
7.16.3.6 simulate . . . 56
7.16.3.7 slew_at_fanout_node . . . 57
7.17 Timing_Analysis::Multi_Fanout_Edge< T > Class Template Reference . . . 57
7.17.1 Detailed Description . . . 60
7.17.2 Constructor & Destructor Documentation . . . 60
7.17.2.1 Multi_Fanout_Edge . . . 60
7.17.3 Member Function Documentation . . . 60
7.17.3.1 to . . . 60
7.18 Circuit_Netlist::Net Struct Reference . . . 60
7.18.1 Detailed Description . . . 62
7.18.2 Constructor & Destructor Documentation . . . 62
7.18.2.1 Net . . . 62
7.18.3 Member Function Documentation . . . 62

7.18.3.1 addSink . . . 62
7.18.4 Member Data Documentation . . . 62
7.18.4.1 dummyNet . . . 62
7.18.4.2 name . . . 62
7.18.4.3 sinks . . . 62
7.18.4.4 sourceNode . . . 62
7.18.4.5 sourcePin . . . 62
7.19 SpefNetISPD2013::Node Struct Reference . . . 62
7.19.1 Detailed Description . . . 63
7.19.2 Constructor & Destructor Documentation . . . 63
7.19.2.1 Node . . . 64
7.19.3 Member Data Documentation . . . 65
7.19.3.1 capacitance . . . 65
7.19.3.2 name . . . 65
7.19.3.3 nodeIndex . . . 65
7.19.3.4 resistors . . . 65
7.20 std::numeric_limits< Transitions< double > > Class Template Reference . . . 65
7.20.1 Detailed Description . . . 66
7.20.2 Member Function Documentation . . . 66
7.20.2.1 max . . . 66
7.20.2.2 min . . . 66
7.20.2.3 zero . . . 66
7.21 Timing_Analysis::Option Class Reference . . . 66
7.21.1 Detailed Description . . . 67
7.21.2 Constructor & Destructor Documentation . . . 67
7.21.2.1 Option . . . 67
7.21.2.2 Option . . . 67
7.21.3 Friends And Related Function Documentation . . . 67
7.21.3.1 Timing_Analysis . . . 67
7.22 Parser Class Reference . . . 68
7.22.1 Detailed Description . . . 69
7.22.2 Constructor & Destructor Documentation . . . 69
7.22.2.1 Parser . . . 69
7.22.2.2 ~Parser . . . 69
7.22.3 Member Function Documentation . . . 69
7.22.3.1 isSpecialChar . . . 69
7.22.3.2 readLineAsTokens . . . 69
7.22.4 Member Data Documentation . . . 69
7.22.4.1 is . . . 69
7.23 Prime_Time_Output_Parser::Pin_Timing Struct Reference . . . 69

7.23.1 Detailed Description . . . 71
7.23.2 Friends And Related Function Documentation . . . 71
7.23.2.1 operator<< . . . 71
7.23.3 Member Data Documentation . . . 71
7.23.3.1 arrival_time . . . 71
7.23.3.2 pin_name . . . 71
7.23.3.3 slack . . . 71
7.23.3.4 slew . . . 71
7.24 Prime_Time_Output_Parser::Port_Timing Struct Reference . . . 71
7.24.1 Detailed Description . . . 73
7.24.2 Friends And Related Function Documentation . . . 73
7.24.2.1 operator<< . . . 73
7.24.3 Member Data Documentation . . . 73
7.24.3.1 port_name . . . 73
7.24.3.2 slack . . . 73
7.24.3.3 slew . . . 73
7.25 Prime_Time_Output_Parser::Prime_Time_Output Class Reference . . . 73
7.25.1 Detailed Description . . . 74
7.25.2 Member Function Documentation . . . 74
7.25.2.1 pin . . . 74
7.25.2.2 pins_size . . . 74
7.25.2.3 port . . . 74
7.25.2.4 ports_size . . . 75
7.25.3 Friends And Related Function Documentation . . . 75
7.25.3.1 Prime_Time_Output_Parser . . . 75
7.26 Prime_Time_Output_Parser Class Reference . . . 75
7.26.1 Detailed Description . . . 77
7.26.2 Constructor & Destructor Documentation . . . 77
7.26.2.1 Prime_Time_Output_Parser . . . 77
7.26.3 Member Function Documentation . . . 77
7.26.3.1 parse_prime_time_output_file . . . 77
7.27 RCTreeWireDelayModel Class Reference . . . 77
7.27.1 Detailed Description . . . 80
7.27.2 Constructor & Destructor Documentation . . . 80
7.27.2.1 RCTreeWireDelayModel . . . 80
7.27.3 Member Function Documentation . . . 80
7.27.3.1 clear . . . 80
7.27.3.2 delay_at_fanout_node . . . 80
7.27.3.3 root_delay . . . 81
7.27.3.4 root_slew . . . 81

7.27.3.5 setFanoutPinCapacitance . . . 81
7.27.3.6 simulate . . . 81
7.27.3.7 slew_at_fanout_node . . . 82
7.28 SpefNetISPD2013::Resistor Struct Reference . . . 82
7.28.1 Detailed Description . . . 83
7.28.2 Constructor & Destructor Documentation . . . 83
7.28.2.1 Resistor . . . 83
7.28.3 Member Function Documentation . . . 83
7.28.3.1 getOtherNode . . . 83
7.28.4 Member Data Documentation . . . 83
7.28.4.1 node1 . . . 83
7.28.4.2 node2 . . . 83
7.28.4.3 value . . . 83
7.29 Timer::Result Class Reference . . . 84
7.29.1 Detailed Description . . . 84
7.29.2 Member Function Documentation . . . 84
7.29.2.1 set . . . 84
7.29.3 Friends And Related Function Documentation . . . 84
7.29.3.1 operator<< . . . 84
7.29.3.2 Timer . . . 85
7.30 SDCParser Class Reference . . . 85
7.30.1 Detailed Description . . . 86
7.30.2 Member Function Documentation . . . 86
7.30.2.1 readFile . . . 86
7.31 Timing_Analysis::Single_Fanout_Edge< T > Class Template Reference . . . 87
7.31.1 Detailed Description . . . 90
7.31.2 Constructor & Destructor Documentation . . . 90
7.31.2.1 Single_Fanout_Edge . . . 90
7.31.3 Member Function Documentation . . . 90
7.31.3.1 to . . . 90
7.32 Circuit_Netlist::Sink Struct Reference . . . 90
7.32.1 Detailed Description . . . 91
7.32.2 Constructor & Destructor Documentation . . . 91
7.32.2.1 Sink . . . 91
7.32.3 Member Data Documentation . . . 91
7.32.3.1 gate . . . 91
7.32.3.2 pinName . . . 92
7.33 SpefNetISPD2012 Class Reference . . . 92
7.33.1 Detailed Description . . . 92
7.33.2 Constructor & Destructor Documentation . . . 93

7.33.2.1 SpefNetISPD2012 . . . 93
7.33.3 Member Data Documentation . . . 93
7.33.3.1 netLumpedCap . . . 93
7.33.3.2 netName . . . 93
7.34 SpefNetISPD2013 Class Reference . . . 93
7.34.1 Detailed Description . . . 94
7.34.2 Constructor & Destructor Documentation . . . 95
7.34.2.1 SpefNetISPD2013 . . . 95
7.34.2.2 ∼SpefNetISPD2013 . . . 95
7.34.3 Member Function Documentation . . . 95
7.34.3.1 addCapacitor . . . 95
7.34.3.2 addResistor . . . 95
7.34.3.3 getNode . . . 95
7.34.3.4 getNodeIndex . . . 95
7.34.3.5 getResistor . . . 96
7.34.3.6 nodesSize . . . 96
7.34.3.7 resistorsSize . . . 96
7.34.3.8 set . . . 96
7.34.4 Friends And Related Function Documentation . . . 96
7.34.4.1 operator<< . . . 96
7.34.5 Member Data Documentation . . . 97
7.34.5.1 netLumpedCap . . . 97
7.34.5.2 netName . . . 97
7.35 SpefParserISPD2012 Class Reference . . . 97
7.35.1 Detailed Description . . . 98
7.35.2 Member Function Documentation . . . 98
7.35.2.1 readFile . . . 98
7.36 SpefParserISPD2013 Class Reference . . . 99
7.36.1 Detailed Description . . . 100
7.36.2 Member Function Documentation . . . 100
7.36.2.1 readFile . . . 100
7.37 Timer Class Reference . . . 101
7.37.1 Detailed Description . . . 102
7.37.2 Constructor & Destructor Documentation . . . 102
7.37.2.1 Timer . . . 102
7.37.2.2 ∼Timer . . . 102
7.37.3 Member Function Documentation . . . 102
7.37.3.1 end . . . 102
7.37.3.2 start . . . 102
7.37.3.3 value . . . 102

7.37.4 Member Data Documentation . . . 103
7.37.4.1 MICRO . . . 103
7.37.4.2 MILI . . . 103
7.38 TimerInterface Class Reference . . . 103
7.38.1 Detailed Description . . . 103
7.38.2 Member Enumeration Documentation . . . 104
7.38.2.1 Status . . . 104
7.38.3 Member Function Documentation . . . 104
7.38.3.1 getTimerStatus . . . 104
7.38.3.2 runTimingAnalysisBlocking . . . 104
7.38.3.3 startTimingAnalysisNonBlocking . . . 104
7.38.3.4 wait . . . 105
7.39 Timing_Analysis::Timing_Analysis Class Reference . . . 105
7.39.1 Detailed Description . . . 106
7.39.2 Constructor & Destructor Documentation . . . 106
7.39.2.1 Timing_Analysis . . . 106
7.39.2.2 ∼Timing_Analysis . . . 106
7.39.3 Member Function Documentation . . . 107
7.39.3.1 call_prime_time . . . 107
7.39.3.2 first_PO_index . . . 107
7.39.3.3 full_timing_analysis . . . 107
7.39.3.4 number_of_options . . . 107
7.39.3.5 option . . . 107
7.39.3.6 option . . . 107
7.39.3.7 pin_capacitance . . . 107
7.39.3.8 pin_load . . . 108
7.39.3.9 print_circuit_info . . . 108
7.39.3.10 print_info . . . 108
7.39.3.11 report_timing . . . 108
7.39.3.12 timing_arc . . . 108
7.39.3.13 timing_arcs_size . . . 109
7.39.3.14 timing_net . . . 109
7.39.3.15 timing_nets_size . . . 109
7.39.3.16 timing_point . . . 109
7.39.3.17 timing_points_size . . . 109
7.39.3.18 validate_with_prime_time . . . 110
7.40 Timing_Analysis::Timing_Arc Class Reference . . . 110
7.40.1 Detailed Description . . . 113
7.40.2 Constructor & Destructor Documentation . . . 113
7.40.2.1 Timing_Arc . . . 113

7.40.2.2 ∼Timing_Arc . . . 113
7.40.3 Member Function Documentation . . . 113
7.40.3.1 arc_number . . . 113
7.40.3.2 clear . . . 114
7.40.3.3 delay . . . 114
7.40.3.4 delay . . . 114
7.40.3.5 gate_number . . . 115
7.40.3.6 slew . . . 115
7.40.3.7 slew . . . 115
7.40.4 Friends And Related Function Documentation . . . 116
7.40.4.1 operator<< . . . 116
7.41 Timing_Analysis::Timing_Net Class Reference . . . 116
7.41.1 Detailed Description . . . 119
7.41.2 Constructor & Destructor Documentation . . . 119
7.41.2.1 Timing_Net . . . 119
7.41.2.2 ∼Timing_Net . . . 119
7.41.3 Member Function Documentation . . . 119
7.41.3.1 name . . . 119
7.41.3.2 wire_delay_model . . . 119
7.41.4 Friends And Related Function Documentation . . . 120
7.41.4.1 operator<< . . . 120
7.41.4.2 Timing_Analysis . . . 120
7.41.4.3 Timing_Point . . . 120
7.42 Timing_Analysis::Timing_Point Class Reference . . . 120
7.42.1 Detailed Description . . . 122
7.42.2 Constructor & Destructor Documentation . . . 122
7.42.2.1 Timing_Point . . . 122
7.42.2.2 ∼Timing_Point . . . 122
7.42.3 Member Function Documentation . . . 122
7.42.3.1 arc . . . 122
7.42.3.2 arc . . . 122
7.42.3.3 arc . . . 122
7.42.3.4 arrival_time . . . 123
7.42.3.5 arrival_time . . . 123
7.42.3.6 ceff . . . 124
7.42.3.7 ceff . . . 124
7.42.3.8 clear_timing_info . . . 125
7.42.3.9 gate_number . . . 125
7.42.3.10 is_input_pin . . . 125
7.42.3.11 is_output_pin . . . 125

7.42.3.12 is_PI . . . 125
7.42.3.13 is_PI_input . . . 126
7.42.3.14 is_PO . . . 126
7.42.3.15 is_reg_input . . . 126
7.42.3.16 load . . . 126
7.42.3.17 name . . . 126
7.42.3.18 net . . . 126
7.42.3.19 net . . . 126
7.42.3.20 net . . . 127
7.42.3.21 required_time . . . 127
7.42.3.22 slack . . . 127
7.42.3.23 slack . . . 128
7.42.3.24 slew . . . 128
7.42.3.25 slew . . . 128
7.42.3.26 update_slack . . . 129
7.42.4 Friends And Related Function Documentation . . . 129
7.42.4.1 operator<< . . . 129
7.42.4.2 Timing_Net . . . 129
7.43 Traits Class Reference . . . 130
7.43.1 Detailed Description . . . 130
7.43.2 Member Data Documentation . . . 130
7.43.2.1 ISPD_2012 . . . 130
7.43.2.2 ispd_contest_benchmark . . . 130
7.43.2.3 ispd_contest_root . . . 130
7.43.2.4 STD_THRESHOLD . . . 131
7.44 Transitions< T > Class Template Reference . . . 131
7.44.1 Detailed Description . . . 134
7.44.2 Constructor & Destructor Documentation . . . 134
7.44.2.1 Transitions . . . 134
7.44.2.2 Transitions . . . 134
7.44.3 Member Function Documentation . . . 135
7.44.3.1 aggregate . . . 135
7.44.3.2 getFall . . . 135
7.44.3.3 getMax . . . 135
7.44.3.4 getMin . . . 136
7.44.3.5 getReversed . . . 136
7.44.3.6 getRise . . . 136
7.44.3.7 operator= . . . 136
7.44.3.8 operator[] . . . 136
7.44.3.9 operator[] . . . 137

7.44.3.10 reverse . . . 137
7.44.3.11 set . . . 137
7.44.4 Friends And Related Function Documentation . . . 137
7.44.4.1 abs . . . 137
7.44.4.2 exp . . . 137
7.44.4.3 max . . . 138
7.44.4.4 min . . . 138
7.44.4.5 operator<< . . . 138
7.44.4.6 pow . . . 138
7.44.4.7 sqrt . . . 138
7.45 VerilogParser Class Reference . . . 139
7.45.1 Detailed Description . . . 140
7.45.2 Constructor & Destructor Documentation . . . 140
7.45.2.1 ~VerilogParser . . . 141
7.45.3 Member Function Documentation . . . 141
7.45.3.1 readFile . . . 141
7.46 WireDelayModel Class Reference . . . 141
7.46.1 Detailed Description . . . 144
7.46.2 Constructor & Destructor Documentation . . . 144
7.46.2.1 WireDelayModel . . . 144
7.46.2.2 ~WireDelayModel . . . 144
7.46.3 Member Function Documentation . . . 144
7.46.3.1 clear . . . 144
7.46.3.2 delay_at_fanout_node . . . 145
7.46.3.3 lumped_capacitance . . . 145
7.46.3.4 root_delay . . . 145
7.46.3.5 root_slew . . . 145
7.46.3.6 setFanoutPinCapacitance . . . 145
7.46.3.7 simulate . . . 146
7.46.3.8 slew_at_fanout_node . . . 146
7.46.4 Member Data Documentation . . . 146
7.46.4.1 _lumped_capacitance . . . 146
7.46.4.2 interpolator . . . 146

8 File Documentation **147**
8.1 TimingAnalysis/doc/Index.dox File Reference . . . 147
8.2 TimingAnalysis/src/include/circuit_netlist.h File Reference . . . 147
8.3 TimingAnalysis/src/include/configuration.h File Reference . . . 148
8.3.1 Typedef Documentation . . . 150
8.3.1.1 Parasitics . . . 150

8.3.1.2 SpefNet . . . 150
8.3.1.3 SpefParser . . . 150
8.4 TimingAnalysis/src/include/design_constraints.h File Reference . . . 150
8.5 TimingAnalysis/src/include/edge.h File Reference . . . 151
8.6 TimingAnalysis/src/include/liberty_library.h File Reference . . . 153
8.6.1 Enumeration Type Documentation . . . 155
8.6.1.1 Unateness . . . 155
8.6.2 Function Documentation . . . 155
8.6.2.1 operator<< . . . 155
8.6.2.2 operator<< . . . 155
8.6.2.3 operator<< . . . 155
8.6.2.4 operator<< . . . 155
8.7 TimingAnalysis/src/include/multi_fanout_edge.h File Reference . . . 156
8.8 TimingAnalysis/src/include/parser.h File Reference . . . 157
8.9 TimingAnalysis/src/include/single_fanout_edge.h File Reference . . . 159
8.10 TimingAnalysis/src/include/spef_net.h File Reference . . . 161
8.10.1 Typedef Documentation . . . 163
8.10.1.1 Parasitics2012 . . . 163
8.10.1.2 Parasitics2013 . . . 163
8.11 TimingAnalysis/src/include/timer.h File Reference . . . 163
8.12 TimingAnalysis/src/include/timer_interface.h File Reference . . . 164
8.13 TimingAnalysis/src/include/timing_analysis.h File Reference . . . 165
8.14 TimingAnalysis/src/include/timing_arc.h File Reference . . . 166
8.15 TimingAnalysis/src/include/timing_net.h File Reference . . . 168
8.16 TimingAnalysis/src/include/timing_point.h File Reference . . . 169
8.17 TimingAnalysis/src/include/transitions.h File Reference . . . 170
8.17.1 Macro Definition Documentation . . . 172
8.17.1.1 MAKE_OPERATOR . . . 172
8.17.1.2 MAKE_SELF_OPERATOR . . . 172
8.17.2 Enumeration Type Documentation . . . 172
8.17.2.1 EdgeType . . . 172
8.18 TimingAnalysis/src/include/wire_delay_model.h File Reference . . . 172

Index 175

Chapter 1

Main Page

Set ISPD Contest Edition in the include/Configuration.h file

Naming Convention

Class Name: Class_Name Header File: src/include/class_name.h Source File: src/class_name.cpp

Code Style

Class_Name::class_method(int method_parameter) { .....int local_var = 0; .....return local_var; }
..... is a <TAB>

To run

./RunTimer <BENCHMARK> <2012 or 2013>

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

std 13

Timing_Analysis

Namespace which contains classes relative to timing analysis graph model 13

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

SpefNetISPD2013::Capacitor . . . 15
Circuit_Netlist . . . 17
Design_Constraints . . . 22
Timing_Analysis::Edge< T > . . . 27
  Timing_Analysis::Multi_Fanout_Edge< T > . . . 57
  Timing_Analysis::Single_Fanout_Edge< T > . . . 87
Timing_Analysis::Edge< Timing_Point > . . . 27
  Timing_Analysis::Multi_Fanout_Edge< Timing_Point > . . . 57
    Timing_Analysis::Timing_Net . . . 116
  Timing_Analysis::Single_Fanout_Edge< Timing_Point > . . . 87
    Timing_Analysis::Timing_Arc . . . 110
IF< cond, ThenType, ElseType > . . . 32
IF< false, ThenType, ElseType > . . . 33
LibertyCellInfo . . . 34
LibertyLibrary . . . 35
LibertyLookupTable . . . 38
LibertyLookupTableInterpolator . . . 40
  LinearLibertyLookupTableInterpolator . . . 48
LibertyPinInfo . . . 45
LibertyTimingInfo . . . 46
Circuit_Netlist::Logic_Gate . . . 51
Circuit_Netlist::Net . . . 60
SpefNetISPD2013::Node . . . 62
std::numeric_limits< Transitions< double > > . . . 65
Timing_Analysis::Option . . . 66
Parser . . . 68
  LibertyParser . . . 42
  Prime_Time_Output_Parser . . . 75
  SDCParser . . . 85
  SpefParserISPD2012 . . . 97
  SpefParserISPD2013 . . . 99
  VerilogParser . . . 139
Prime_Time_Output_Parser::Pin_Timing . . . 69
Prime_Time_Output_Parser::Port_Timing . . . 71
Prime_Time_Output_Parser::Prime_Time_Output . . . 73
SpefNetISPD2013::Resistor . . . 82
Timer::Result . . . 84

Circuit_Netlist::Sink 90
SpefNetISPD2012 92
SpefNetISPD2013 93
Timer 101
TimerInterface 103
Timing_Analysis::Timing_Analysis 105
Timing_Analysis::Timing_Point 120
Traits 130
Transitions< T > 131
Transitions< double > 131
WireDelayModel 141
LumpedCapacitanceWireDelayModel 52
RCTreeWireDelayModel 77

Chapter 4

Class Index

4.1 Class List

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

SpefNetISPD2013::Capacitor
Struct which represents a capacitor 15

Circuit_Netlist
File which describe the set of elements of the circuit. Generated by the VerilogParser class . . 17

Design_Constraints
Represents the design constraints of the case in study. Generated by the SDCParser class . . 22

Timing_Analysis::Edge< T >
Template class which represents an edge of the graph model. An edge in graph theory is a connection between vertices 27

IF< cond, ThenType, ElseType >
This is a Configuration File To switch between ISPD2012 (Lumped Capacitance Wire Delay Model) or ISPD2013 (Distributed RC Wire Delay Model) SPEF format 32

IF< false, ThenType, ElseType >
Metaprogrammed IF 33

LibertyCellInfo
Struct which has informations about a cell 34

LibertyLibrary
Standard cell Library is a collection of logic functions info. Contains details about implementations of logic functions, within its layout, parasitic models, Verilog models, and more. Generated by the LibertyParser class 35

LibertyLookupTable
Struct representing a lookup table to store delay or slew functions 38

LibertyLookupTableInterpolator
Interpolation calculator 40

LibertyParser
Inherits from Parser. Used to generate a LibertyLibrary object from a file 42

LibertyPinInfo
Struct which has informations about a pin 45

LibertyTimingInfo
Struct which has informations about timing 46

LinearLibertyLookupTableInterpolator
Linear interpolation calculator 48

Circuit_Netlist::Logic_Gate
Struct which represents a Logic_Gate 51

LumpedCapacitanceWireDelayModel
This model describes the parasitic capacitances distributed along a wire as a single circuit element. Does not take resistors into account 52

Timing_Analysis::Multi_Fanout_Edge< T >
Inherits from Edge. An multi fanout edge is a group of edges which connects a vertex to more than one vertexes . . . 57
Circuit_Netlist::Net
Struct which represents a Net, which connect two or more circuit elements together . . . 60
SpefNetISPD2013::Node
Struct which represents a node of the circuit. A node is a point of the circuit where two or more elements meet . . . 62
std::numeric_limits< Transitions< double > >
Class used to make easier to work with double variable numeric limits . . . 65
Timing_Analysis::Option
Refers to the implementation option of a logic gate . . . 66
Parser
Superclass. A Parser is a program which reads an input and creates a data structure. In this case, the Parser will read a string, validate it and build essential files for use in the program using information read from the string . . . 68
Prime_Time_Output_Parser::Pin_Timing
Struct containing the timing values of a pin . . . 69
Prime_Time_Output_Parser::Port_Timing
Struct containing the timing values of a port . . . 71
Prime_Time_Output_Parser::Prime_Time_Output
Represents Prime Time output. Provides access methods to it . . . 73
Prime_Time_Output_Parser
Inherits from Parser. Parse the Prime Time output. Used to generate a Prime_Time_Output object for later use . . . 75
RCTreeWireDelayModel
This model describes the parasitic elements(capacitors and resistors) distributed along a wire as a single circuit element . . . 77
SpefNetISPD2013::Resistor
Struct which represents a resistor . . . 82
Timer::Result
Represents the result from the Timer class . . . 84
SDCParser
Inherits from Parser. SDC stands for Synopsys Design Constraints. Used to generate a Design_Constraints object from file . . . 85
Timing_Analysis::Single_Fanout_Edge< T >
Inherits from Edge. A single fanout edge is an edge which connects two vertex to each other . 87
Circuit_Netlist::Sink
Struct which represents a Sink. A sink is a vertex which has input Edge's, but no output Edge's (outdegree zero) . . . 90
SpefNetISPD2012
SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2012 92
SpefNetISPD2013
SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2013 93
SpefParserISPD2012
Inherits from Parser. Used to generate SpefNetISPD2012 files from an input . . . 97
SpefParserISPD2013
Inherits from Parser. Used to generate SpefNetISPD2013 file from an input . . . 99
Timer
Represents a Timer . . . 101
TimerInterface
This class contains functions for the timing analysis interface. To use any function belonging to this class, call TimerInterface::<function_name>(<argument_list>); . . . 103
Timing_Analysis::Timing_Analysis
Class which performs the timing analysis . . . 105

Timing_Analysis::Timing_Arc
Describes a Timing_Arc of the timing graph model. Vertex which connects one Timing_Point to another timing point only . . . 110
Timing_Analysis::Timing_Net
Describes a Timing_Net of the timing graph model. Vertex which connects one Timing_Point to multiple Timing_Point's . . . 116
Timing_Analysis::Timing_Point
Describes a Timing_Point of the timing graph model . . . 120
Traits
User configurable . . . 130
Transitions< T >
Template class which encapsulates any [rise,fall] twosome values and provides methods to work with it . . . 131
VerilogParser
Inherits from Parser. Parse verilog files. Used to generate circuit netlists . . . 139
WireDelayModel
Describes parasitc elements present in wires in a circuit which cause significant delay and are taken into account in static timing analysis . . . 141

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

TimingAnalysis/src/include/circuit_netlist.h 147
TimingAnalysis/src/include/configuration.h 148
TimingAnalysis/src/include/design_constraints.h 150
TimingAnalysis/src/include/edge.h 151
TimingAnalysis/src/include/liberty_library.h 153
TimingAnalysis/src/include/multi_fanout_edge.h 156
TimingAnalysis/src/include/parser.h 157
TimingAnalysis/src/include/single_fanout_edge.h 159
TimingAnalysis/src/include/spef_net.h 161
TimingAnalysis/src/include/timer.h 163
TimingAnalysis/src/include/timer_interface.h 164
TimingAnalysis/src/include/timing_analysis.h 165
TimingAnalysis/src/include/timing_arc.h 166
TimingAnalysis/src/include/timing_net.h 168
TimingAnalysis/src/include/timing_point.h 169
TimingAnalysis/src/include/transitions.h 170
TimingAnalysis/src/include/wire_delay_model.h 172

Chapter 6

Namespace Documentation

6.1 std Namespace Reference

Classes

- class numeric_limits< Transitions< double > >

  Class used to make easier to work with double variable numeric limits.

6.2 Timing_Analysis Namespace Reference

Namespace which contains classes relative to timing analysis graph model.

Classes

- class Edge

  Template class which represents an edge of the graph model. An edge in graph theory is a connection between vertices.
- class Multi_Fanout_Edge

  Inherits from Edge. An multi fanout edge is a group of edges which connects a vertex to more than one vertexes.
- class Single_Fanout_Edge

  Inherits from Edge. A single fanout edge is an edge which connects two vertex to each other.
- class Option

  Refers to the implementation option of a logic gate.
- class Timing_Analysis

  Class which performs the timing analysis.
- class Timing_Arc

  Describes a Timing_Arc of the timing graph model. Vertex which connects one Timing_Point to another timing point only.
- class Timing_Net

  Describes a Timing_Net of the timing graph model. Vertex which connects one Timing_Point to multiple Timing_Point's.
- class Timing_Point

  Describes a Timing_Point of the timing graph model.

Enumerations

- enum Timing_Point_Type {
  INPUT, OUTPUT, PI_INPUT, REGISTER_INPUT,
  PI, PO }

6.2.1 Detailed Description

Namespace which contains classes relative to timing analysis graph model.

6.2.2 Enumeration Type Documentation

6.2.2.1 enum Timing_Analysis::Timing_Point_Type

Enumerator

INPUT

OUTPUT

PI_INPUT

REGISTER_INPUT

PI

PO

Chapter 7

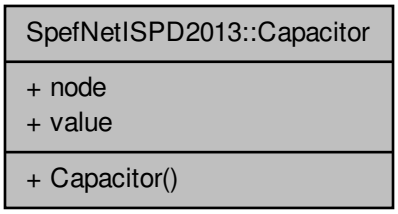
Class Documentation

7.1 SpefNetISPD2013::Capacitor Struct Reference

Struct which represents a capacitor.

`#include <spef_net.h>`

Collaboration diagram for SpefNetISPD2013::Capacitor:



Public Member Functions

- Capacitor (const int &node, const double &value)

  Capacitor constructor.

Public Attributes

- int node
- double value

7.1.1 Detailed Description

Struct which represents a capacitor.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 SpefNetISPD2013::Capacitor::Capacitor (const int & *node,* const double & *value*) [inline]

Capacitor constructor.

Parameters

const	string & node, const double & value

7.1.3 Member Data Documentation

7.1.3.1 int SpefNetlSPD2013::Capacitor::node

7.1.3.2 double SpefNetlSPD2013::Capacitor::value

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

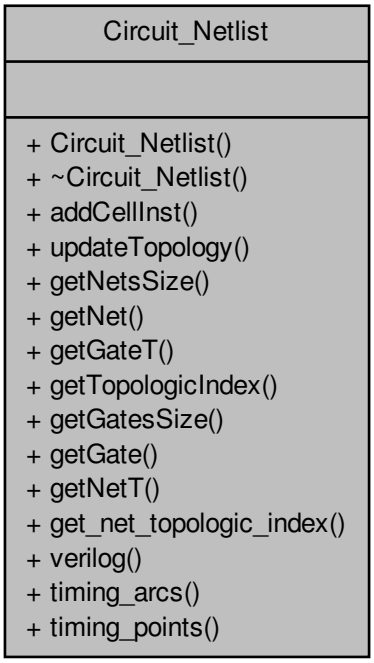
- TimingAnalysis/src/include/spef_net.h

7.2 Circuit_Netlist Class Reference

File which describe the set of elements of the circuit. Generated by the VerilogParser class.

```
#include <circuit_netlist.h>
```

Collaboration diagram for Circuit_Netlist:



Classes

- struct Logic_Gate

  Struct which represents a Logic_Gate.

- struct Net

  Struct which represents a Net, which connect two or more circuit elements together.
- struct Sink

  Struct which represents a Sink. A sink is a vertex which has input Edge's, but no output Edge's (outdegree zero).

Public Member Functions

- Circuit_Netlist ()

  Circuit_Netlist default constructor.
- virtual ~Circuit_Netlist ()

  Empty Circuit_Netlist destructor.
- void addCellInst (const string name, const string cellType, vector< pair< string, string > > inputPinPairs, const bool isSequential=false, const bool isInputDriver=false, const bool primary_output=false)

  Adds a cell instance.
- void updateTopology ()

  Updates netlist topology.
- size_t getNetsSize () const

  Returns number of nets.
- Net & getNet (const size_t &i)

  Returns Net & at index i.
- const Logic_Gate & getGateT (const size_t &i) const

  Returns Logic_Gate & at index i.
- int getTopologicIndex (const int &i) const

  Returns Topologic index of element at index i.
- size_t getGatesSize () const

  Returns number of gates.
- Logic_Gate & getGate (const size_t &i)

  Returns Logic_Gate & at index i.
- const Net & getNetT (const size_t &i) const

  Returns Net & at index i.
- int get_net_topologic_index (const size_t &i) const

  Returns topologic index of Net at index i.
- const vector< pair< int, string > > verilog () const

  Returns list of pairs containing Logic_Gate topologic number and its name.
- int timing_arcs () const

  Returns number of timings arcs.
- int timing_points () const

  Returns number of timings points.

Friends

- ostream & operator<< (ostream &out, const Circuit_Netlist netlist)

  Redefinition of << operator. Inserts formatted description of Netlist including net/gate lists' size and elements.
- ostream & operator<< (ostream &out, const Circuit_Netlist::Net net)

  Redefinition of << operator. Inserts formatted description of Net including source node/pin and list of sinks.
- ostream & operator<< (ostream &out, const Circuit_Netlist::Sink sink)

  Redefinition of << operator. Inserts formatted description of Sink including gate and pin name.
- ostream & operator<< (ostream &out, const Circuit_Netlist::Logic_Gate gate)

  Redefinition of << operator. Inserts formatted description of Logic_Gate including cell type, name, and input identifiers.

7.2.1 Detailed Description

File which describe the set of elements of the circuit. Generated by the VerilogParser class.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Circuit_Netlist::Circuit_Netlist ( ) [inline]

Circuit_Netlist default constructor.

7.2.2.2 virtual Circuit_Netlist::~Circuit_Netlist ( ) [inline], [virtual]

Empty Circuit_Netlist destructor.

7.2.3 Member Function Documentation

7.2.3.1 void Circuit_Netlist::addCellInst (const string *name*, const string *cellType*, vector< pair< string, string > > *inputPinPairs*, const bool *isSequential* = false, const bool *isInputDriver* = false, const bool *primary_output* = false)

Adds a cell instance.

Parameters

const	string name, const string cellType, vector<pair<string, string> > inputPinPairs, const bool isSequential(false default), const bool isInputDriver(false default), const bool *primary_output(false default)

Returns

void

7.2.3.2 int Circuit_Netlist::get_net_topologic_index (const size_t & *i*) const [inline]

Returns topologic index of Net at index i.

Parameters

const	size_t & i

Returns

int

7.2.3.3 Logic_Gate& Circuit_Netlist::getGate (const size_t & *i*) [inline]

Returns Logic_Gate & at index i.

Parameters

const	size_t & i

Returns

Logic_Gate &

7.2.3.4 size_t Circuit_Netlist::getGatesSize ( ) const [inline]

Returns number of gates.

Returns

size_t

7.2.3.5 const Logic_Gate& Circuit_Netlist::getGateT (const size_t & *i*) const [inline]

Returns Logic_Gate & at index i.

Parameters

const	size_t & i

Returns

const Logic_Gate &

7.2.3.6 Net& Circuit_Netlist::getNet (const size_t & *i*) [inline]

Returns Net & at index i.

Parameters

const	size_t & i

Returns

Net &

7.2.3.7 size_t Circuit_Netlist::getNetsSize ( ) const [inline]

Returns number of nets.

Returns

size_t

7.2.3.8 const Net& Circuit_Netlist::getNetT (const size_t & *i*) const [inline]

Returns Net & at index i.

Parameters

const	size_t & i

Returns

const Net &

7.2.3.9 int Circuit_Netlist::getTopologicIndex (const int & *i*) const [inline]

Returns Topologic index of element at index i.

Parameters

const	size_t & i

Returns

int

7.2.3.10 int Circuit_Netlist::timing_arcs ( ) const

Returns number of timings arcs.

Returns

int

7.2.3.11 int Circuit_Netlist::timing_points ( ) const

Returns number of timings points.

Returns

int

7.2.3.12 void Circuit_Netlist::updateTopology ( )

Updates netlist topology.

Returns

void

7.2.3.13 const vector<pair<int, string> > Circuit_Netlist::verilog ( ) const

Returns list of pairs containing Logic_Gate topologic number and its name.

Returns

const vector<pair<int, string> >

7.2.4 Friends And Related Function Documentation

7.2.4.1 ostream& operator<< (ostream & *out*, const Circuit_Netlist *netlist*) [friend]

Redefinition of << operator. Inserts formatted description of Netlist including net/gate lists' size and elements.

Parameters

ostream	& out, const Circuit_Netlist netlist

7.2.4.2 ostream& operator<< (ostream & *out*, const Circuit_Netlist::Net *net*) [friend]

Redefinition of << operator. Inserts formatted description of Net including source node/pin and list of sinks.

Parameters

ostream	& out, const Circuit_Netlist::Net net

7.2.4.3 ostream& operator<< (ostream & *out*, const Circuit_Netlist::Sink *sink*) [friend]

Redefinition of << operator. Inserts formatted description of Sink including gate and pin name.

Parameters

ostream	& out, const Sink sink

7.2.4.4 ostream& operator<< (ostream & *out*, const Circuit_Netlist::Logic_Gate *gate*) [friend]

Redefinition of << operator. Inserts formatted description of Logic_Gate including cell type, name, and input identifiers.

Parameters

ostream	& out, const Logic_Gate gate

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

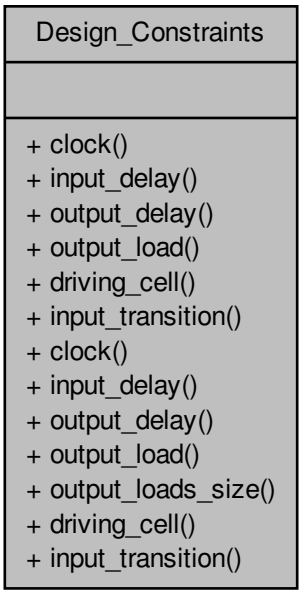
- TimingAnalysis/src/include/circuit_netlist.h

7.3 Design_Constraints Class Reference

Represents the design constraints of the case in study. Generated by the SDCParser class.

```
#include <design_constraints.h>
```

Collaboration diagram for Design_Constraints:



Public Member Functions

- void clock (const double clock)

  Sets clock period. frequency = 1 / clock_period.
- bool input_delay (const string input_name, const Transitions< double > delay)

  Tries to set input delay. Returns true if succeeded.
- bool output_delay (const string output_name, const Transitions< double > delay)

  Tries to set output delay. Returns true if succeeded.
- bool output_load (const string output_name, const double output_load)

  Tries to set output load. Returns true if succeeded.
- bool driving_cell (const string input_name, const string driving_cell)

  Tries to set driving cell. Returns true if succeeded.
- bool input_transition (const string input_name, const Transitions< double > transition)

  Tries to set input transitions. Returns true if succeeded.
- double clock () const

  Returns clock period.
- const Transitions< double > input_delay (const string input_name) const

  Returns Transitions<double> input_delay with input_name name.
- const Transitions< double > output_delay (const string output_name) const

  Returns Transitions<double> output_delay with output_name name.
- double output_load (const string output_name) const

  Returns output load with output_name name.
- size_t output_loads_size () const

  Returns output loads size.
- const string driving_cell (const string input_name) const

Returns driving cell with input_name name.

- const Transitions< double > input_transition (const string input_name) const

  Returns Transition<double> input_transition with input_name name.

7.3.1 Detailed Description

Represents the design constraints of the case in study. Generated by the SDCParser class.

7.3.2 Member Function Documentation

7.3.2.1 void Design_Constraints::clock (const double *clock*)

Sets clock period. frequency = 1 / clock_period.

Returns

void

7.3.2.2 double Design_Constraints::clock ( ) const

Returns clock period.

Returns

double

7.3.2.3 bool Design_Constraints::driving_cell (const string *input_name,* const string *driving_cell*)

Tries to set driving cell. Returns true if succeeded.

Parameters

const	string input_name, const string driving_cell

Returns

bool

7.3.2.4 const string Design_Constraints::driving_cell (const string *input_name*) const

Returns driving cell with input_name name.

Parameters

const	string input_name

Returns

const string

7.3.2.5 bool Design_Constraints::input_delay (const string *input_name,* const Transitions< double > *delay*)

Tries to set input delay. Returns true if succeeded.

Parameters

const	string input_name, const Transitions<double> delay

Returns

bool

7.3.2.6 const Transitions<double> Design_Constraints::input_delay (const string *input_name*) const

Returns Transitions<double> input_delay with input_name name.

Parameters

const	string input_name

Returns

const Transitions<double>

7.3.2.7 bool Design_Constraints::input_transition (const string *input_name,* const Transitions< double > *transition*)

Tries to set input transitions. Returns true if succeeded.

Parameters

const	string input_name, const Transitions<double> transition

Returns

bool

7.3.2.8 const Transitions<double> Design_Constraints::input_transition (const string *input_name*) const

Returns Transition<double> input_transition with input_name name.

Parameters

const	string input_name

Returns

const Transition<double>

7.3.2.9 bool Design_Constraints::output_delay (const string *output_name,* const Transitions< double > *delay*)

Tries to set output delay. Returns true if succeeded.

Parameters

const	string input_name, const Transitions<double> delay

Returns

bool

7.3.2.10 const Transitions<double> Design_Constraints::output_delay (const string *output_name*) const

Returns Transitions<double> output_delay with output_name name.

Parameters

const	string output_name

Returns

const Transitions<double>

7.3.2.11 bool Design_Constraints::output_load (const string *output_name,* const double *output_load*)

Tries to set output load. Returns true if succeeded.

Parameters

const	string input_name, const Transitions<double> output_load

Returns

bool

7.3.2.12 double Design_Constraints::output_load (const string *output_name*) const

Returns output load with output_name name.

Parameters

const	string output_name

Returns

double

7.3.2.13 size_t Design_Constraints::output_loads_size ( ) const

Returns output loads size.

Returns

size_t

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

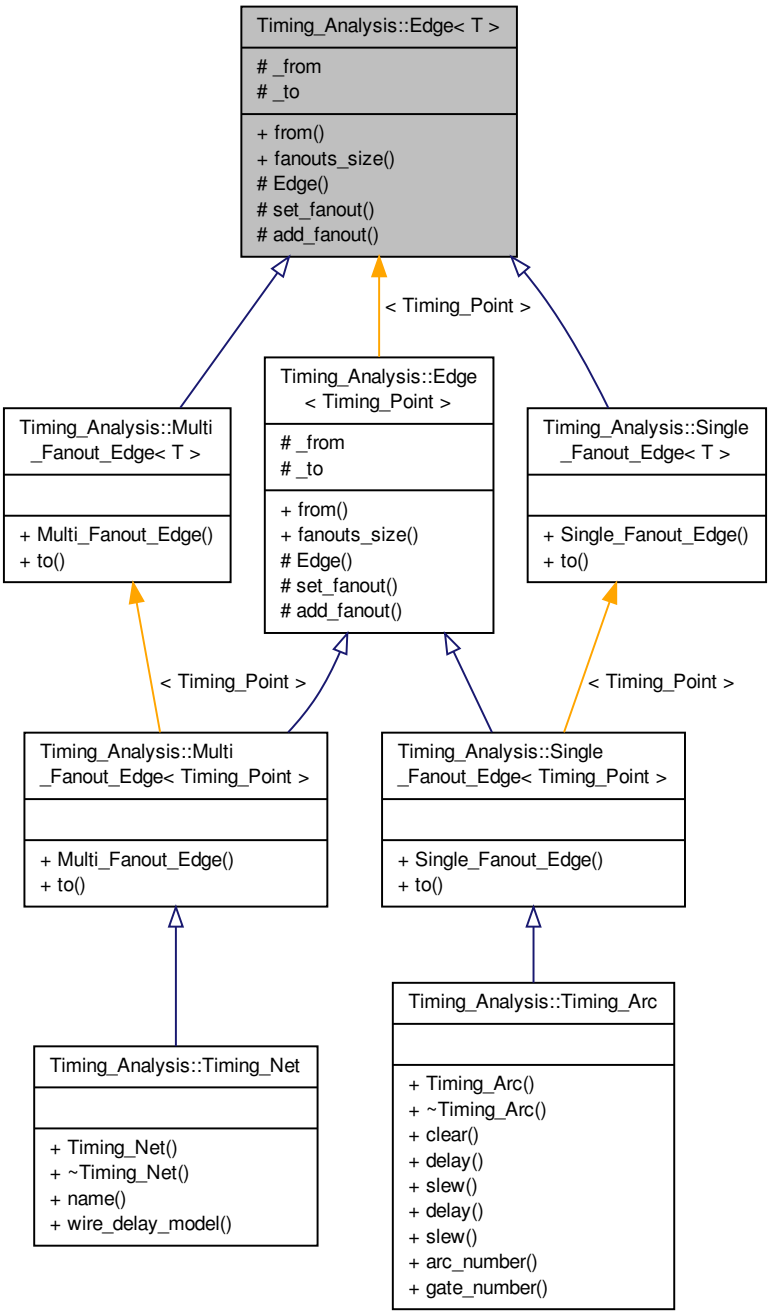
- TimingAnalysis/src/include/design_constraints.h

7.4 Timing_Analysis::Edge< T > Class Template Reference

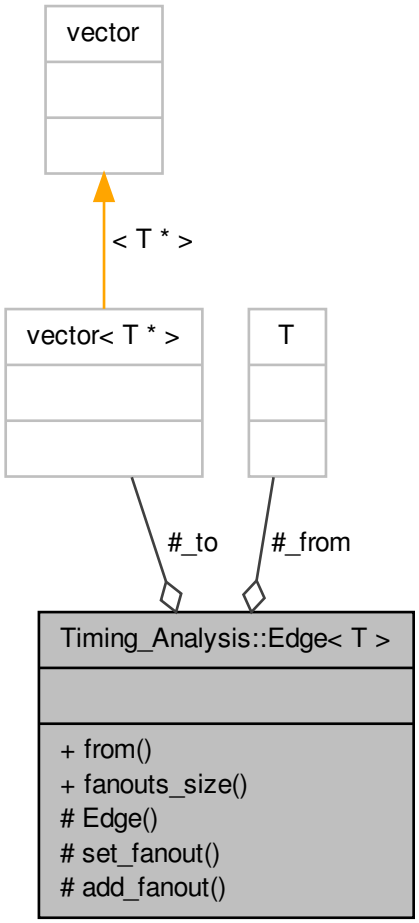
Template class which represents an edge of the graph model. An edge in graph theory is a connection between vertices.

```
#include <edge.h>
```

Inheritance diagram for Timing_Analysis::Edge< T >:



Collaboration diagram for Timing_Analysis::Edge< T >:



Public Member Functions

- T ∗ from () const

  Returns origin vertex.
- size_t fanouts_size () const

  Returns fan out.

Protected Member Functions

- Edge (T ∗from)

  Edge constructor.
- void set_fanout (const int i, T ∗tp)

  Inserts T pointer at index i.
- void add_fanout (T ∗tp)

  Inserts T pointer.

Protected Attributes

- T * _from
- vector< T * > _to

7.4.1 Detailed Description

template<typename T>class Timing_Analysis::Edge< T >

Template class which represents an edge of the graph model. An edge in graph theory is a connection between vertices.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 template<typename T> Timing_Analysis::Edge< T >::Edge (T * *from*) [inline], [protected]

Edge constructor.

Parameters

T	* from

7.4.3 Member Function Documentation

7.4.3.1 template<typename T> void Timing_Analysis::Edge< T >::add_fanout (T * *tp*) [inline], [protected]

Inserts T pointer.

Parameters

T	*tp

Returns

void

7.4.3.2 template<typename T> size_t Timing_Analysis::Edge< T >::fanouts_size ( ) const [inline]

Returns fan out.

Returns

size_t

7.4.3.3 template<typename T> T* Timing_Analysis::Edge< T >::from ( ) const [inline]

Returns origin vertex.

Returns

T *

7.4.3.4 template<typename T> void Timing_Analysis::Edge< T >::set_fanout (const int *i*, T * *tp*) [inline], [protected]

Inserts T pointer at index i.

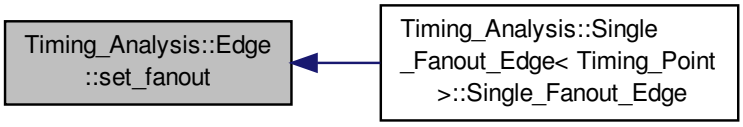
Parameters

const	int i, T *tp

Returns

void

Here is the caller graph for this function:



7.4.4 Member Data Documentation

7.4.4.1 template<typename T> T* Timing_Analysis::Edge< T >::_from [protected]

7.4.4.2 template<typename T> vector<T *> Timing_Analysis::Edge< T >::_to [protected]

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

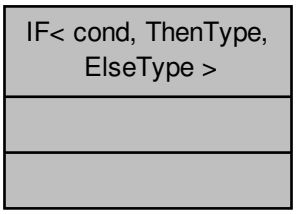
- TimingAnalysis/src/include/edge.h

7.5 IF< cond, ThenType, ElseType > Struct Template Reference

This is a Configuration File To switch between ISPD2012 (Lumped Capacitance Wire Delay Model) or ISPD2013 (Distributed RC Wire Delay Model) SPEF format.

```
#include <configuration.h>
```

Collaboration diagram for IF< cond, ThenType, ElseType >:



Public Types

- typedef ThenType RET

7.5.1 Detailed Description

template<bool cond, class ThenType, class ElseType>struct IF< cond, ThenType, ElseType >

This is a Configuration File To switch between ISPD2012 (Lumped Capacitance Wire Delay Model) or ISPD2013 (Distributed RC Wire Delay Model) SPEF format.

STATIC METAPROGRAMMED IFMetaprogrammed IF

7.5.2 Member Typedef Documentation

7.5.2.1 template<bool cond, class ThenType, class ElseType > typedef ThenType IF< cond, ThenType, ElseType >::RET

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

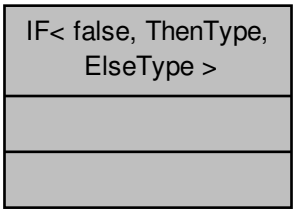
- TimingAnalysis/src/include/configuration.h

7.6 IF< false, ThenType, ElseType > Struct Template Reference

Metaprogrammed IF.

```
#include <configuration.h>
```

Collaboration diagram for IF< false, ThenType, ElseType >:



Public Types

- typedef ElseType RET

7.6.1 Detailed Description

template<class ThenType, class ElseType>struct IF< false, ThenType, ElseType >

Metaprogrammed IF.

7.6.2 Member Typedef Documentation

7.6.2.1 template<class ThenType , class ElseType > typedef ElseType IF< false, ThenType, ElseType >::RET

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

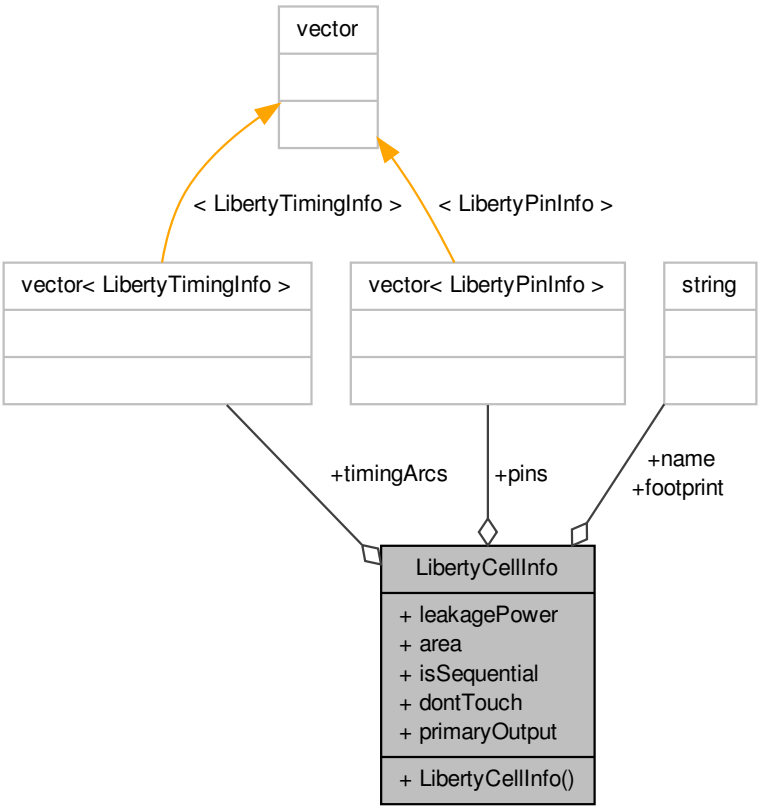
- TimingAnalysis/src/include/configuration.h

7.7 LibertyCellInfo Struct Reference

Struct which has informations about a cell.

`#include <liberty_library.h>`

Collaboration diagram for LibertyCellInfo:



Public Member Functions

- LibertyCellInfo ()

  LibertyCellInfo default constructor.

Public Attributes

- string name

- string footprint
- double leakagePower
- double area
- bool isSequential
- bool dontTouch
- bool primaryOutput
- vector< LibertyPinInfo > pins
- vector< LibertyTimingInfo > timingArcs

7.7.1 Detailed Description

Struct which has informations about a cell.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 LibertyCellInfo::LibertyCellInfo ( ) [inline]

LibertyCellInfo default constructor.

7.7.3 Member Data Documentation

7.7.3.1 double LibertyCellInfo::area

7.7.3.2 bool LibertyCellInfo::dontTouch

7.7.3.3 string LibertyCellInfo::footprint

7.7.3.4 bool LibertyCellInfo::isSequential

7.7.3.5 double LibertyCellInfo::leakagePower

7.7.3.6 string LibertyCellInfo::name

7.7.3.7 vector<LibertyPinInfo> LibertyCellInfo::pins

7.7.3.8 bool LibertyCellInfo::primaryOutput

7.7.3.9 vector<LibertyTimingInfo> LibertyCellInfo::timingArcs

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

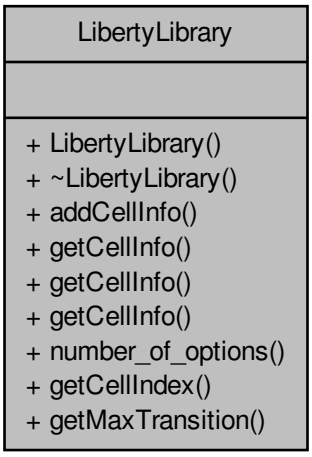
- TimingAnalysis/src/include/liberty_library.h

7.8 LibertyLibrary Class Reference

Standard cell Library is a collection of logic functions info. Contains details about implementations of logic functions, within its layout, parasitic models, Verilog models, and more. Generated by the LibertyParser class.

```
#include <liberty_library.h>
```

Collaboration diagram for LibertyLibrary:



Public Member Functions

- LibertyLibrary (const double maxTransition=0.0f)

  LibertyLibrary default constructor.
- virtual ~LibertyLibrary ()

  Empty LibertyLibrary desctructor.
- const pair< int, int > addCellInfo (const LibertyCellInfo &cellInfo)

  Returns pair<footprint index, option index>
- const LibertyCellInfo & getCellInfo (const string &footPrint, const int &i) const

  Returns LibertyCellInfo in footPrint, at index i.
- const LibertyCellInfo & getCellInfo (const string &cellName) const

  Returns LibertyCellInfo of cellName name.
- const LibertyCellInfo & getCellInfo (const int &footPrintIndex, const int &optionIndex) const

  Returns LibertyCellInfo at index i, in footPrint of footPrintIndex.
- size_t number_of_options (const int footprint_index) const
- const pair< int, int > getCellIndex (const string &cellName) const

  Returns pair<footprint index, option index>
- double getMaxTransition () const

  Returns maxTransition.

7.8.1 Detailed Description

Standard cell Library is a collection of logic functions info. Contains details about implementations of logic functions, within its layout, parasitic models, Verilog models, and more. Generated by the LibertyParser class.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 LibertyLibrary::LibertyLibrary (const double *maxTransition* = `0.0f`)

LibertyLibrary default constructor.

Parameters

const	double MaxTransition

7.8.2.2 virtual LibertyLibrary::~LibertyLibrary ( ) [virtual]

Empty LibertyLibrary desctructor.

7.8.3 Member Function Documentation

7.8.3.1 const pair<int, int> LibertyLibrary::addCellInfo (const LibertyCellInfo & *cellInfo*)

Returns pair<footprint index, option index>

Parameters

const	LibertyCellInfo & cellInfo

Returns

const pair<int, int>

7.8.3.2 const pair<int, int> LibertyLibrary::getCellIndex (const string & *cellName*) const

Returns pair<footprint index, option index>

Parameters

const	string &cellName

Returns

const pair<int, int>

7.8.3.3 const LibertyCellInfo& LibertyLibrary::getCellInfo (const string & *footPrint,* const int & *i*) const

Returns LibertyCellInfo in footPrint, at index i.

Parameters

const	string & footPrint, const int & i

Returns

const LibertyCellInfo &

7.8.3.4 const LibertyCellInfo& LibertyLibrary::getCellInfo (const string & *cellName*) const

Returns LibertyCellInfo of cellName name.

Parameters

const	string & cellName

Returns

const LibertyCellInfo &

7.8.3.5 const LibertyCellInfo& LibertyLibrary::getCellInfo (const int & *footPrintIndex,* const int & *optionIndex*) const

Returns LibertyCellInfo at index i, in footPrint of footPrintIndex.

Parameters

const	string & footPrintIndex, const int & optionIndex

Returns

const LibertyCellInfo &

7.8.3.6 double LibertyLibrary::getMaxTransition ( ) const

Returns maxTransition.

Returns

double

7.8.3.7 size_t LibertyLibrary::number_of_options (const int *footprint_index*) const

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

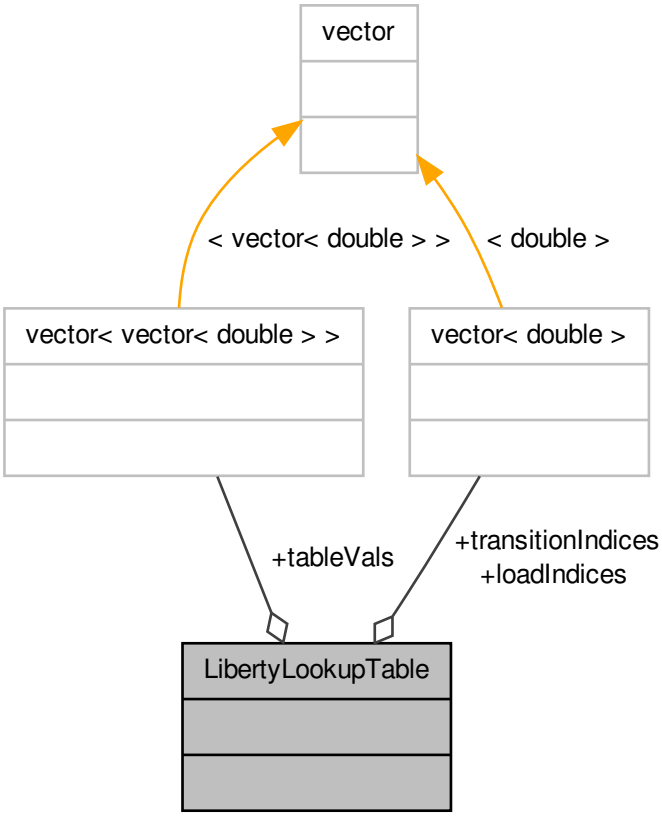
- TimingAnalysis/src/include/liberty_library.h

7.9 LibertyLookupTable Struct Reference

Struct representing a lookup table to store delay or slew functions.

```
#include <liberty_library.h>
```

Collaboration diagram for LibertyLookupTable:



Public Attributes

- vector< double > loadIndices
- vector< double > transitionIndices
- vector< vector< double > > tableVals

7.9.1 Detailed Description

Struct representing a lookup table to store delay or slew functions.

Look up table is indexed by the output load and the input transition values

Let L = loadIndices[i] T = transitionIndices[j] Then, the table value corresponding to L and T will be: table[i][j]

7.9.2 Member Data Documentation

7.9.2.1 vector<double> LibertyLookupTable::loadIndices

7.9.2.2 vector<vector<double> > LibertyLookupTable::tableVals

7.9.2.3 vector<double> LibertyLookupTable::transitionIndices

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

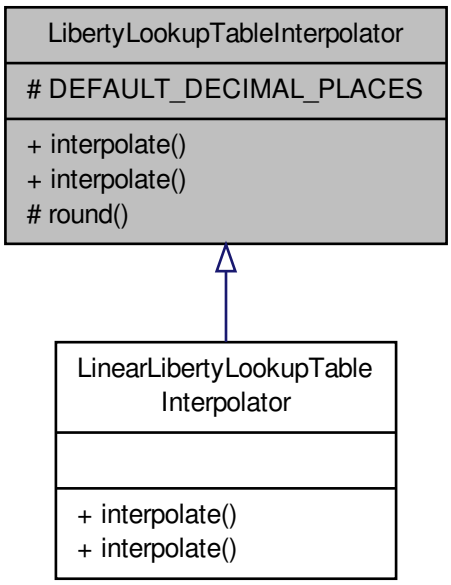
- TimingAnalysis/src/include/liberty_library.h

7.10 LibertyLookupTableInterpolator Class Reference

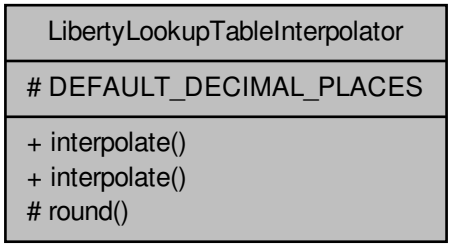
Interpolation calculator.

```
#include <liberty_library.h>
```

Inheritance diagram for LibertyLookupTableInterpolator:



Collaboration diagram for LibertyLookupTableInterpolator:



Public Member Functions

- virtual double interpolate (const LibertyLookupTable &lut, const double load, const double transition)=0

  Returns interpolated value.
- virtual const Transitions< double > interpolate (const LibertyLookupTable &riseLut, const LibertyLookupTable &fallLut, const Transitions< double > load, const Transitions< double > transition, Unateness unateness=NEGATIVE_UNATE, bool is_input_driver=false)=0

  Returns Transitions<double, double> with its rise and fall delay values interpolated.

Protected Member Functions

- void round (Transitions< double > &transitions, const int decimal_places)

  Truncates Transitions<double> to Transitions<double> . Not implemented.

Static Protected Attributes

- static const int DEFAULT_DECIMAL_PLACES

7.10.1 Detailed Description

Interpolation calculator.

7.10.2 Member Function Documentation

7.10.2.1 virtual double LibertyLookupTableInterpolator::interpolate (const LibertyLookupTable & *lut*, const double *load*, const double *transition*) [pure virtual]

Returns interpolated value.

Parameters

const	LibertyLookupTable & lut, const double load, const double transition

Returns

double

Implemented in LinearLibertyLookupTableInterpolator.

7.10.2.2 virtual const Transitions<double> LibertyLookupTableInterpolator::interpolate (const LibertyLookupTable & *riseLut*, const LibertyLookupTable & *fallLut*, const Transitions< double > *load*, const Transitions< double > *transition*, Unateness *unateness* = NEGATIVE_UNATE, bool *is_input_driver* = false) [pure virtual]

Returns Transitions<double, double> with its rise and fall delay values interpolated.

Parameters

const	LibertyLookupTable & riseLut, const LibertyLookupTable & fallLut, const Transitions<double> load, const Transitions<double> transition, Unateness unateness(default NEGATIVE_UNATE), bool is_input_driver(default false)

Returns

const Transitions<double>

Implemented in LinearLibertyLookupTableInterpolator.

7.10.2.3 void LibertyLookupTableInterpolator::round (Transitions< double > & *transitions*, const int *decimal_places*) [protected]

Truncates Transitions<double> to Transitions<double> . Not implemented.

Parameters

Transitions<double>	& transitions, const int decimal_places

Returns

void

7.10.3 Member Data Documentation

7.10.3.1 const int LibertyLookupTableInterpolator::DEFAULT_DECIMAL_PLACES [static], [protected]

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

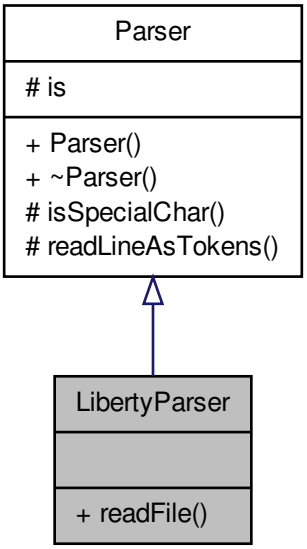
- TimingAnalysis/src/include/liberty_library.h

7.11 LibertyParser Class Reference

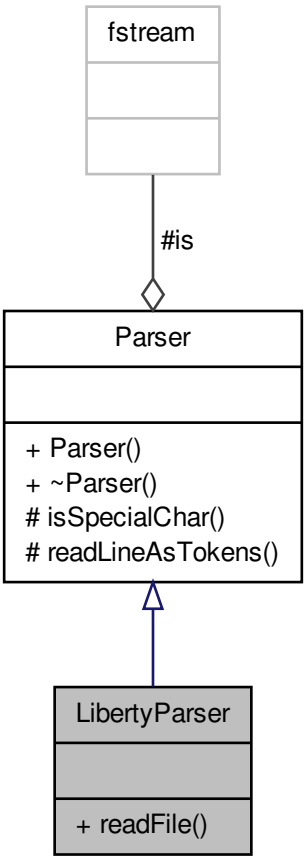
Inherits from Parser. Used to generate a LibertyLibrary object from a file.

```
#include <parser.h>
```

Inheritance diagram for LibertyParser:



Collaboration diagram for LibertyParser:



Public Member Functions

- const LibertyLibrary readFile (const string filename)

  Returns LibertyLibrary generated using values read from file passed as filename parameter.

Additional Inherited Members

7.11.1 Detailed Description

Inherits from Parser. Used to generate a LibertyLibrary object from a file.

Remarks

See test_lib_parser () function in parser_helper.cpp for an example of how to use this class.

7.11.2 Member Function Documentation

7.11.2.1 const LibertyLibrary LibertyParser::readFile (const string *filename*)

Returns LibertyLibrary generated using values read from file passed as filename parameter.

Parameters

const	string filename

Returns

LibertyLivrary

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

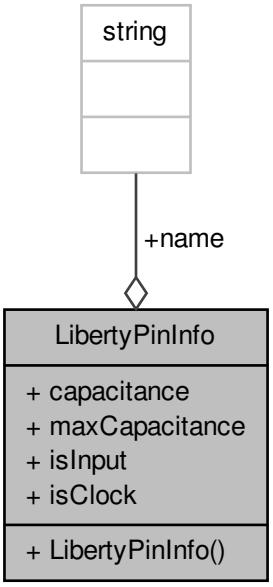
- TimingAnalysis/src/include/parser.h

7.12 LibertyPinInfo Struct Reference

Struct which has informations about a pin.

`#include <liberty_library.h>`

Collaboration diagram for LibertyPinInfo:



Public Member Functions

- LibertyPinInfo ()

  LibertyPinInfo default constructor.

Public Attributes

- string name
- double capacitance

- double maxCapacitance
- bool isInput
- bool isClock

7.12.1 Detailed Description

Struct which has informations about a pin.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 LibertyPinInfo::LibertyPinInfo ( ) [inline]

LibertyPinInfo default constructor.

7.12.3 Member Data Documentation

7.12.3.1 double LibertyPinInfo::capacitance

7.12.3.2 bool LibertyPinInfo::isClock

7.12.3.3 bool LibertyPinInfo::isInput

7.12.3.4 double LibertyPinInfo::maxCapacitance

7.12.3.5 string LibertyPinInfo::name

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

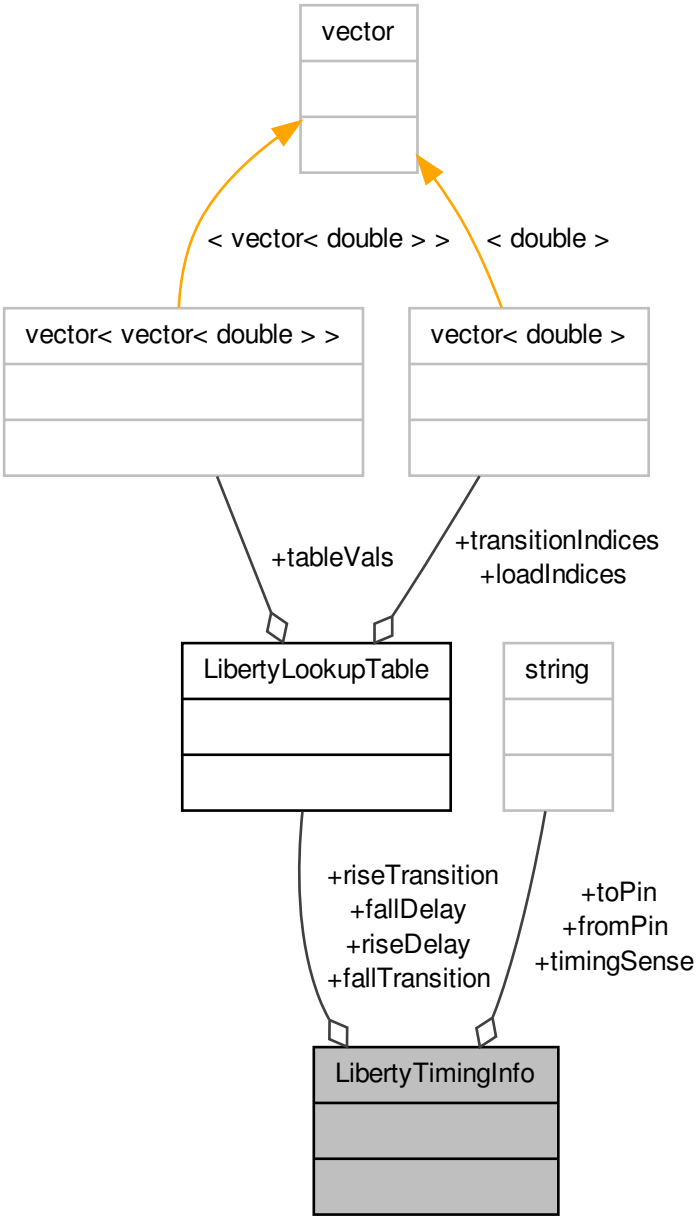
- TimingAnalysis/src/include/liberty_library.h

7.13 LibertyTimingInfo Struct Reference

Struct which has informations about timing.

```
#include <liberty_library.h>
```

Collaboration diagram for LibertyTimingInfo:



Public Attributes

- string fromPin
- string toPin
- string timingSense
- timing model for ISPD
  LibertyLookupTable fallDelay
- LibertyLookupTable riseDelay
- LibertyLookupTable fallTransition

- LibertyLookupTable riseTransition

7.13.1 Detailed Description

Struct which has informations about timing.

"non_unate" or "negative_unate" or "positive_unate". Note that ISPD-13 library will have only negative-unate combinational cells. The clock arcs for sequentials will be non_unate (which can be ignored because of the simplified sequential

7.13.2 Member Data Documentation

7.13.2.1 timing model for ISPD LibertyLookupTable LibertyTimingInfo::fallDelay

7.13.2.2 LibertyLookupTable LibertyTimingInfo::fallTransition

7.13.2.3 string LibertyTimingInfo::fromPin

7.13.2.4 LibertyLookupTable LibertyTimingInfo::riseDelay

7.13.2.5 LibertyLookupTable LibertyTimingInfo::riseTransition

7.13.2.6 string LibertyTimingInfo::timingSense

7.13.2.7 string LibertyTimingInfo::toPin

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

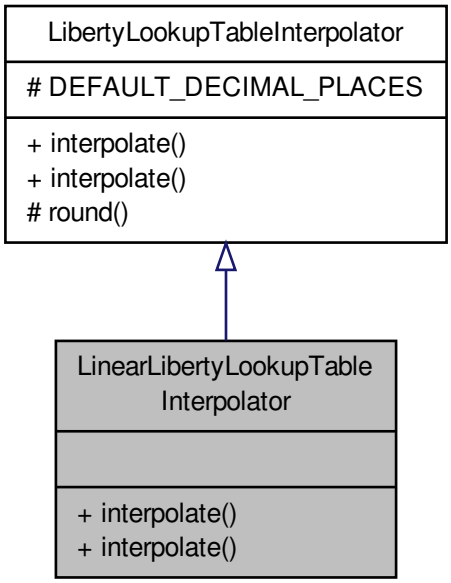
- TimingAnalysis/src/include/liberty_library.h

7.14 LinearLibertyLookupTableInterpolator Class Reference

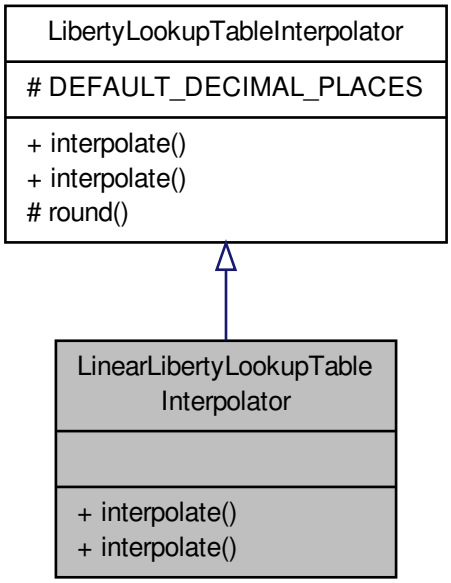
Linear interpolation calculator.

```
#include <liberty_library.h>
```

Inheritance diagram for LinearLibertyLookupTableInterpolator:



Collaboration diagram for LinearLibertyLookupTableInterpolator:



Public Member Functions

- double interpolate (const LibertyLookupTable &lut, const double load, const double transition)

  Returns interpolated value.
- const Transitions< double > interpolate (const LibertyLookupTable &riseLut, const LibertyLookupTable &fallLut, const Transitions< double > load, const Transitions< double > transition, Unateness unateness=NEGATIVE_UNATE, bool is_input_driver=false)

  Returns Transitions<double, double> with its rise and fall delay values linearly interpolated.

Additional Inherited Members

7.14.1 Detailed Description

Linear interpolation calculator.

7.14.2 Member Function Documentation

7.14.2.1 double LinearLibertyLookupTableInterpolator::interpolate (const LibertyLookupTable & *lut*, const double *load*, const double *transition*) [virtual]

Returns interpolated value.

Parameters

const	LibertyLookupTable & lut, const double load, const double transition

Returns

double

Implements LibertyLookupTableInterpolator.

7.14.2.2 const Transitions<double> LinearLibertyLookupTableInterpolator::interpolate (const LibertyLookupTable & *riseLut*, const LibertyLookupTable & *fallLut*, const Transitions< double > *load*, const Transitions< double > *transition*, Unateness *unateness* = NEGATIVE_UNATE, bool *is_input_driver* = false) [virtual]

Returns Transitions<double, double> with its rise and fall delay values linearly interpolated.

Parameters

const	LibertyLookupTable & riseLut, const LibertyLookupTable & fallLut, const Transitions<double> load, const Transitions<double> transition, Unateness unateness(default NEGATIVE_UNATE), bool is_input_driver(default false)

Returns

const Transitions<double>

Implements LibertyLookupTableInterpolator.

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

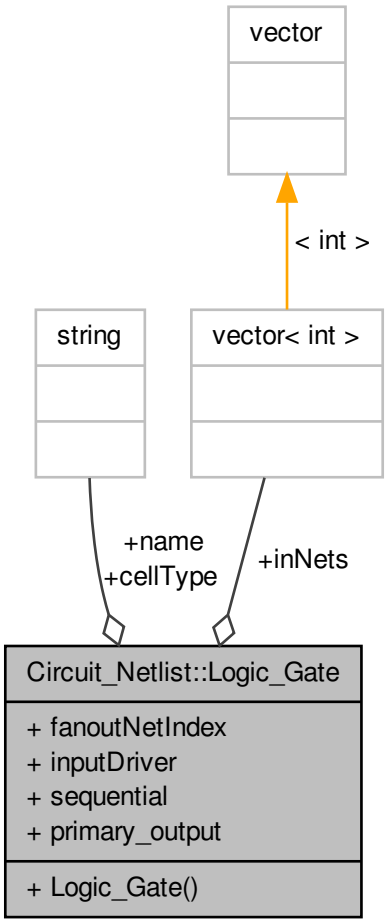
- TimingAnalysis/src/include/liberty_library.h

7.15 Circuit_Netlist::Logic_Gate Struct Reference

Struct which represents a Logic_Gate.

```
#include <circuit_netlist.h>
```

Collaboration diagram for Circuit_Netlist::Logic_Gate:



Public Member Functions

- Logic_Gate (const string name, const string cellType, const unsigned inputs, int fanoutNetIndex, const bool inputDriver=false, const bool primary_output=false)

  Logic_Gate constructor.

Public Attributes

- string name
- string cellType
- vector< int > inNets
- int fanoutNetIndex

- bool inputDriver
- bool sequential
- bool primary_output

7.15.1 Detailed Description

Struct which represents a Logic_Gate.

7.15.2 Constructor & Destructor Documentation

7.15.2.1 Circuit_Netlist::Logic_Gate::Logic_Gate (const string *name*, const string *cellType*, const unsigned *inputs*, int *fanoutNetIndex*, const bool *inputDriver* = `false`, const bool *primary_output* = `false`) `[inline]`

Logic_Gate constructor.

Parameters

const	string name, const string cellType, const unsigned inputs, int fanoutNetIndex, const bool inputDriver(false default), const bool primary_output(false default)

7.15.3 Member Data Documentation

7.15.3.1 string Circuit_Netlist::Logic_Gate::cellType

7.15.3.2 int Circuit_Netlist::Logic_Gate::fanoutNetIndex

7.15.3.3 vector<int> Circuit_Netlist::Logic_Gate::inNets

7.15.3.4 bool Circuit_Netlist::Logic_Gate::inputDriver

7.15.3.5 string Circuit_Netlist::Logic_Gate::name

7.15.3.6 bool Circuit_Netlist::Logic_Gate::primary_output

7.15.3.7 bool Circuit_Netlist::Logic_Gate::sequential

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

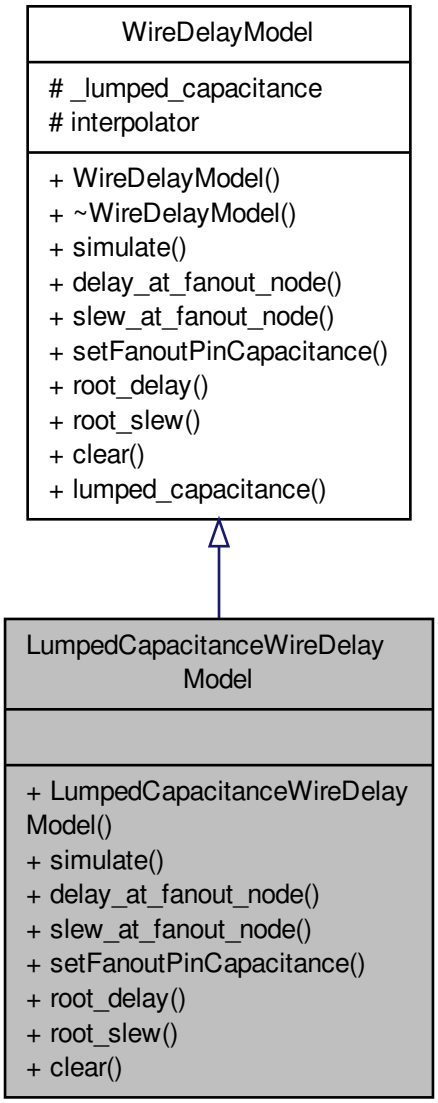
- TimingAnalysis/src/include/circuit_netlist.h

7.16 LumpedCapacitanceWireDelayModel Class Reference

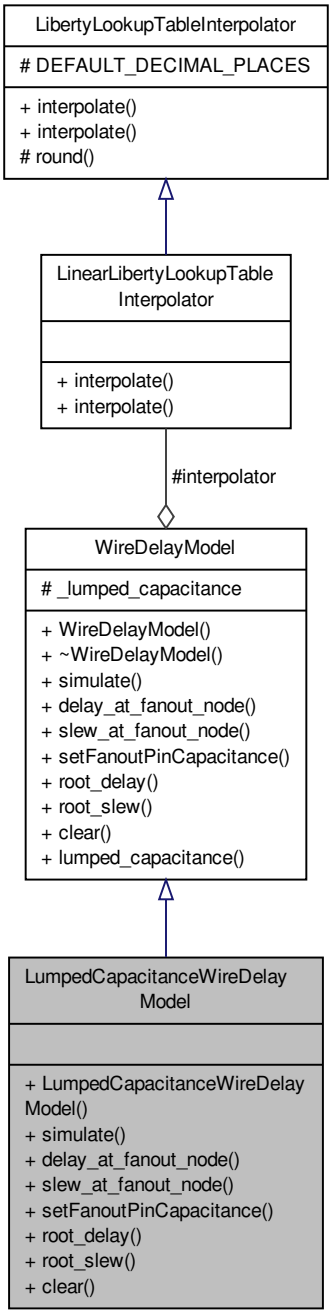
This model describes the parasitic capacitances distributed along a wire as a single circuit element. Does not take resistors into account.

```
#include <wire_delay_model.h>
```

Inheritance diagram for LumpedCapacitanceWireDelayModel:



Collaboration diagram for LumpedCapacitanceWireDelayModel:



Public Member Functions

- LumpedCapacitanceWireDelayModel (const SpefNet &descriptor, const string root_node, const bool dummy_edge=false)

  LumpedCapacitanceWireDelayModel default constructor.

- const Transitions< double > simulate (const LibertyCellInfo &cellInfo, const int input, const Transitions< double > _slew, bool is_input_driver)

Updates delay and slew times and returns Transitions with lumped capacitance values.

- const Transitions< double > delay_at_fanout_node (const string fanout_node_name) const

  Returns Transitions with rise and fall delay time values set to zero.
- const Transitions< double > slew_at_fanout_node (const string fanout_node_name) const

  Returns Transitions with rise and fall slew time values set to zero.
- void setFanoutPinCapacitance (const string fanout_name_and_pin, const double pinCapacitance)

  Increments total lumped capacitance with pinCapacitance.
- Transitions< double > root_delay (int arc_number)

  Returns delay time value.
- Transitions< double > root_slew (int arc_number)

  Returns slew time value.
- void clear ()

  Not implemented. Included for completeness.

Additional Inherited Members

7.16.1 Detailed Description

This model describes the parasitic capacitances distributed along a wire as a single circuit element. Does not take resistors into account.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 LumpedCapacitanceWireDelayModel::LumpedCapacitanceWireDelayModel (const SpefNet & *descriptor,* const string *root_node,* const bool *dummy_edge* = `false`) `[inline]`

LumpedCapacitanceWireDelayModel default constructor.

Parameters

const	SpefNet & descriptor, const string root_node, const bool dummy_edge(default false)

7.16.3 Member Function Documentation

7.16.3.1 void LumpedCapacitanceWireDelayModel::clear ( ) `[virtual]`

Not implemented. Included for completeness.

Returns

void

Implements WireDelayModel.

7.16.3.2 const Transitions<double> LumpedCapacitanceWireDelayModel::delay_at_fanout_node (const string *fanout_node_name*) const `[virtual]`

Returns Transitions with rise and fall delay time values set to zero.

Parameters

const	string fanout_node_name

Returns

const Transitions<double>

Implements WireDelayModel.

7.16.3.3 Transitions<double> LumpedCapacitanceWireDelayModel::root_delay (int *arc_number*) [virtual]

Returns delay time value.

Parameters

int	arc_number

Returns

Transitions<double>

Implements WireDelayModel.

7.16.3.4 Transitions<double> LumpedCapacitanceWireDelayModel::root_slew (int *arc_number*) [virtual]

Returns slew time value.

Parameters

int	arc_number

Returns

Transitions<double>

Implements WireDelayModel.

7.16.3.5 void LumpedCapacitanceWireDelayModel::setFanoutPinCapacitance (const string *fanout_name_and_pin,* const double *pinCapacitance*) [inline],[virtual]

Increments total lumped capacitance with pinCapacitance.

Parameters

const	string fanout_name_and_pin, const double pinCapacitance

Returns

void

Implements WireDelayModel.

7.16.3.6 const Transitions<double> LumpedCapacitanceWireDelayModel::simulate (const LibertyCellInfo & *cellInfo,* const int *input,* const Transitions< double > *_slew,* bool *is_input_driver*) [virtual]

Updates delay and slew times and returns Transitions with lumped capacitance values.

Parameters

const	LibertyCellInfo & cellInfo, const int input, const Transitions<double> _slew, bool is_input_driver

Returns

const Transitions<double>

Implements WireDelayModel.

7.16.3.7 const Transitions<double> LumpedCapacitanceWireDelayModel::slew_at_fanout_node (const string *fanout_node_name*) const [virtual]

Returns Transitions with rise and fall slew time values set to zero.

Parameters

const	string fanout_node_name

Returns

const Transitions<double>

Implements WireDelayModel.

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

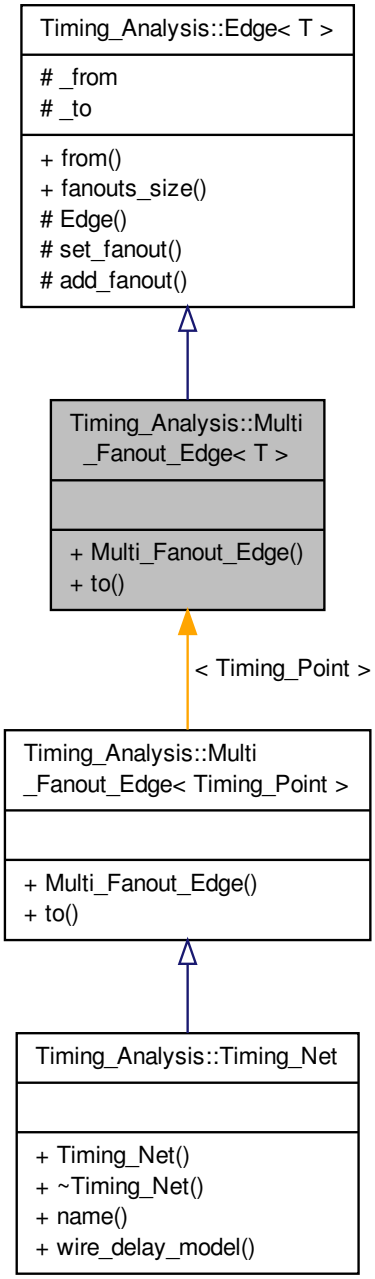
- TimingAnalysis/src/include/wire_delay_model.h

7.17 Timing_Analysis::Multi_Fanout_Edge< T > Class Template Reference

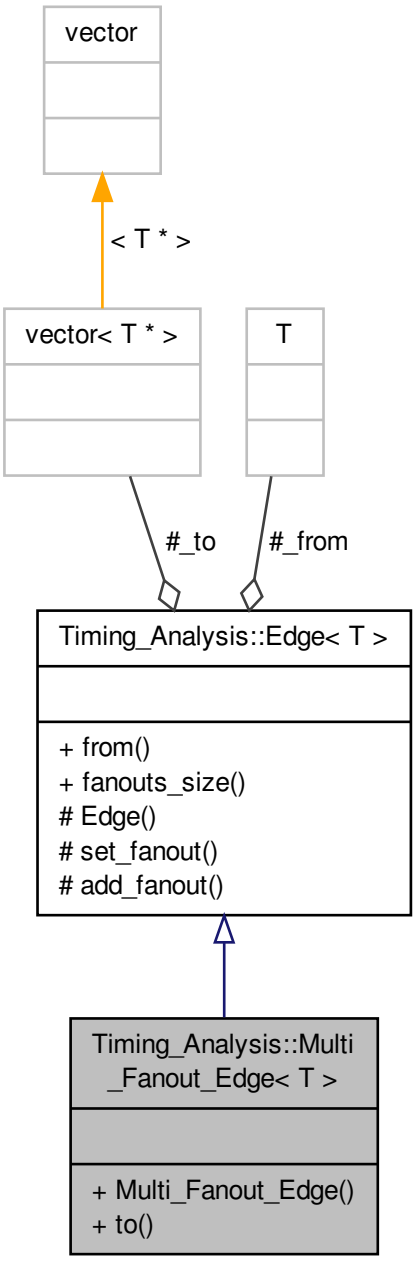
Inherits from Edge. An multi fanout edge is a group of edges which connects a vertex to more than one vertexes.

```
#include <multi_fanout_edge.h>
```

Inheritance diagram for Timing_Analysis::Multi_Fanout_Edge< T >:



Collaboration diagram for Timing_Analysis::Multi_Fanout_Edge< T >:



Public Member Functions

- Multi_Fanout_Edge (T *from)

  Multi_Fanout_Edge constructor.

- T & to (const int i) const

  Returns Edge<T> at index i.

Additional Inherited Members

7.17.1 Detailed Description

template<class T>class Timing_Analysis::Multi_Fanout_Edge< T >

Inherits from Edge. An multi fanout edge is a group of edges which connects a vertex to more than one vertexes.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 template<class T> Timing_Analysis::Multi_Fanout_Edge< T >::Multi_Fanout_Edge (T * *from*) [inline]

Multi_Fanout_Edge constructor.

7.17.3 Member Function Documentation

7.17.3.1 template<class T> T& Timing_Analysis::Multi_Fanout_Edge< T >::to (const int *i*) const [inline]

Returns Edge<T> at index i.

Parameters

const	int i

Returns

T &

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

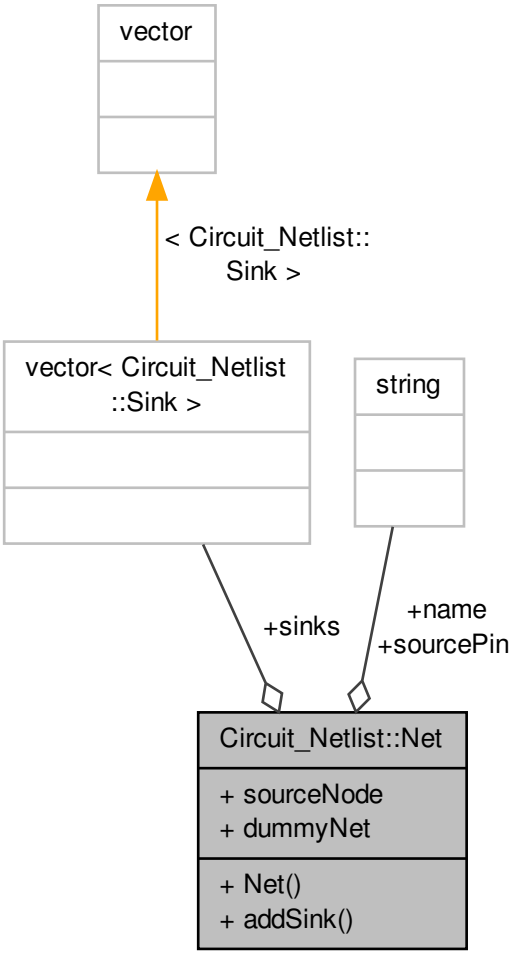
- TimingAnalysis/src/include/multi_fanout_edge.h

7.18 Circuit_Netlist::Net Struct Reference

Struct which represents a Net, which connect two or more circuit elements together.

```
#include <circuit_netlist.h>
```

Collaboration diagram for Circuit_Netlist::Net:



Public Member Functions

- Net (const string name, const int sourceNode, const string sourcePin, const bool dummyNet=false)

  Net constructor.
- void addSink (const Sink sinkNode)

  Adds sink to sinks list.

Public Attributes

- string name
- int sourceNode
- string sourcePin
- vector< Sink > sinks
- bool dummyNet

7.18.1 Detailed Description

Struct which represents a Net, which connect two or more circuit elements together.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Circuit_Netlist::Net::Net (const string *name,* const int *sourceNode,* const string *sourcePin,* const bool *dummyNet* = `false`) `[inline]`

Net constructor.

Parameters

const	string name, const int sourceNode, const string sourcePin, const bool dummyNet(false default)

7.18.3 Member Function Documentation

7.18.3.1 void Circuit_Netlist::Net::addSink (const Sink *sinkNode*) `[inline]`

Adds sink to sinks list.

Parameters

const	Sink sinkNode

Returns

void

7.18.4 Member Data Documentation

7.18.4.1 bool Circuit_Netlist::Net::dummyNet

7.18.4.2 string Circuit_Netlist::Net::name

7.18.4.3 vector<Sink> Circuit_Netlist::Net::sinks

7.18.4.4 int Circuit_Netlist::Net::sourceNode

7.18.4.5 string Circuit_Netlist::Net::sourcePin

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

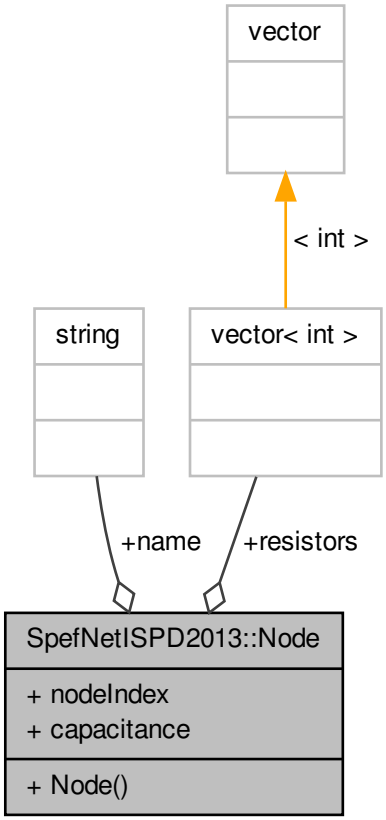
- TimingAnalysis/src/include/circuit_netlist.h

7.19 SpefNetlSPD2013::Node Struct Reference

Struct which represents a node of the circuit. A node is a point of the circuit where two or more elements meet.

`#include <spef_net.h>`

Collaboration diagram for SpefNetISPD2013::Node:



Public Member Functions

- Node (const int &index, const string &name)

  Node constructor.

Public Attributes

- int nodeIndex
- string name
- vector< int > resistors
- double capacitance

7.19.1 Detailed Description

Struct which represents a node of the circuit. A node is a point of the circuit where two or more elements meet.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 SpefNetlSPD2013::Node::Node (const int & *index*, const string & *name*) [inline]

Node constructor.

Parameters

const	int & index, const string & name

7.19.3 Member Data Documentation

7.19.3.1 double SpefNetISPD2013::Node::capacitance

7.19.3.2 string SpefNetISPD2013::Node::name

7.19.3.3 int SpefNetISPD2013::Node::nodeIndex

7.19.3.4 vector<int> SpefNetISPD2013::Node::resistors

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

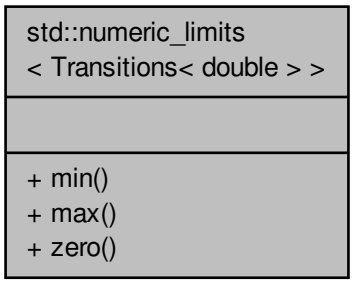
- TimingAnalysis/src/include/spef_net.h

7.20 std::numeric_limits< Transitions< double > > Class Template Reference

Class used to make easier to work with double variable numeric limits.

```
#include <transitions.h>
```

Collaboration diagram for std::numeric_limits< Transitions< double > >:



Static Public Member Functions

- static Transitions< double > min ()

  Sets Returns a Transition with the minimum values of rise and fall times possible in IEEE floating point double precision representation.
- static Transitions< double > max ()

  Sets Returns a Transition with the maximum values of rise and fall times possible in IEEE floating point double precision representation.
- static Transitions< double > zero ()

  Sets Returns a Transition with value zero for rise and fall times.

7.20.1 Detailed Description

template<>class std::numeric_limits< Transitions< double > >

Class used to make easier to work with double variable numeric limits.

7.20.2 Member Function Documentation

7.20.2.1 static Transitions<double> std::numeric_limits< Transitions< double > >::max ( ) [inline], [static]

Sets Returns a Transition with the maximum values of rise and fall times possible in IEEE floating point double precision representation.

Returns

Transitions<double>

7.20.2.2 static Transitions<double> std::numeric_limits< Transitions< double > >::min ( ) [inline], [static]

Sets Returns a Transition with the minimum values of rise and fall times possible in IEEE floating point double precision representation.

Returns

Transitions<double>

7.20.2.3 static Transitions<double> std::numeric_limits< Transitions< double > >::zero ( ) [inline], [static]

Sets Returns a Transition with value zero for rise and fall times.

Returns

Transitions<double>

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

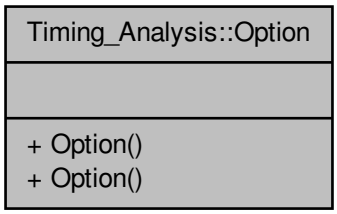
- TimingAnalysis/src/include/transitions.h

7.21 Timing_Analysis::Option Class Reference

Refers to the implementation option of a logic gate.

```
#include <timing_analysis.h>
```

Collaboration diagram for Timing_Analysis::Option:



Public Member Functions

- Option ()

  Option default constructor.
- Option (const int footprintIndex, const int optionIndex)

  Option default constructor.

Friends

- class Timing_Analysis

7.21.1 Detailed Description

Refers to the implementation option of a logic gate.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 Timing_Analysis::Option::Option ( ) [inline]

Option default constructor.

7.21.2.2 Timing_Analysis::Option::Option (const int *footprintIndex*, const int *optionIndex*) [inline]

Option default constructor.

Parameters

const	int footprintIndex, const int optionIndex

7.21.3 Friends And Related Function Documentation

7.21.3.1 friend class Timing_Analysis [friend]

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

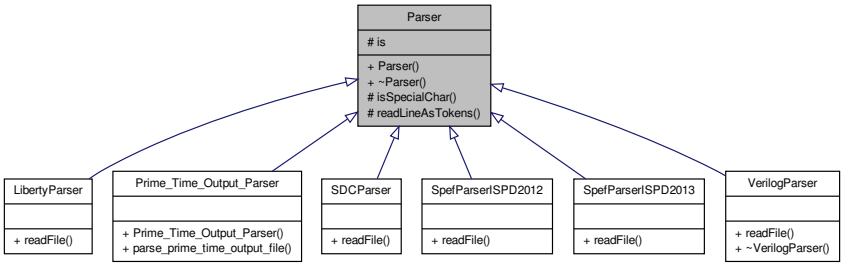
- TimingAnalysis/src/include/timing_analysis.h

7.22 Parser Class Reference

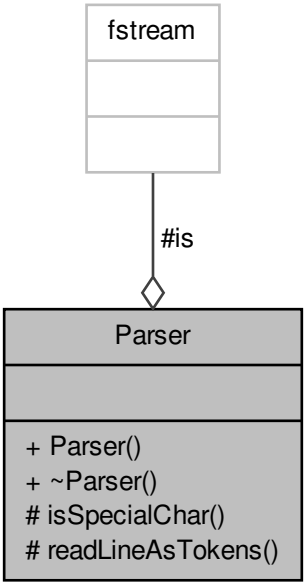
Superclass. A Parser is a program which reads an input and creates a data structure. In this case, the Parser will read a string, validate it and build essential files for use in the program using information read from the string.

`#include <parser.h>`

Inheritance diagram for Parser:



Collaboration diagram for Parser:



Public Member Functions

- Parser ()

  Empty Parser constructor.
- virtual ~Parser ()

  Empty Parser destructor.

Protected Member Functions

- bool isSpecialChar (const char &c)
- bool readLineAsTokens (istream &is, vector< string > &tokens, bool includeSpecialChars=false)

Protected Attributes

- fstream is

7.22.1 Detailed Description

Superclass. A Parser is a program which reads an input and creates a data structure. In this case, the Parser will read a string, validate it and build essential files for use in the program using information read from the string.

7.22.2 Constructor & Destructor Documentation

7.22.2.1 Parser::Parser ( )

Empty Parser constructor.

7.22.2.2 virtual Parser::~Parser ( ) [virtual]

Empty Parser destructor.

7.22.3 Member Function Documentation

7.22.3.1 bool Parser::isSpecialChar (const char & *c*) [protected]

7.22.3.2 bool Parser::readLineAsTokens (istream & *is*, vector< string > & *tokens*, bool *includeSpecialChars* = false) [protected]

7.22.4 Member Data Documentation

7.22.4.1 fstream Parser::is [protected]

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

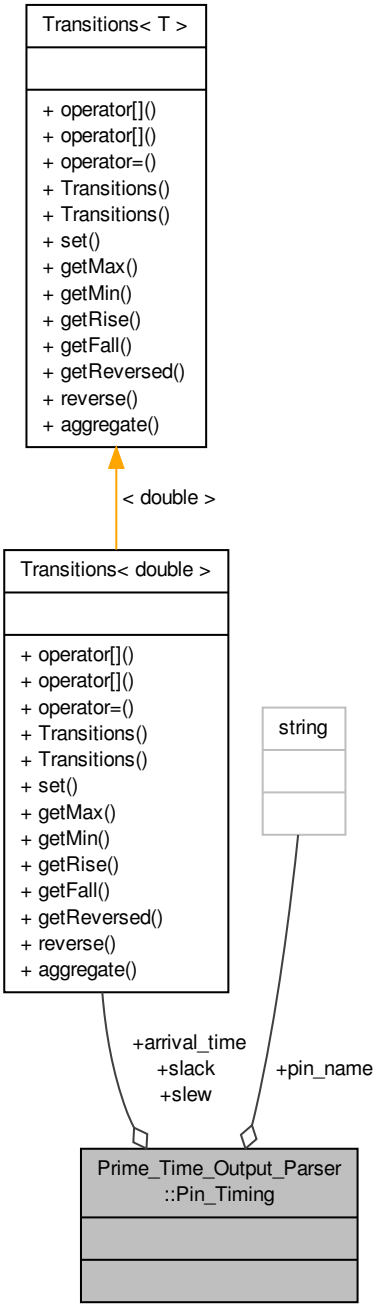
- TimingAnalysis/src/include/parser.h

7.23 Prime_Time_Output_Parser::Pin_Timing Struct Reference

Struct containing the timing values of a pin.

```
#include <parser.h>
```

Collaboration diagram for Prime_Time_Output_Parser::Pin_Timing:



Public Attributes

- string pin_name
- Transitions< double > slack
- Transitions< double > slew
- Transitions< double > arrival_time

Friends

- ostream & operator<< (ostream &out, const Pin_Timing &pin)

  Redefinition of << operator. Inserts formatted description including pin name and slack, slew and arrival time values.

7.23.1 Detailed Description

Struct containing the timing values of a pin.

7.23.2 Friends And Related Function Documentation

7.23.2.1 ostream& operator<< (ostream & *out*, const Pin_Timing & *pin*) [friend]

Redefinition of << operator. Inserts formatted description including pin name and slack, slew and arrival time values.

7.23.3 Member Data Documentation

7.23.3.1 Transitions<double> Prime_Time_Output_Parser::Pin_Timing::arrival_time

7.23.3.2 string Prime_Time_Output_Parser::Pin_Timing::pin_name

7.23.3.3 Transitions<double> Prime_Time_Output_Parser::Pin_Timing::slack

7.23.3.4 Transitions<double> Prime_Time_Output_Parser::Pin_Timing::slew

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

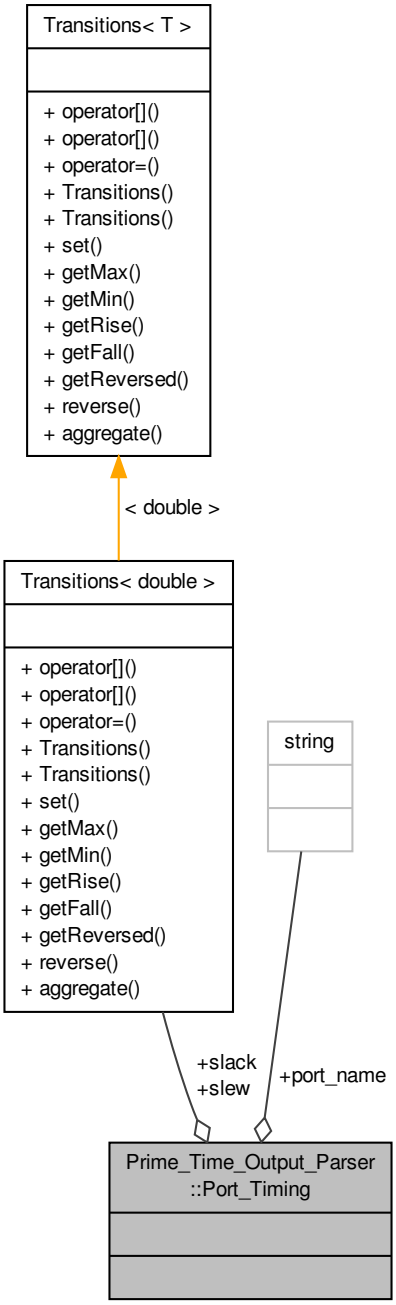
- TimingAnalysis/src/include/parser.h

7.24 Prime_Time_Output_Parser::Port_Timing Struct Reference

Struct containing the timing values of a port.

```
#include <parser.h>
```

Collaboration diagram for Prime_Time_Output_Parser::Port_Timing:



Public Attributes

- string port_name
- Transitions< double > slack
- Transitions< double > slew

Friends

- ostream & operator<< (ostream &out, const Port_Timing &port)

  Redefinition of << operator. Inserts formatted description including port name, slack and slew time values.

7.24.1 Detailed Description

Struct containing the timing values of a port.

7.24.2 Friends And Related Function Documentation

7.24.2.1 ostream& operator<< (ostream & *out*, const Port_Timing & *port*) [friend]

Redefinition of << operator. Inserts formatted description including port name, slack and slew time values.

7.24.3 Member Data Documentation

7.24.3.1 string Prime_Time_Output_Parser::Port_Timing::port_name

7.24.3.2 Transitions<double> Prime_Time_Output_Parser::Port_Timing::slack

7.24.3.3 Transitions<double> Prime_Time_Output_Parser::Port_Timing::slew

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

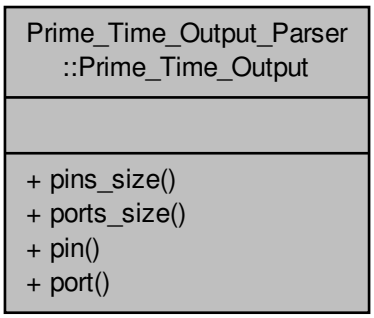
- TimingAnalysis/src/include/parser.h

7.25 Prime_Time_Output_Parser::Prime_Time_Output Class Reference

Represents Prime Time output. Provides access methods to it.

```
#include <parser.h>
```

Collaboration diagram for Prime_Time_Output_Parser::Prime_Time_Output:



Public Member Functions

- size_t pins_size () const

  Returns number of pins.
- size_t ports_size () const

  Returns number of ports.
- const Pin_Timing & pin (const size_t i) const

  Returns Pin_Timing reference at index i.
- const Port_Timing & port (const size_t i) const

  Returns Pin_Timing reference at index i.

Friends

- class Prime_Time_Output_Parser

7.25.1 Detailed Description

Represents Prime Time output. Provides access methods to it.

7.25.2 Member Function Documentation

7.25.2.1 const Pin_Timing& Prime_Time_Output_Parser::Prime_Time_Output::pin (const size_t *i*) const [inline]

Returns Pin_Timing reference at index i.

Parameters

const	size_t i

Returns

Pin_Timing &

7.25.2.2 size_t Prime_Time_Output_Parser::Prime_Time_Output::pins_size ( ) const [inline]

Returns number of pins.

Returns

size_t

7.25.2.3 const Port_Timing& Prime_Time_Output_Parser::Prime_Time_Output::port (const size_t *i*) const [inline]

Returns Pin_Timing reference at index i.

Parameters

const	size_t i

Returns

Port_Timing &

7.25.2.4 size_t Prime_Time_Output_Parser::Prime_Time_Output::ports_size ( ) const [inline]

Returns number of ports.

Returns

size_t

7.25.3 Friends And Related Function Documentation

7.25.3.1 friend class Prime_Time_Output_Parser [friend]

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

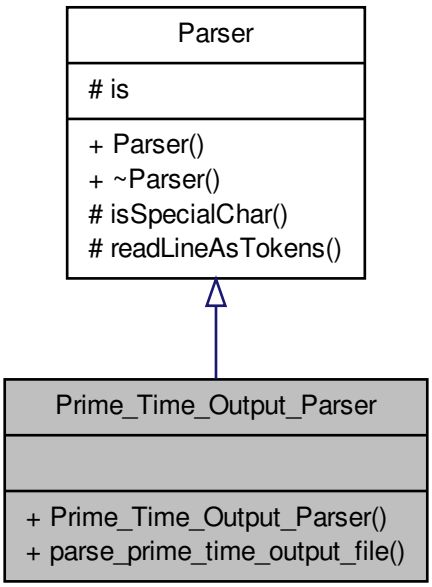
- TimingAnalysis/src/include/parser.h

7.26 Prime_Time_Output_Parser Class Reference

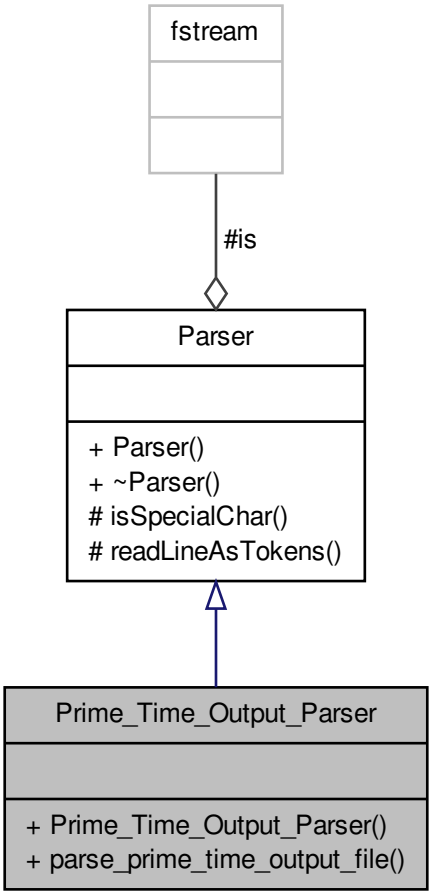
Inherits from Parser. Parse the Prime Time output. Used to generate a Prime_Time_Output object for later use.

`#include <parser.h>`

Inheritance diagram for Prime_Time_Output_Parser:



Collaboration diagram for Prime_Time_Output_Parser:



Classes

- struct Pin_Timing

  Struct containing the timing values of a pin.
- struct Port_Timing

  Struct containing the timing values of a port.
- class Prime_Time_Output

  Represents Prime Time output. Provides access methods to it.

Public Member Functions

- Prime_Time_Output_Parser ()

  Prime_Time_Output_Parser empty constructor.
- const Prime_Time_Output parse_prime_time_output_file (const string filename)

  Returns Prime_Time_Output object generated using values read from file passed as filename parameter.

Additional Inherited Members

7.26.1 Detailed Description

Inherits from Parser. Parse the Prime Time output. Used to generate a Prime_Time_Output object for later use.

7.26.2 Constructor & Destructor Documentation

7.26.2.1 Prime_Time_Output_Parser::Prime_Time_Output_Parser ( ) [inline]

Prime_Time_Output_Parser empty constructor.

7.26.3 Member Function Documentation

7.26.3.1 const Prime_Time_Output Prime_Time_Output_Parser::parse_prime_time_output_file (const string *filename*)

Returns Prime_Time_Output object generated using values read from file passed as filename parameter.

Parameters

const	string filename

Returns

const Prime_Time_Output

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

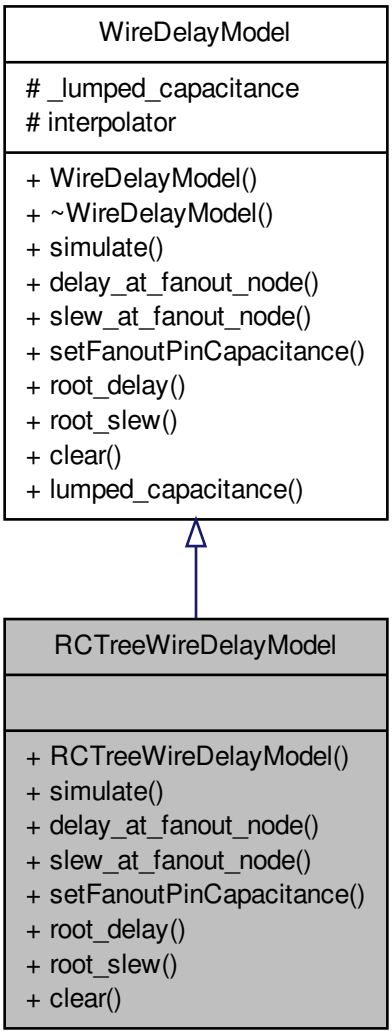
- TimingAnalysis/src/include/parser.h

7.27 RCTreeWireDelayModel Class Reference

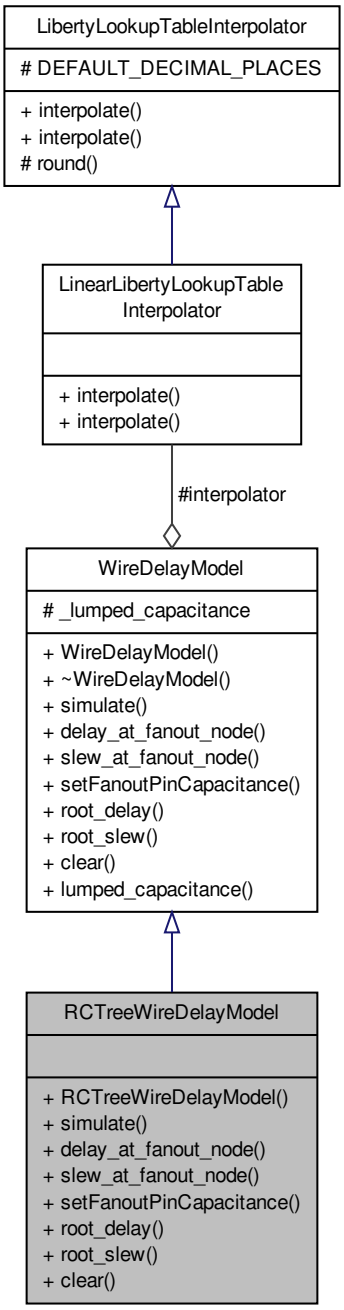
This model describes the parasitic elements(capacitors and resistors) distributed along a wire as a single circuit element.

```
#include <wire_delay_model.h>
```

Inheritance diagram for RCTreeWireDelayModel:



Collaboration diagram for RCTreeWireDelayModel:



Public Member Functions

- RCTreeWireDelayModel (const SpefNetISPD2013 &descriptor, const string rootNode, const size_t arcs_size, const bool dummyEdge=false)

  RCtreeWireDelayModel default constructor.

- const Transitions< double > simulate (const LibertyCellInfo &cellInfo, const int input, const Transitions< double > slew, bool is_input_driver)

Calculates and returns Transitions<double> representing effective capacitance of the wire.

- const Transitions< double > delay_at_fanout_node (const string fanout_node_name) const

  Returns maximum delay time value at node fanout_node_name.
- const Transitions< double > slew_at_fanout_node (const string fanout_node_name) const

  Returns maximum slew time value at node fanout_node_name.
- void setFanoutPinCapacitance (const string fanoutNameAndPin, const double pinCapacitance)

  Increments effective capacitance of fanoutNameAndPin by pinCapacitance.
- Transitions< double > root_delay (int arc_number)

  Returns Transitions<double> representing delay time value at arc_number.
- Transitions< double > root_slew (int arc_number)

  Returns Transitions<double> representing slew time value at arc_number.
- void clear ()

  Sets slew and delay time values to zero.

Additional Inherited Members

7.27.1 Detailed Description

This model describes the parasitic elements(capacitors and resistors) distributed along a wire as a single circuit element.

7.27.2 Constructor & Destructor Documentation

7.27.2.1 RCTreeWireDelayModel::RCTreeWireDelayModel (const SpefNetISPD2013 & *descriptor,* const string *rootNode,* const size_t *arcs_size,* const bool *dummyEdge* = `false`)

RCtreeWireDelayModel default constructor.

Parameters

const	SpefNetISPD2013 & descriptor, const string rootNode, const size_t arcs_size, const bool dummyEdge(false default)

7.27.3 Member Function Documentation

7.27.3.1 void RCTreeWireDelayModel::clear ( ) `[virtual]`

Sets slew and delay time values to zero.

Returns

void

Implements WireDelayModel.

7.27.3.2 const Transitions<double> RCTreeWireDelayModel::delay_at_fanout_node (const string *fanout_node_name*) const `[virtual]`

Returns maximum delay time value at node fanout_node_name.

Parameters

const	string fanout_node_name

Returns

const Transitions<double>

Implements WireDelayModel.

7.27.3.3 Transitions<double> RCTreeWireDelayModel::root_delay (int *arc_number*) [virtual]

Returns Transitions<double> representing delay time value at arc_number.

Parameters

int	arc_number

Returns

Transitions<double>

Implements WireDelayModel.

7.27.3.4 Transitions<double> RCTreeWireDelayModel::root_slew (int *arc_number*) [virtual]

Returns Transitions<double> representing slew time value at arc_number.

Parameters

int	arc_number

Returns

Transitions<double>

Implements WireDelayModel.

7.27.3.5 void RCTreeWireDelayModel::setFanoutPinCapacitance (const string *fanoutNameAndPin,* const double *pinCapacitance*) [virtual]

Increments effective capacitance of fanoutNameAndPin by pinCapacitance.

Parameters

const	string fanoutNameAndPin, const double pinCapacitance

Returns

void

Implements WireDelayModel.

7.27.3.6 const Transitions<double> RCTreeWireDelayModel::simulate (const LibertyCellInfo & *cellInfo,* const int *input,* const Transitions< double > *slew,* bool *is_input_driver*) [virtual]

Calculates and returns Transitions<double> representing effective capacitance of the wire.

Parameters

const	LibertyCellInfo & cellInfo, const int input, const Transitions<double> slew, bool is_input_driver

Returns

const Transitions<double>

Implements WireDelayModel.

7.27.3.7 const Transitions<double> RCTreeWireDelayModel::slew_at_fanout_node (const string *fanout_node_name*) const [virtual]

Returns maximum slew time value at node fanout_node_name.

Parameters

const	string fanout_node_name

Returns

const Transitions<double>

Implements WireDelayModel.

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

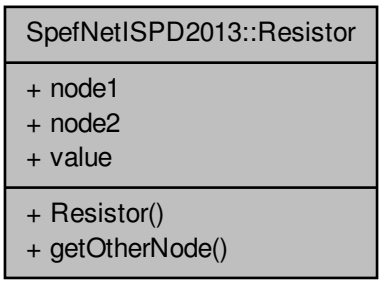
- TimingAnalysis/src/include/wire_delay_model.h

7.28 SpefNetISPD2013::Resistor Struct Reference

Struct which represents a resistor.

`#include <spef_net.h>`

Collaboration diagram for SpefNetISPD2013::Resistor:



Public Member Functions

- Resistor (const int &node1, const int &node2, const double &value)

Resistor constructor.

- int getOtherNode (const int &node) const

  If called by node1, returns node2, and vice versa.

Public Attributes

- int node1
- int node2
- double value

7.28.1 Detailed Description

Struct which represents a resistor.

7.28.2 Constructor & Destructor Documentation

7.28.2.1 SpefNetlSPD2013::Resistor::Resistor (const int & *node1*, const int & *node2*, const double & *value*) [inline]

Resistor constructor.

Parameters

const	string & node1, const string & node2, const double & value

7.28.3 Member Function Documentation

7.28.3.1 int SpefNetlSPD2013::Resistor::getOtherNode (const int & *node*) const [inline]

If called by node1, returns node2, and vice versa.

Parameters

const	string & node1, const string & node2, const double & value

Returns

int

7.28.4 Member Data Documentation

7.28.4.1 int SpefNetlSPD2013::Resistor::node1

7.28.4.2 int SpefNetlSPD2013::Resistor::node2

7.28.4.3 double SpefNetlSPD2013::Resistor::value

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

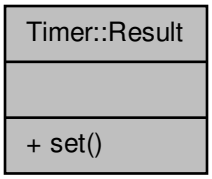
- TimingAnalysis/src/include/spef_net.h

7.29 Timer::Result Class Reference

Represents the result from the Timer class.

`#include <timer.h>`

Collaboration diagram for Timer::Result:



Public Member Functions

- void set (float time, string unity)

  Sets time and unity atributes.

Friends

- class Timer
- ostream & operator<< (ostream &out, const Timer::Result &result)

  Redefinition of << operator. Inserts formatted description of Timer containing its result time within its unity.

7.29.1 Detailed Description

Represents the result from the Timer class.

7.29.2 Member Function Documentation

7.29.2.1 void Timer::Result::set (float *time*, string *unity*) [inline]

Sets time and unity atributes.

Parameters

float	time, string unity

Returns

void

7.29.3 Friends And Related Function Documentation

7.29.3.1 ostream& operator<< (ostream & *out*, const Timer::Result & *result*) [friend]

Redefinition of << operator. Inserts formatted description of Timer containing its result time within its unity.

Parameters

ostream	& out, const Timer::Result & result

7.29.3.2 friend class Timer [friend]

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

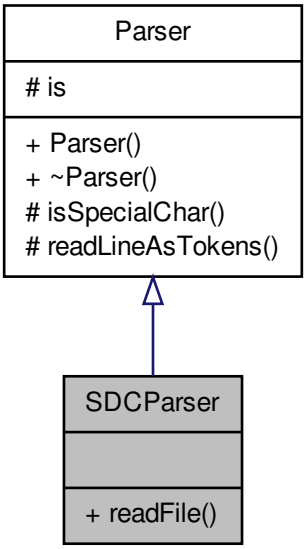
- TimingAnalysis/src/include/timer.h

7.30 SDCParser Class Reference

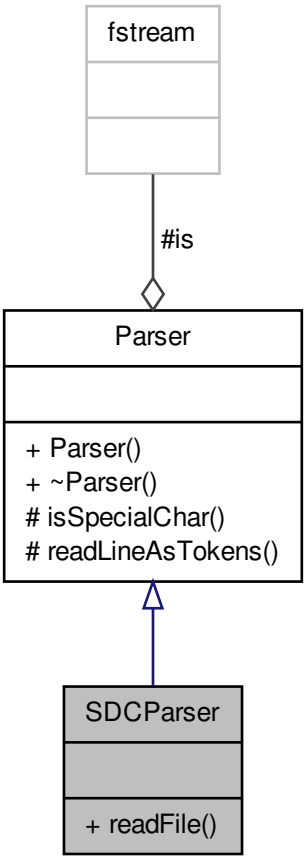
Inherits from Parser. SDC stands for Synopsys Design Constraints. Used to generate a Design_Constraints object from file.

`#include <parser.h>`

Inheritance diagram for SDCParser:



Collaboration diagram for SDCParser:



Public Member Functions

- const Design_Constraints readFile (const string filename)

  Returns Design_Constraints object generated using informations read from file passed as filename parameter.

Additional Inherited Members

7.30.1 Detailed Description

Inherits from Parser. SDC stands for Synopsys Design Constraints. Used to generate a Design_Constraints object from file.

7.30.2 Member Function Documentation

7.30.2.1 const Design_Constraints SDCParser::readFile (const string *filename*)

Returns Design_Constraints object generated using informations read from file passed as filename parameter.

Parameters

const	string filename

Returns

const Design_Constraints

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

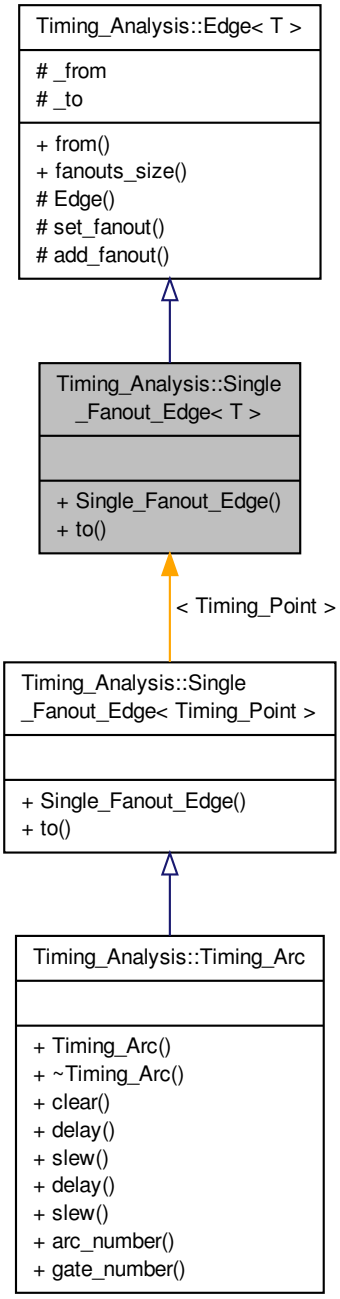
- TimingAnalysis/src/include/parser.h

7.31 Timing_Analysis::Single_Fanout_Edge< T > Class Template Reference

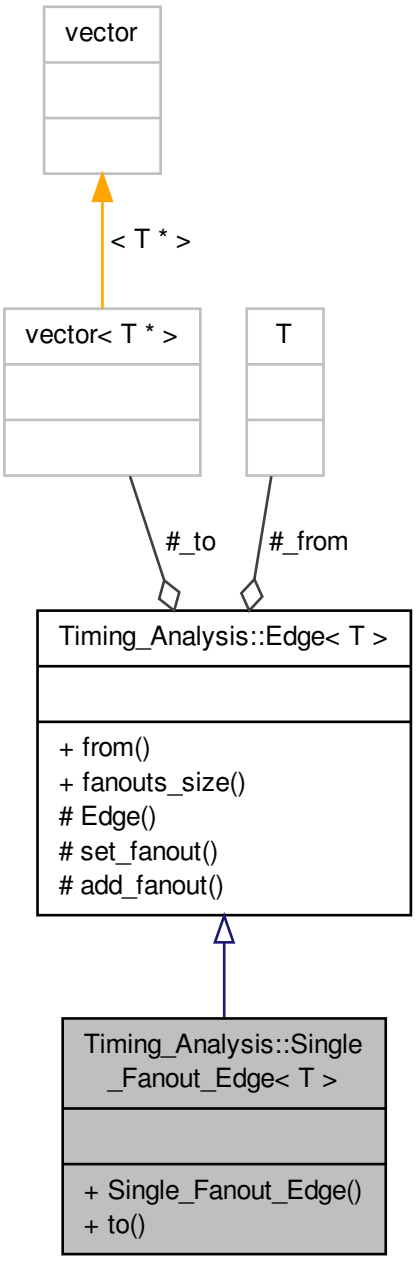
Inherits from Edge. A single fanout edge is an edge which connects two vertex to each other.

```
#include <single_fanout_edge.h>
```

Inheritance diagram for Timing_Analysis::Single_Fanout_Edge< T >:



Collaboration diagram for Timing_Analysis::Single_Fanout_Edge< T >:



Public Member Functions

- Single_Fanout_Edge (T *from, T *to)

  Single_Fanout_Edge constructor.
- T & to (void) const

  Returns Edge<T>

Additional Inherited Members

7.31.1 Detailed Description

template<class T>class Timing_Analysis::Single_Fanout_Edge< T >

Inherits from Edge. A single fanout edge is an edge which connects two vertex to each other.

7.31.2 Constructor & Destructor Documentation

7.31.2.1 template<class T> Timing_Analysis::Single_Fanout_Edge< T >::Single_Fanout_Edge (T * *from*, T * *to*) [inline]

Single_Fanout_Edge constructor.

7.31.3 Member Function Documentation

7.31.3.1 template<class T> T& Timing_Analysis::Single_Fanout_Edge< T >::to (void) const [inline]

Returns Edge<T>

Returns

T &

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

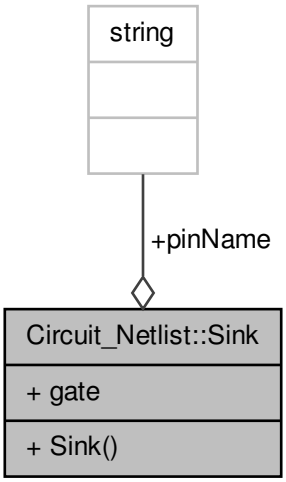
- TimingAnalysis/src/include/single_fanout_edge.h

7.32 Circuit_Netlist::Sink Struct Reference

Struct which represents a Sink. A sink is a vertex which has input Edge's, but no output Edge's (outdegree zero).

`#include <circuit_netlist.h>`

Collaboration diagram for Circuit_Netlist::Sink:



Public Member Functions

- Sink (const int gate, const string pinName)

  Sink constructor.

Public Attributes

- int gate
- string pinName

7.32.1 Detailed Description

Struct which represents a Sink. A sink is a vertex which has input Edge's, but no output Edge's (outdegree zero).

7.32.2 Constructor & Destructor Documentation

7.32.2.1 Circuit_Netlist::Sink::Sink (const int *gate*, const string *pinName*) [inline]

Sink constructor.

Parameters

const	int gate, const string pinName

7.32.3 Member Data Documentation

7.32.3.1 int Circuit_Netlist::Sink::gate

7.32.3.2 string Circuit_Netlist::Sink::pinName

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

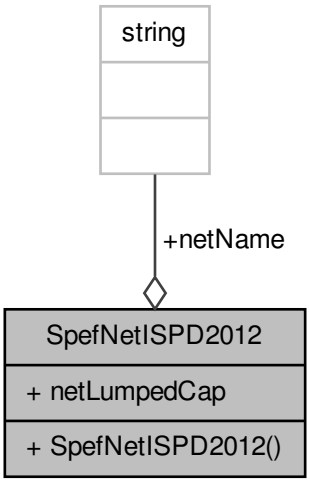
- TimingAnalysis/src/include/circuit_netlist.h

7.33 SpefNetISPD2012 Class Reference

SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2012.

```
#include <spef_net.h>
```

Collaboration diagram for SpefNetISPD2012:



Public Member Functions

- SpefNetISPD2012 ()

Public Attributes

- string netName
- double netLumpedCap

7.33.1 Detailed Description

SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2012.

7.33.2 Constructor & Destructor Documentation

7.33.2.1 SpefNetISPD2012::SpefNetISPD2012 ( ) [inline]

7.33.3 Member Data Documentation

7.33.3.1 double SpefNetISPD2012::netLumpedCap

7.33.3.2 string SpefNetISPD2012::netName

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

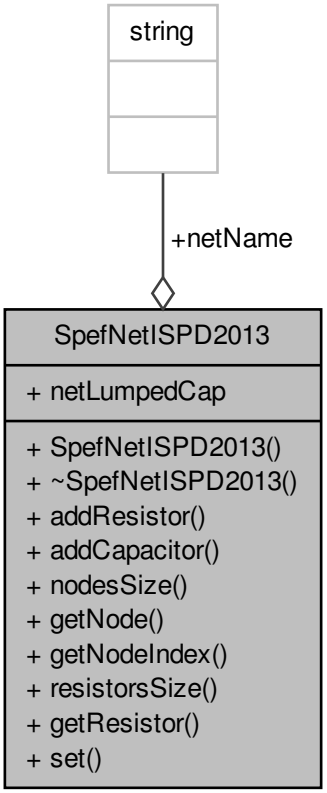
- TimingAnalysis/src/include/spef_net.h

7.34 SpefNetISPD2013 Class Reference

SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2013.

```
#include <spef_net.h>
```

Collaboration diagram for SpefNetISPD2013:



Classes

- struct Capacitor

  Struct which represents a capacitor.
- struct Node

  Struct which represents a node of the circuit. A node is a point of the circuit where two or more elements meet.
- struct Resistor

  Struct which represents a resistor.

Public Member Functions

- SpefNetISPD2013 ()

  Empty SpefNet constructor.
- virtual ~SpefNetISPD2013 ()

  Empty SpefNet destructor.
- void addResistor (const string &node1, const string &node2, const double &value)

  Adds new resistor to nodes/resistors list.
- void addCapacitor (const string &node, const double &value)

  Adds new capacitor to nodes/capacitors list.
- size_t nodesSize () const

  Returns number of nodes.
- const Node & getNode (const unsigned &i) const

  Gets node at index i.
- int getNodeIndex (const string &name) const

  Returns index of node name. If not found, returns -1.
- size_t resistorsSize () const

  Retuns number of resistors.
- const Resistor & getResistor (const unsigned &i) const

  Returns resistor at index i.
- void set (string name, double lumpedCapacitance)

  Sets SPEF netlist atributtes.

Public Attributes

- string netName
- double netLumpedCap

Friends

- ostream & operator<< (ostream &out, const SpefNetISPD2013 &descriptor)

  Redefinition of << operator. Inserts description including node name, index, capacitance and resistances list.

7.34.1 Detailed Description

SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2013.

7.34.2 Constructor & Destructor Documentation

7.34.2.1 SpefNetISPD2013::SpefNetISPD2013 ( ) [inline]

Empty SpefNet constructor.

7.34.2.2 virtual SpefNetISPD2013::~SpefNetISPD2013 ( ) [inline],[virtual]

Empty SpefNet destructor.

7.34.3 Member Function Documentation

7.34.3.1 void SpefNetISPD2013::addCapacitor (const string & *node,* const double & *value*)

Adds new capacitor to nodes/capacitors list.

Parameters

const	string & node, const double & value

Returns

void

7.34.3.2 void SpefNetISPD2013::addResistor (const string & *node1,* const string & *node2,* const double & *value*)

Adds new resistor to nodes/resistors list.

Parameters

const	string & node1, const string & node2, const double & value

Returns

void

7.34.3.3 const Node& SpefNetISPD2013::getNode (const unsigned & *i*) const [inline]

Gets node at index i.

Parameters

const	unsigned & i

Returns

const Node

7.34.3.4 int SpefNetISPD2013::getNodeIndex (const string & *name*) const

Returns index of node name. If not found, returns -1.

Parameters

const	string & name

Returns

int

7.34.3.5 const Resistor& SpefNetlSPD2013::getResistor (const unsigned & *i*) const [inline]

Returns resistor at index i.

Parameters

const	unsigned & i

Returns

const Resistor

7.34.3.6 size_t SpefNetlSPD2013::nodesSize ( ) const [inline]

Returns number of nodes.

Returns

size_t

7.34.3.7 size_t SpefNetlSPD2013::resistorsSize ( ) const [inline]

Retuns number of resistors.

Returns

size_t

7.34.3.8 void SpefNetlSPD2013::set (string *name,* double *lumpedCapacitance*)

Sets SPEF netlist atributtes.

Parameters

string	name,double lumpedCapacitance

Returns

void

7.34.4 Friends And Related Function Documentation

7.34.4.1 ostream& operator<< (ostream & *out,* const SpefNetlSPD2013 & *descriptor*) [friend]

Redefinition of << operator. Inserts description including node name, index, capacitance and resistances list.

7.34.5 Member Data Documentation

7.34.5.1 double SpefNetISPD2013::netLumpedCap

7.34.5.2 string SpefNetISPD2013::netName

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

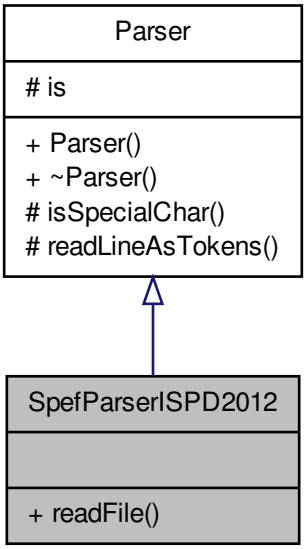
- TimingAnalysis/src/include/spef_net.h

7.35 SpefParserISPD2012 Class Reference

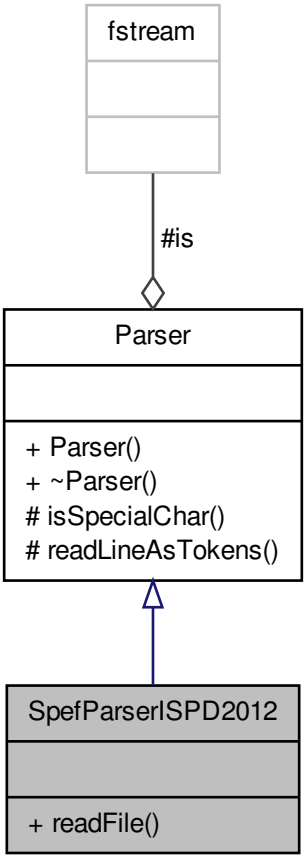
Inherits from Parser. Used to generate SpefNetISPD2012 files from an input.

```
#include <parser.h>
```

Inheritance diagram for SpefParserISPD2012:



Collaboration diagram for SpefParserISPD2012:



Public Member Functions

- const Parasitics2012 readFile (const string filename)

  Returns Parasitics2012 object generated using informations read from file passed as filename parameter.

Additional Inherited Members

7.35.1 Detailed Description

Inherits from Parser. Used to generate SpefNetISPD2012 files from an input.

7.35.2 Member Function Documentation

7.35.2.1 const Parasitics2012 SpefParserISPD2012::readFile (const string *filename*)

Returns Parasitics2012 object generated using informations read from file passed as filename parameter.

Parameters

const	string filename

Returns

Parasitics2012

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

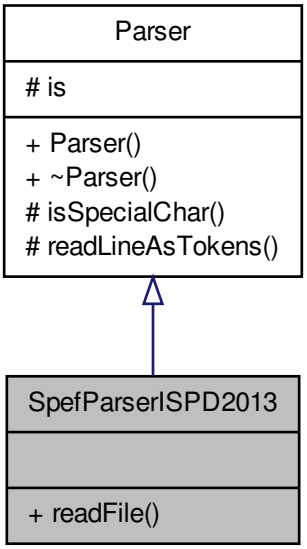
- TimingAnalysis/src/include/parser.h

7.36 SpefParserISPD2013 Class Reference

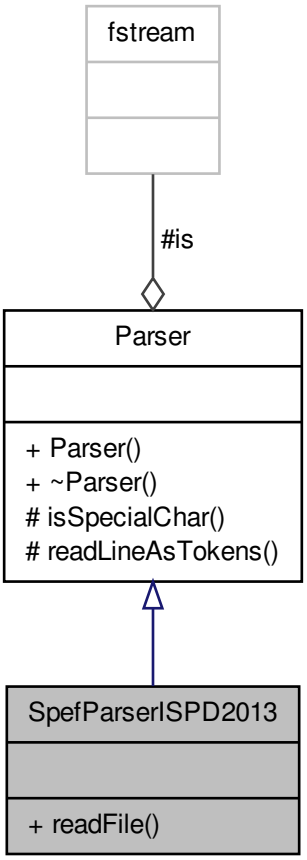
Inherits from Parser. Used to generate SpefNetISPD2013 file from an input.

`#include <parser.h>`

Inheritance diagram for SpefParserISPD2013:



Collaboration diagram for SpefParserISPD2013:



Public Member Functions

- const Parasitics2013 readFile (const string filename)

  Returns Parasitics2013 object generated using informations read from file passed as filename parameter.

Additional Inherited Members

7.36.1 Detailed Description

Inherits from Parser. Used to generate SpefNetISPD2013 file from an input.

7.36.2 Member Function Documentation

7.36.2.1 const Parasitics2013 SpefParserISPD2013::readFile (const string *filename*)

Returns Parasitics2013 object generated using informations read from file passed as filename parameter.

Parameters

const	string filename

Returns

Parasitics2013

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

- TimingAnalysis/src/include/parser.h

7.37 Timer Class Reference

Represents a Timer.

`#include <timer.h>`

Collaboration diagram for Timer:

Classes

- class Result

  Represents the result from the Timer class.

Public Member Functions

- Timer ()

  Timer empty constructor.
- virtual ~Timer ()

  Timer empty destructor.
- void start ()

  Starts Timer counting.
- void end ()

  Ends the counting and sets the execution time.
- const Timer::Result & value (const double time_definition)

  Returns execution time.

Static Public Attributes

- static const double MICRO
- static const double MILI

7.37.1 Detailed Description

Represents a Timer.

7.37.2 Constructor & Destructor Documentation

7.37.2.1 Timer::Timer ( )

Timer empty constructor.

7.37.2.2 virtual Timer::~Timer ( ) [virtual]

Timer empty destructor.

7.37.3 Member Function Documentation

7.37.3.1 void Timer::end ( )

Ends the counting and sets the execution time.

Returns

void

7.37.3.2 void Timer::start ( )

Starts Timer counting.

Returns

void

7.37.3.3 const Timer::Result& Timer::value (const double *time_definition*)

Returns execution time.

Parameters

const	double time_definition

Returns

const Timer::Result &

7.37.4 Member Data Documentation

7.37.4.1 const double Timer::MICRO [static]

7.37.4.2 const double Timer::MILI [static]

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

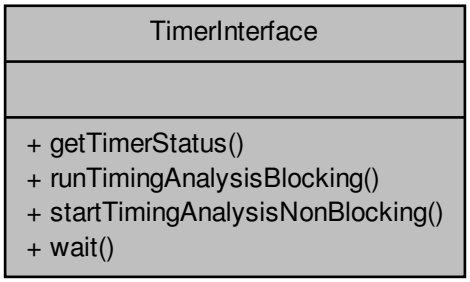
- TimingAnalysis/src/include/timer.h

7.38 TimerInterface Class Reference

This class contains functions for the timing analysis interface. To use any function belonging to this class, call TimerInterface::<function_name>(<argument_list>);.

```
#include <timer_interface.h>
```

Collaboration diagram for TimerInterface:



Public Types

- enum Status {
  TIMER_NOT_STARTED = 0, TIMER_BUSY, TIMER_FINISHED_SUCCESS, TIMER_FINISHED_ERROR, TIMER_INTERFACEERROR }

Static Public Member Functions

- static Status getTimerStatus (const std::string &contest_root, const std::string &benchmark)
- static Status runTimingAnalysisBlocking (const std::vector< std::pair< std::string, std::string > > &sizes, const std::string &contest_root, const std::string &benchmark, const unsigned pollingTime)
- static Status startTimingAnalysisNonBlocking (const std::vector< std::pair< std::string, std::string > > &sizes, const std::string &contest_root, const std::string &benchmark)
- static void wait (int seconds)

7.38.1 Detailed Description

This class contains functions for the timing analysis interface. To use any function belonging to this class, call TimerInterface::<function_name>(<argument_list>);.

Timing Analysis Interface helper class to interface with the timer. This code is provided for description purposes only. The contest organizers cannot guarantee that the provided code is free of bugs or defects. !!!! USE THIS CODE AT YOUR OWN RISK !!!!!

The contestants are free to use these functions as-is or make modifications. If the contestants choose to use the provided code, they are responsible for making sure that it works as expected.

The code provided here has no real or implied warranties.

7.38.2 Member Enumeration Documentation

7.38.2.1 enum TimerInterface::Status

Enumerator

TIMER_NOT_STARTED

TIMER_BUSY

TIMER_FINISHED_SUCCESS

TIMER_FINISHED_ERROR

TIMER_INTERFACEERROR

7.38.3 Member Function Documentation

7.38.3.1 static Status TimerInterface::getTimerStatus (const std::string & *contest_root,* const std::string & *benchmark*) [static]

Get timer status Inputs: contest root directory (string) benchmark name (string) Return: status (see enum Status above)

7.38.3.2 static Status TimerInterface::runTimingAnalysisBlocking (const std::vector< std::pair< std::string, std::string > > & *sizes,* const std::string & *contest_root,* const std::string & *benchmark,* const unsigned *pollingTime*) [static]

Write sizes and run timing analysis in blocking mode

1. Write sizes
2. Starts timing analysis
3. Waits for timing analysis to be completed Inputs: vector of pairs where first value is instance name (string) and second value is cell name (string) contest root directory (string) benchmark name (string) polling time (number of seconds that the function should wait before polling timer status to check whether timer is done) Return: timer status

7.38.3.3 static Status TimerInterface::startTimingAnalysisNonBlocking (const std::vector< std::pair< std::string, std::string > > & *sizes,* const std::string & *contest_root,* const std::string & *benchmark*) [static]

Start timing analysis in non-blocking mode

1. Write sizes
2. Starts timing analysis and returns (does not wait for timing analysis to be completed) Inputs: vector of pairs where first value is instance name (string) and second value is cell name (string) contest root directory (string) benchmark name (string) Return: timer status

7.38.3.4 static void TimerInterface::wait (int *seconds*) [static]

Wait for given number of seconds (useful function if you want to wait before checking timer status after calling startTimingAnalysisNonBlocking) Input: seconds to wait

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

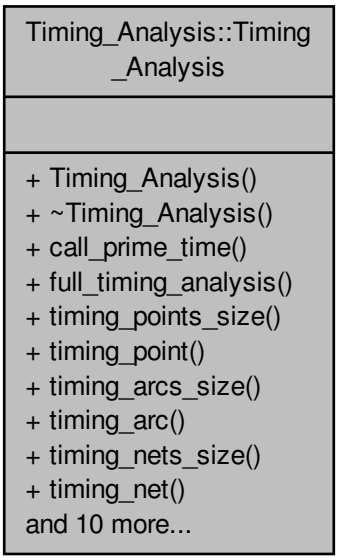
- TimingAnalysis/src/include/timer_interface.h

7.39 Timing_Analysis::Timing_Analysis Class Reference

Class which performs the timing analysis.

`#include <timing_analysis.h>`

Collaboration diagram for Timing_Analysis::Timing_Analysis:



Public Member Functions

- Timing_Analysis (const Circuit_Netlist &netlist, const LibertyLibrary ∗lib, const Parasitics ∗_parasitics, const Design_Constraints ∗sdc)

  Timing_Analysis default constructor.
- virtual ∼Timing_Analysis ()
- void call_prime_time ()

  Calls prime time to set time values.
- void full_timing_analysis ()

  Updates all timing values of all elements.
- size_t timing_points_size ()

  Returns number of timing points.
- const Timing_Point & timing_point (const int i)

Returns timing point reference at index i.

- size_t timing_arcs_size ()

  Returns number of timing arcs.
- const Timing_Arc & timing_arc (const int i)

  Returns timing arc reference at index i.
- size_t timing_nets_size ()

  Returns number of timing nets.
- const Timing_Net & timing_net (const int i)

  Returns timing net reference at index i.
- double pin_capacitance (const int timing_point_index) const

  Returns pin capacitance at timing point index.
- double pin_load (const int timing_point_index) const

  Returns pin load at timing point index.
- int option (const int gate_number)

  Returns option index at gate_number index.
- size_t number_of_options (const int gate_index)
- int first_PO_index () const

  Returns first primary output index.
- bool option (const int gate_index, const int option)

  Returns true if gate_index option is "don't touch".
- bool validate_with_prime_time ()

  Validates timing_file with prime time, prints results and returns true.
- void print_info ()

  Prints graph and circuit info.
- void print_circuit_info ()

  Prints timing info.
- void report_timing ()

  Prints results from analysis.

7.39.1 Detailed Description

Class which performs the timing analysis.

7.39.2 Constructor & Destructor Documentation

7.39.2.1 Timing_Analysis::Timing_Analysis::Timing_Analysis (const Circuit_Netlist & *netlist,* const LibertyLibrary ∗ *lib,* const Parasitics ∗ *_parasitics,* const Design_Constraints ∗ *sdc*)

Timing_Analysis default constructor.

Parameters

const	Circuit_Netlist & netlist, const LibertyLibrary ∗ lib, const Parasitics ∗ _parasitics, const Design_Constraints ∗ sdc

7.39.2.2 virtual Timing_Analysis::Timing_Analysis::∼Timing_Analysis ( ) [virtual]

Timing_Analysis empty destructor

7.39.3 Member Function Documentation

7.39.3.1 void Timing_Analysis::Timing_Analysis::call_prime_time ( )

Calls prime time to set time values.

Returns

void

7.39.3.2 int Timing_Analysis::Timing_Analysis::first_PO_index ( ) const [inline]

Returns first primary output index.

Returns

int

7.39.3.3 void Timing_Analysis::Timing_Analysis::full_timing_analysis ( )

Updates all timing values of all elements.

Returns

void

7.39.3.4 size_t Timing_Analysis::Timing_Analysis::number_of_options (const int *gate_index*)

7.39.3.5 int Timing_Analysis::Timing_Analysis::option (const int *gate_number*)

Returns option index at gate_number index.

Parameters

const	int gate_number

Returns

int

7.39.3.6 bool Timing_Analysis::Timing_Analysis::option (const int *gate_index,* const int *option*)

Returns true if gate_index option is "don't touch".

Parameters

const	int gate_index, const int option

Returns

bool

7.39.3.7 double Timing_Analysis::Timing_Analysis::pin_capacitance (const int *timing_point_index*) const

Returns pin capacitance at timing point index.

Parameters

const	int timing_point_index

Returns

double

7.39.3.8 double Timing_Analysis::Timing_Analysis::pin_load (const int *timing_point_index*) const

Returns pin load at timing point index.

Parameters

const	int i

Returns

const Timing_Net &

7.39.3.9 void Timing_Analysis::Timing_Analysis::print_circuit_info ( )

Prints timing info.

Returns

void

7.39.3.10 void Timing_Analysis::Timing_Analysis::print_info ( )

Prints graph and circuit info.

Returns

void

7.39.3.11 void Timing_Analysis::Timing_Analysis::report_timing ( )

Prints results from analysis.

Returns

void

7.39.3.12 const Timing_Arc& Timing_Analysis::Timing_Analysis::timing_arc (const int *i*) [inline]

Returns timing arc reference at index i.

Parameters

const	int i

Returns

const Timing_Arc &

7.39.3.13 size_t Timing_Analysis::Timing_Analysis::timing_arcs_size ( ) [inline]

Returns number of timing arcs.

Returns

size_t

7.39.3.14 const Timing_Net& Timing_Analysis::Timing_Analysis::timing_net (const int *i*) [inline]

Returns timing net reference at index i.

Parameters

const	int i

Returns

const Timing_Net &

7.39.3.15 size_t Timing_Analysis::Timing_Analysis::timing_nets_size ( ) [inline]

Returns number of timing nets.

Returns

size_t

7.39.3.16 const Timing_Point& Timing_Analysis::Timing_Analysis::timing_point (const int *i*) [inline]

Returns timing point reference at index i.

Parameters

const	int i

Returns

const Timing_Point &

7.39.3.17 size_t Timing_Analysis::Timing_Analysis::timing_points_size ( ) [inline]

Returns number of timing points.

Returns

size_t

7.39.3.18 bool Timing_Analysis::Timing_Analysis::validate_with_prime_time ( )

Validates timing_file with prime time, prints results and returns true.

Returns

bool

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

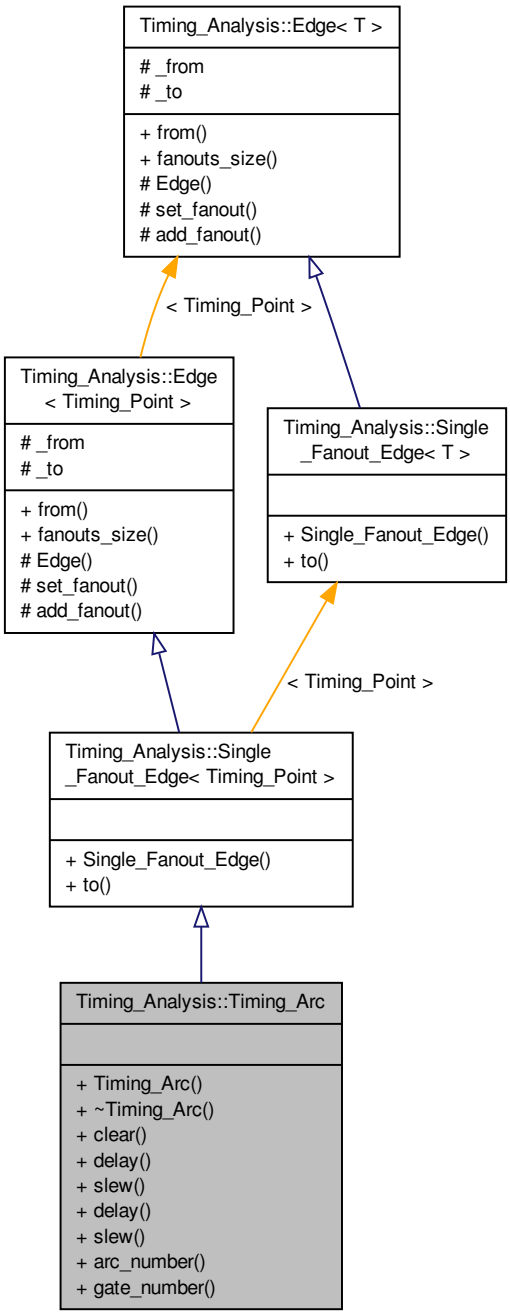
- TimingAnalysis/src/include/timing_analysis.h

7.40 Timing_Analysis::Timing_Arc Class Reference

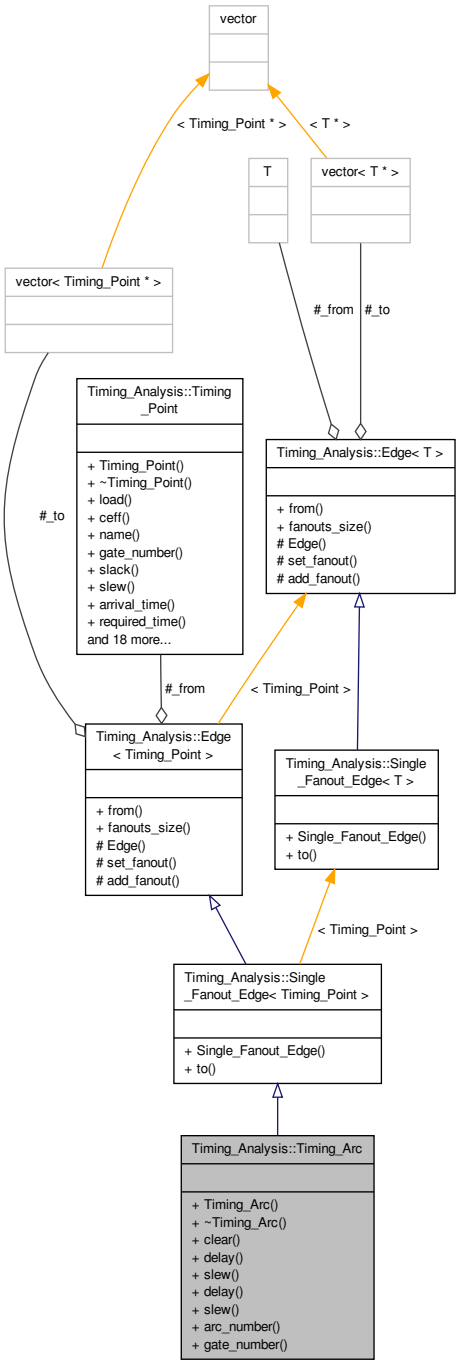
Describes a Timing_Arc of the timing graph model. Vertex which connects one Timing_Point to another timing point only.

```
#include <timing_arc.h>
```

Inheritance diagram for Timing_Analysis::Timing_Arc:



Collaboration diagram for Timing_Analysis::Timing_Arc:



Public Member Functions

- Timing_Arc (Timing_Point *from, Timing_Point *to, const int arcNumber, const int gate_number)

  Timing_Arc constructor.
- virtual ~Timing_Arc ()

  Timing_Arc destructor.
- void clear ()

Sets delay and slew time to zero.

- Transitions< double > delay () const

  Returns delay time.
- Transitions< double > slew () const

  Returns slew time.
- void delay (const Transitions< double > &delay)

  Sets delay time.
- void slew (const Transitions< double > &slew)

  Sets slew time.
- int arc_number () const

  Returns arc number.
- int gate_number () const

  Returns gate number.

Friends

- ostream & operator<< (ostream &out, const Timing_Arc &ta)

  Inserts formatted description of timing arc, including the origin vertex and destiny vertex.

Additional Inherited Members

7.40.1 Detailed Description

Describes a Timing_Arc of the timing graph model. Vertex which connects one Timing_Point to another timing point only.

7.40.2 Constructor & Destructor Documentation

7.40.2.1 Timing_Analysis::Timing_Arc::Timing_Arc (Timing_Point * *from*, Timing_Point * *to*, const int *arcNumber*, const int *gate_number*) [inline]

Timing_Arc constructor.

Parameters

Timing_Point	* from, Timing_Point * to, const int arcNumber, const int gate_number) : Single_Fanout_Edge<Timing_Point>(from, to), _delay(0.0f, 0.0f), _slew(0.0f, 0.0f), _arc_number (arcNumber), _gate_number(gate_number)

7.40.2.2 virtual Timing_Analysis::Timing_Arc::~Timing_Arc ( ) [inline],[virtual]

Timing_Arc destructor.

7.40.3 Member Function Documentation

7.40.3.1 int Timing_Analysis::Timing_Arc::arc_number ( ) const [inline]

Returns arc number.

Returns

int

7.40.3.2 void Timing_Analysis::Timing_Arc::clear ( )

Sets delay and slew time to zero.

Returns

void

7.40.3.3 Transitions<double> Timing_Analysis::Timing_Arc::delay ( ) const [inline]

Returns delay time.

Returns

Transitions<double>

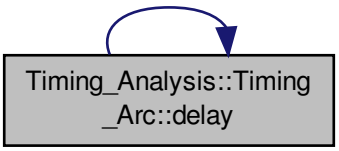
7.40.3.4 void Timing_Analysis::Timing_Arc::delay (const Transitions< double > & *delay*) [inline]

Sets delay time.

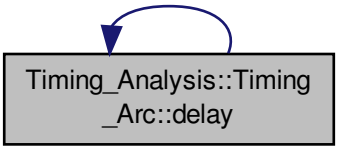
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



7.40.3.5 int Timing_Analysis::Timing_Arc::gate_number ( ) const [inline]

Returns gate number.

Returns

void

7.40.3.6 Transitions<double> Timing_Analysis::Timing_Arc::slew ( ) const [inline]

Returns slew time.

Returns

Transitions<double>

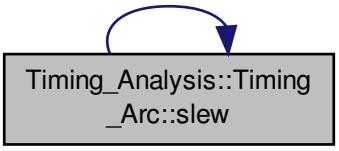
7.40.3.7 void Timing_Analysis::Timing_Arc::slew (const Transitions< double > & *slew*) [inline]

Sets slew time.

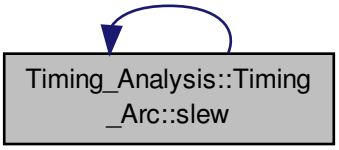
Returns

void

Here is the call graph for this function:



Here is the caller graph for this function:



7.40.4 Friends And Related Function Documentation

7.40.4.1 ostream& operator<< (ostream & *out*, const Timing_Arc & *ta*) [friend]

Inserts formatted description of timing arc, including the origin vertex and destiny vertex.

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

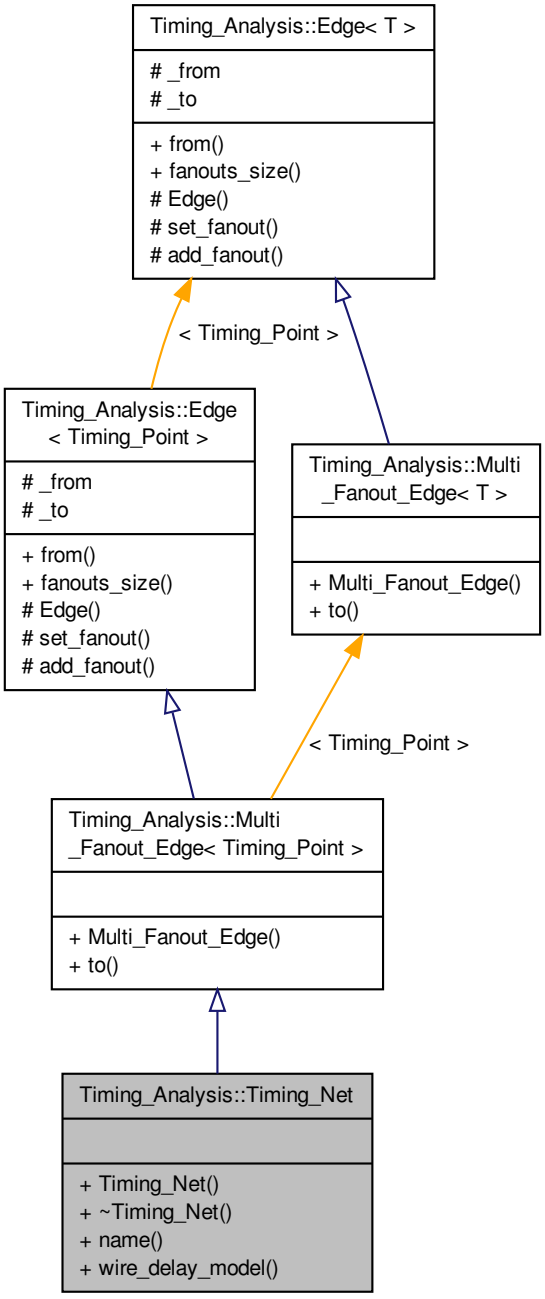
- TimingAnalysis/src/include/timing_arc.h

7.41 Timing_Analysis::Timing_Net Class Reference

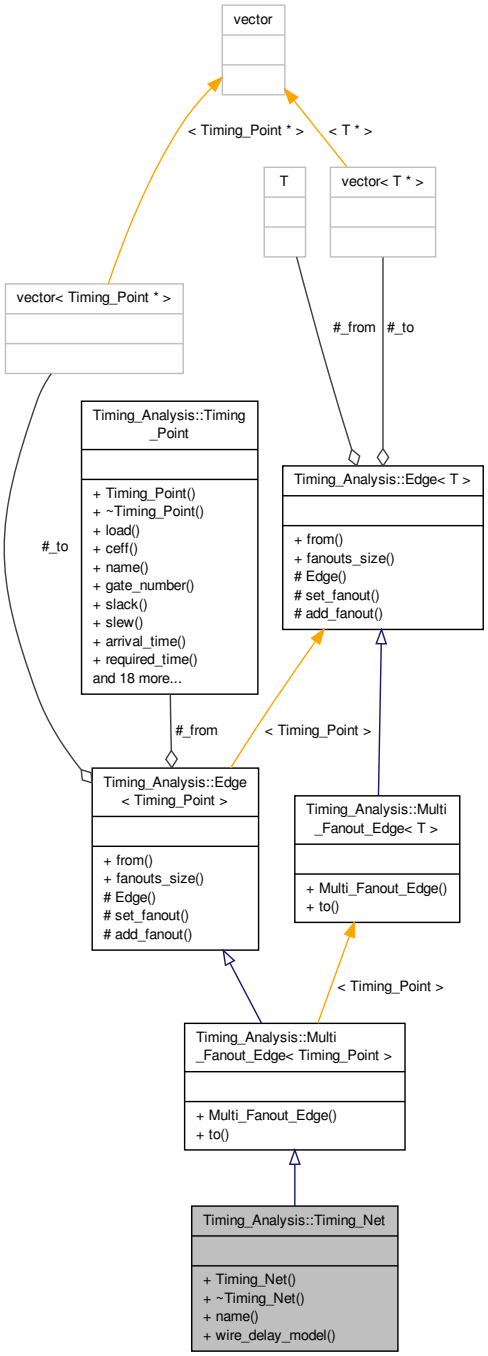
Describes a Timing_Net of the timing graph model. Vertex which connects one Timing_Point to multiple Timing_Point's.

```
#include <timing_net.h>
```

Inheritance diagram for Timing_Analysis::Timing_Net:



Collaboration diagram for Timing_Analysis::Timing_Net:



Public Member Functions

- Timing_Net (const string &name, Timing_Point *from, WireDelayModel *wire_delay_model)

  Timing_Net constructor.
- virtual ~Timing_Net ()

  Empty Timing_Net destructor.
- const string name () const

Returns Timing_Net name.

- WireDelayModel * wire_delay_model ()

  Returns pointer to WireDelayModel.

Friends

- class Timing_Analysis
- class Timing_Point
- ostream & operator<< (ostream &out, const Timing_Net &tn)

  Redefinition of << operator. Inserts formatted description including Timing_Net name.

Additional Inherited Members

7.41.1 Detailed Description

Describes a Timing_Net of the timing graph model. Vertex which connects one Timing_Point to multiple Timing_Point's.

7.41.2 Constructor & Destructor Documentation

7.41.2.1 Timing_Analysis::Timing_Net::Timing_Net (const string & *name*, Timing_Point * *from*, WireDelayModel * *wire_delay_model*) [inline]

Timing_Net constructor.

Parameters

const	string & name, Timing_Point * from, WireDelayModel * wire_delay_model

7.41.2.2 virtual Timing_Analysis::Timing_Net::~Timing_Net ( ) [inline], [virtual]

Empty Timing_Net destructor.

7.41.3 Member Function Documentation

7.41.3.1 const string Timing_Analysis::Timing_Net::name ( ) const

Returns Timing_Net name.

Returns

const string

7.41.3.2 WireDelayModel* Timing_Analysis::Timing_Net::wire_delay_model ( ) [inline]

Returns pointer to WireDelayModel.

Returns

WireDelayModel *

7.41.4 Friends And Related Function Documentation

7.41.4.1 ostream& operator<< (ostream & *out*, const Timing_Net & *tn*) [friend]

Redefinition of << operator. Inserts formatted description including Timing_Net name.

7.41.4.2 friend class Timing_Analysis [friend]

7.41.4.3 friend class Timing_Point [friend]

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

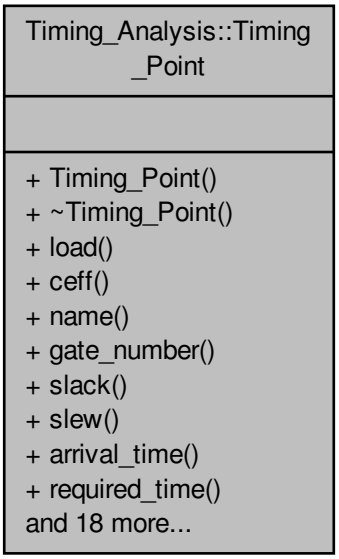
- TimingAnalysis/src/include/timing_net.h

7.42 Timing_Analysis::Timing_Point Class Reference

Describes a Timing_Point of the timing graph model.

`#include <timing_point.h>`

Collaboration diagram for Timing_Analysis::Timing_Point:



Public Member Functions

- Timing_Point (string name, const size_t gate_number, Timing_Point_Type type)

  Timing_Point default constructor.
- virtual ~Timing_Point ()

  Empty Timing_Point destructor.
- double load () const

Returns capacitive load of timing point.

- Transitions< double > ceff () const
- const string name () const

  Returns Timing_Net name.
- int gate_number () const

  Returns Timing_Net gate number.
- const Transitions< double > slack () const

  Returns Timing_Point slack time.
- const Transitions< double > slew () const

  Returns Timing_Point slew time.
- const Transitions< double > arrival_time () const

  Returns Timing_Point arrival time.
- const Transitions< double > required_time () const

  Returns Timing_Point required time.
- Timing_Net & net ()

  Returns reference to Timing_Net net.
- Timing_Arc & arc () const

  Returns reference to Timing_Arc arc.
- Timing_Net & net () const

  Sets the effective capacitance of the Timing_Net.
- Timing_Arc & arc () const
- void ceff (Transitions< double > &ceff)
- void slack (const Transitions< double > &slack)

  Sets the slack time of the Timing_Net.
- void slew (const Transitions< double > &slew)

  Sets the slew time of the Timing_Net.
- void arrival_time (const Transitions< double > &arrival_time)

  Sets the arrival time of the Timing_Net.
- void net (Timing_Net *net)

  Sets pointer to Timing_Net.
- void arc (Timing_Arc *arc)

  Sets pointer to Timing_Arc.
- const Transitions< double > update_slack (const Transitions< double > required_time)

  Returns updated slack. Slack = required_time - arrival_time.
- void clear_timing_info ()

  Slack, slew and arrival time parameters are set to zero.
- bool is_PO () const

  Returns true if caller is a primary ouput.
- bool is_PI () const

  Returns true if caller is a primary input.
- bool is_input_pin () const

  Returns true if caller is a input pin.
- bool is_output_pin () const

  Returns true if caller is a ouput pin.
- bool is_PI_input () const

  Returns true if caller is a primary_input input.
- bool is_reg_input () const

  Returns true if caller is a register input.

Friends

- class Timing_Net
- ostream & operator<< (ostream &out, const Timing_Point &tp)

  Redefinition of << operator. Inserts formatted description including arrival, required and slack times.

7.42.1 Detailed Description

Describes a Timing_Point of the timing graph model.

7.42.2 Constructor & Destructor Documentation

7.42.2.1 Timing_Analysis::Timing_Point::Timing_Point (string *name*, const size_t *gate_number*, Timing_Point_Type *type*)

Timing_Point default constructor.

Parameters

string	name, const size_t gate_number, Timing_Point_Type type

7.42.2.2 virtual Timing_Analysis::Timing_Point::~Timing_Point ( ) [inline], [virtual]

Empty Timing_Point destructor.

7.42.3 Member Function Documentation

7.42.3.1 Timing_Arc& Timing_Analysis::Timing_Point::arc ( ) const [inline]

Returns reference to Timing_Arc arc.

Returns

Timing_Arc &

7.42.3.2 Timing_Arc& Timing_Analysis::Timing_Point::arc ( ) const [inline]

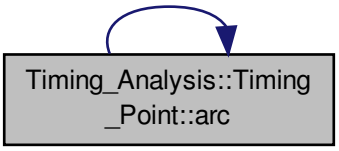
7.42.3.3 void Timing_Analysis::Timing_Point::arc (Timing_Arc * *arc*) [inline]

Sets pointer to Timing_Arc.

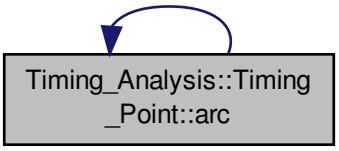
Parameters

Timing_Arc	* arc

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.4 const Transitions<double> Timing_Analysis::Timing_Point::arrival_time ( ) const [inline]

Returns Timing_Point arrival time.

Returns

const Transitions<double>

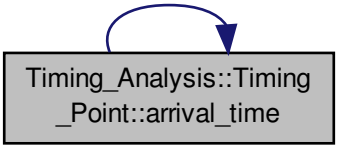
7.42.3.5 void Timing_Analysis::Timing_Point::arrival_time (const Transitions< double > & *arrival_time*) [inline]

Sets the arrival time of the Timing_Net.

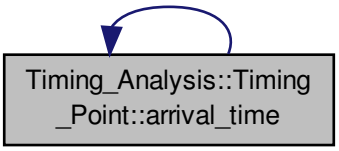
Parameters

Transitions<double>	& arrival_time

Here is the call graph for this function:



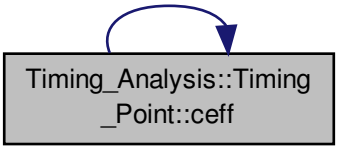
Here is the caller graph for this function:



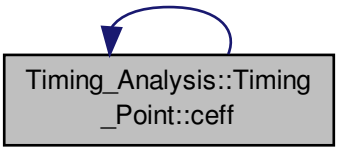
7.42.3.6 Transitions<double> Timing_Analysis::Timing_Point::ceff ( ) const

7.42.3.7 void Timing_Analysis::Timing_Point::ceff (Transitions< double > & *ceff*) [inline]

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.8 void Timing_Analysis::Timing_Point::clear_timing_info ( )

Slack, slew and arrival time parameters are set to zero.

Returns

void

7.42.3.9 int Timing_Analysis::Timing_Point::gate_number ( ) const [inline]

Returns Timing_Net gate number.

Returns

int

7.42.3.10 bool Timing_Analysis::Timing_Point::is_input_pin ( ) const [inline]

Returns true if caller is a input pin.

Returns

bool

7.42.3.11 bool Timing_Analysis::Timing_Point::is_output_pin ( ) const [inline]

Returns true if caller is a ouput pin.

Returns

bool

7.42.3.12 bool Timing_Analysis::Timing_Point::is_PI ( ) const [inline]

Returns true if caller is a primary input.

Returns

bool

7.42.3.13 bool Timing_Analysis::Timing_Point::is_PI_input ( ) const [inline]

Returns true if caller is a primary_input input.

Returns

bool

7.42.3.14 bool Timing_Analysis::Timing_Point::is_PO ( ) const [inline]

Returns true if caller is a primary ouput.

Returns

bool

7.42.3.15 bool Timing_Analysis::Timing_Point::is_reg_input ( ) const [inline]

Returns true if caller is a register input.

Returns

bool

7.42.3.16 double Timing_Analysis::Timing_Point::load ( ) const

Returns capacitive load of timing point.

Returns

double

7.42.3.17 const string Timing_Analysis::Timing_Point::name ( ) const [inline]

Returns Timing_Net name.

Returns

const string

7.42.3.18 Timing_Net& Timing_Analysis::Timing_Point::net ( ) [inline]

Returns reference to Timing_Net net.

Returns

Timing_Net &

7.42.3.19 Timing_Net& Timing_Analysis::Timing_Point::net ( ) const [inline]

Sets the effective capacitance of the Timing_Net.

Parameters

Transitions<double>	& ceff

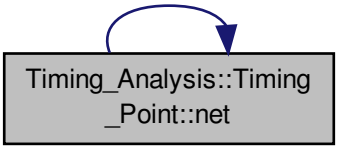
7.42.3.20 void Timing_Analysis::Timing_Point::net (Timing_Net * *net*) [inline]

Sets pointer to Timing_Net.

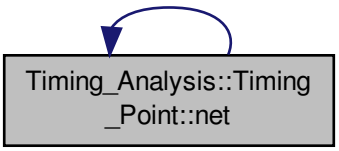
Parameters

Timing_Net	* net

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.21 const Transitions<double> Timing_Analysis::Timing_Point::required_time ( ) const [inline]

Returns Timing_Point required time.

Returns

const Transitions<double>

7.42.3.22 const Transitions<double> Timing_Analysis::Timing_Point::slack ( ) const [inline]

Returns Timing_Point slack time.

Returns

const Transitions<double>

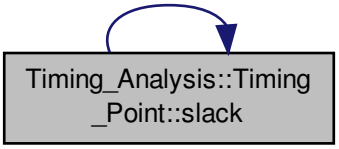
7.42.3.23 void Timing_Analysis::Timing_Point::slack (const Transitions< double > & *slack*) [inline]

Sets the slack time of the Timing_Net.

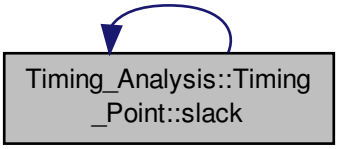
Parameters

Transitions<double>	& slack

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.24 const Transitions<double> Timing_Analysis::Timing_Point::slew ( ) const [inline]

Returns Timing_Point slew time.

Returns

const Transitions<double>

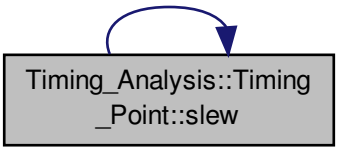
7.42.3.25 void Timing_Analysis::Timing_Point::slew (const Transitions< double > & *slew*) [inline]

Sets the slew time of the Timing_Net.

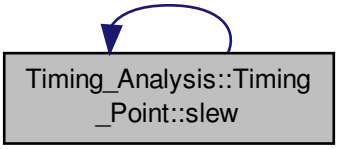
Parameters

Transitions<double>	& slew

Here is the call graph for this function:



Here is the caller graph for this function:



7.42.3.26 const Transitions<double> Timing_Analysis::Timing_Point::update_slack (const Transitions< double > *required_time*)

Returns updated slack. Slack = required_time - arrival_time.

Parameters

const	Transitions<double> required_time

Returns

const Transitions<double>

7.42.4 Friends And Related Function Documentation

7.42.4.1 ostream& operator<< (ostream & *out*, const Timing_Point & *tp*) [friend]

Redefinition of << operator. Inserts formatted description including arrival, required and slack times.

7.42.4.2 friend class Timing_Net [friend]

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

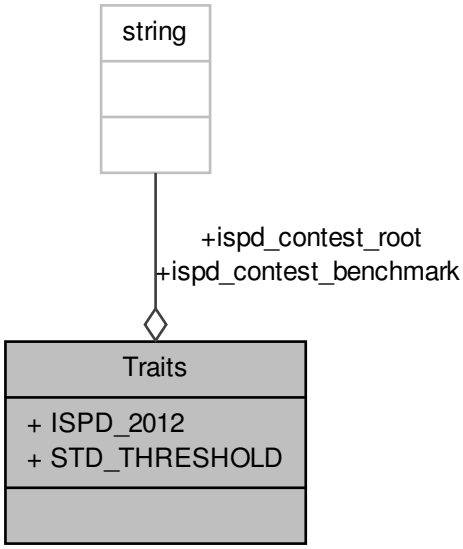
- TimingAnalysis/src/include/timing_point.h

7.43 Traits Class Reference

User configurable.

```
#include <configuration.h>
```

Collaboration diagram for Traits:



Static Public Attributes

- static const bool ISPD_2012 = true
- static const double STD_THRESHOLD = 0.01
- static string ispd_contest_root
- static string ispd_contest_benchmark

7.43.1 Detailed Description

User configurable.

7.43.2 Member Data Documentation

7.43.2.1 const bool Traits::ISPD_2012 = true [static]

7.43.2.2 string Traits::ispd_contest_benchmark [static]

7.43.2.3 string Traits::ispd_contest_root [static]

7.43.2.4 const double Traits::STD_THRESHOLD = 0.01 [static]

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

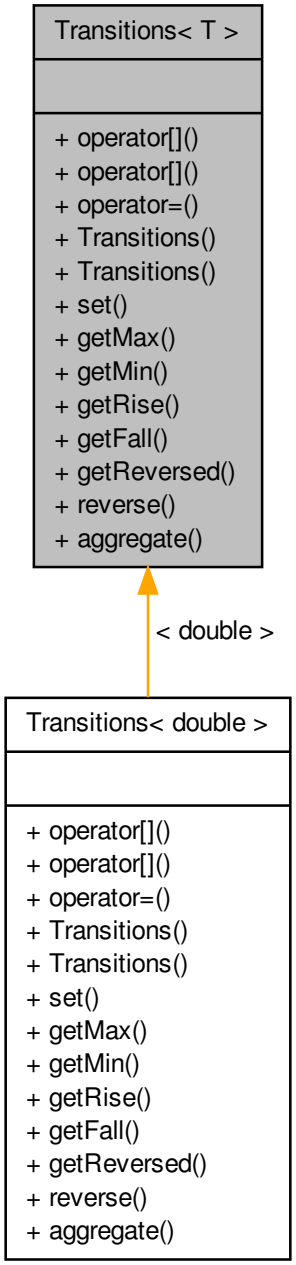
- TimingAnalysis/src/include/configuration.h

7.44 Transitions< T > Class Template Reference

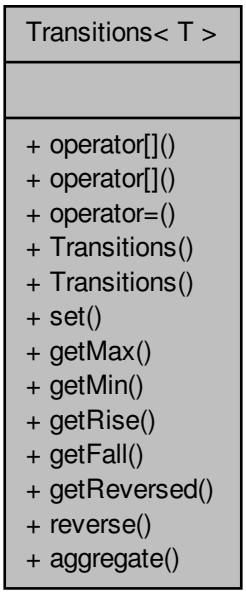
Template class which encapsulates any [rise,fall] twosome values and provides methods to work with it.

```
#include <transitions.h>
```

Inheritance diagram for Transitions< T >:



Collaboration diagram for Transitions< T >:



Public Member Functions

- T & operator[] (const EdgeType edgeType)

  Refefinition of [] operator. Returns value of edgeType.
- T operator[] (const EdgeType edgeType) const

  Refefinition of [] operator. Returns value of edgeType.
- Transitions & operator= (const Transitions &array)

  Refefinition of = operator. Sets caller transition time values to array time values and returns a pointer to it.
- Transitions (const T rise, const T fall)

  Transitions default constructor.
- Transitions ()

  Empty Transitions constructor.
- void set (const T rise, const T fall)

  Sets rise and fall time values.
- T getMax () const

  Returns max value between rise and fall time.
- T getMin () const

  Returns min value between rise and fall time.
- T getRise () const

  Returns rise time value.
- T getFall () const

  Returns fall time value.
- Transitions< T > getReversed () const

  Returns Transitions with exchanged value. Ex: If called by Transitions<rise, fall>, returns Transitions<fall, rise>
- void reverse ()

Exchanges rise and fall time values. Ex: Transitions<rise, fall> becomes Transitions<fall, rise>

- T aggregate () const

  Returns the sum of rise and fall time values.

Friends

- ostream & operator<< (ostream &out, const Transitions< T > array)

  Redefinition of << operator. Inserts formatted description including rise and fall times, in this order.

- Transitions< T > max (const Transitions< T > v0, const Transitions< T > v1)

  Definition of max operator. Returns Transition<T> with the higher values for rise and fall transition times.

- Transitions< T > min (const Transitions< T > v0, const Transitions< T > v1)

  Definition of min operator. Returns Transition<T> with the lower values for rise and fall transition times.

- Transitions< T > abs (const Transitions< T > v)

  Definition of abs operator. Returns Transition<T> with the absolute values in IEEE floating point representation of rise and fall transition times.

- Transitions< T > pow (const Transitions< T > v, const double exp)

  Definition of pow operator. Returns Transition<T> with rise and fall transition time values raised to exp.

- Transitions< T > sqrt (const Transitions< T > v)

  Definition of abs operator. Returns Transition<T> with the square root values of rise and fall transition times.

- Transitions< T > exp (const Transitions< T > v)

  Definition of abs operator. Returns Transition<T> with rise and fall transition time values in the form: <e^(rise time), e^(fall time)>, where 'e' is the Euler number e=2.718...

7.44.1 Detailed Description

template<typename T>class Transitions< T >

Template class which encapsulates any [rise,fall] twosome values and provides methods to work with it.

7.44.2 Constructor & Destructor Documentation

7.44.2.1 template<typename T> Transitions< T >::Transitions (const T *rise*, const T *fall*) [inline]

Transitions default constructor.

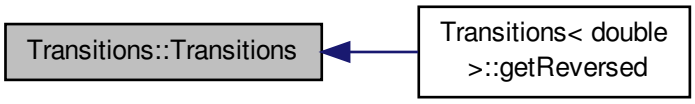
Parameters

const	T rise, const T fall

7.44.2.2 template<typename T> Transitions< T >::Transitions ( ) [inline]

Empty Transitions constructor.

Here is the caller graph for this function:



7.44.3 Member Function Documentation

7.44.3.1 template<typename T> T Transitions< T >::aggregate ( ) const [inline]

Returns the sum of rise and fall time values.

Returns

T

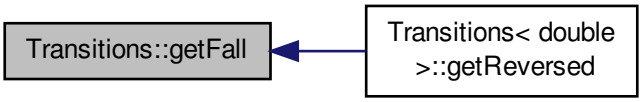
7.44.3.2 template<typename T> T Transitions< T >::getFall ( ) const [inline]

Returns fall time value.

Returns

T

Here is the caller graph for this function:



7.44.3.3 template<typename T> T Transitions< T >::getMax ( ) const [inline]

Returns max value between rise and fall time.

Returns

T

7.44.3.4 template<typename T> T Transitions< T >::getMin ( ) const [inline]

Returns min value between rise and fall time.

Returns

T

7.44.3.5 template<typename T> Transitions<T> Transitions< T >::getReversed ( ) const [inline]

Returns Transitions with exchanged value. Ex: If called by Transitions<rise, fall>, returns Transitions<fall, rise>

Returns

Transitions<T>

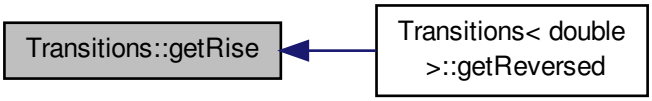
7.44.3.6 template<typename T> T Transitions< T >::getRise ( ) const [inline]

Returns rise time value.

Returns

T

Here is the caller graph for this function:



7.44.3.7 template<typename T> Transitions& Transitions< T >::operator= (const Transitions< T > & *array*) [inline]

Refefinition of = operator. Sets caller transition time values to array time values and returns a pointer to it.

Parameters

const	Transitions & array

7.44.3.8 template<typename T> T& Transitions< T >::operator[] (const EdgeType *edgeType*) [inline]

Refefinition of [] operator. Returns value of edgeType.

Parameters

const	EdgeType edgeType

7.44.3.9 template<typename T> T Transitions< T >::operator[] (const EdgeType *edgeType*) const [inline]

Refefinition of [] operator. Returns value of edgeType.

Parameters

const	EdgeType edgeType

7.44.3.10 template<typename T> void Transitions< T >::reverse ( ) [inline]

Exchanges rise and fall time values. Ex: Transitions<rise, fall> becomes Transitions<fall, rise>

Returns

T

7.44.3.11 template<typename T> void Transitions< T >::set (const T *rise,* const T *fall*) [inline]

Sets rise and fall time values.

Parameters

const	T rise, const T fall

Returns

void

7.44.4 Friends And Related Function Documentation

7.44.4.1 template<typename T> Transitions<T> abs (const Transitions< T > *v*) [friend]

Definition of abs operator. Returns Transition<T> with the absolute values in IEEE floating point representation of rise and fall transition times.

Parameters

const	Transitions<T> v

Returns

Transition<T>

7.44.4.2 template<typename T> Transitions<T> exp (const Transitions< T > *v*) [friend]

Definition of abs operator. Returns Transition<T> with rise and fall transition time values in the form: <e^(rise time), e^(fall time)>, where 'e' is the Euler number e=2.718...

Parameters

const	Transitions<T> v

Returns

Transition<T>

7.44.4.3 template<typename T> Transitions<T> max (const Transitions< T > *v0,* const Transitions< T > *v1*) [friend]

Definition of max operator. Returns Transition<T> with the higher values for rise and fall transition times.

Parameters

const	Transitions<T> v0

Returns

Transition<T>

7.44.4.4 template<typename T> Transitions<T> min (const Transitions< T > *v0,* const Transitions< T > *v1*) [friend]

Definition of min operator. Returns Transition<T> with the lower values for rise and fall transition times.

Parameters

const	Transitions<T> v0, const Transitions<T> v1

Returns

Transition<T>

7.44.4.5 template<typename T> ostream& operator<< (ostream & *out,* const Transitions< T > *array*) [friend]

Redefinition of << operator. Inserts formatted description including rise and fall times, in this order.

Parameters

ostream	&out, const Transitions<T> array

7.44.4.6 template<typename T> Transitions<T> pow (const Transitions< T > *v,* const double *exp*) [friend]

Definition of pow operator. Returns Transition<T> with rise and fall transition time values raised to exp.

Parameters

const	Transitions<T> v, const double exp

Returns

Transition<T>

7.44.4.7 template<typename T> Transitions<T> sqrt (const Transitions< T > *v*) [friend]

Definition of abs operator. Returns Transition<T> with the square root values of rise and fall transition times.

Parameters

const	Transitions<T> v

Returns

Transition<T>

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

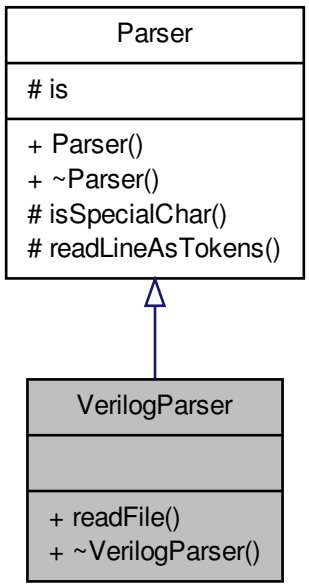
- TimingAnalysis/src/include/transitions.h

7.45 VerilogParser Class Reference

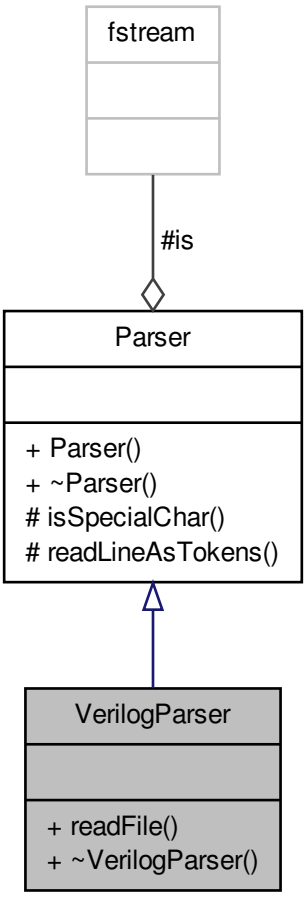
Inherits from Parser. Parse verilog files. Used to generate circuit netlists.

`#include <parser.h>`

Inheritance diagram for VerilogParser:



Collaboration diagram for VerilogParser:



Public Member Functions

- const Circuit_Netlist readFile (const string filename)

  Returns Circuit_Netlist object generated by information read from file passed as filename parameter.
- virtual ~VerilogParser ()

  Empty VerilogParser destructor.

Additional Inherited Members

7.45.1 Detailed Description

Inherits from Parser. Parse verilog files. Used to generate circuit netlists.

7.45.2 Constructor & Destructor Documentation

7.45.2.1 virtual VerilogParser::~VerilogParser ( ) [inline],[virtual]

Empty VerilogParser destructor.

7.45.3 Member Function Documentation

7.45.3.1 const Circuit_Netlist VerilogParser::readFile (const string *filename*)

Returns Circuit_Netlist object generated by information read from file passed as filename parameter.

Parameters

const	string filename

Returns

const Circuit_Netlist

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

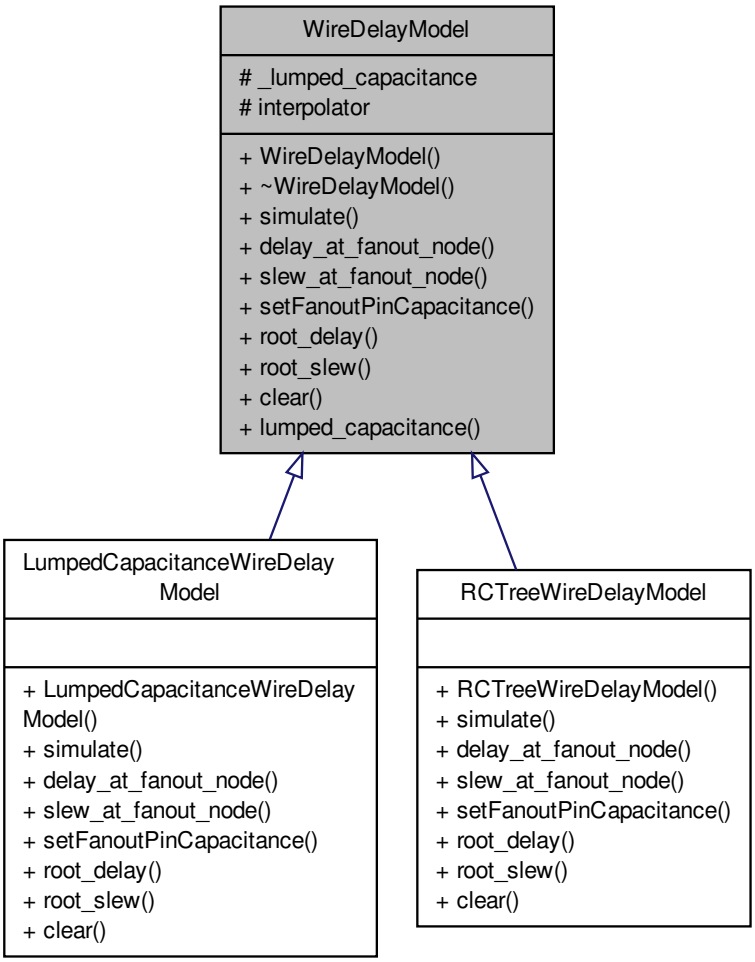
- TimingAnalysis/src/include/parser.h

7.46 WireDelayModel Class Reference

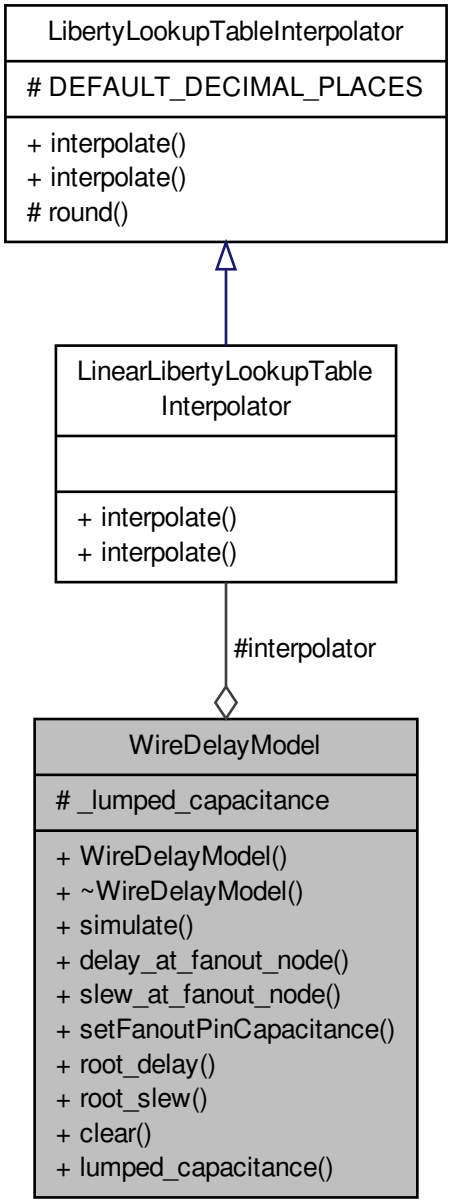
Describes parasitc elements present in wires in a circuit which cause significant delay and are taken into account in static timing analysis.

```
#include <wire_delay_model.h>
```

Inheritance diagram for WireDelayModel:



Collaboration diagram for WireDelayModel:



Public Member Functions

- WireDelayModel (const double &lumped_capacitance)

  WireDelayModel default constructor.
- virtual ~WireDelayModel ()

  Empty WideDelayModel destructor.
- virtual const Transitions< double > simulate (const LibertyCellInfo &cellInfo, const int input, const Transitions< double > slew, bool is_input_driver)=0
- virtual const Transitions< double > delay_at_fanout_node (const string fanout_node_name) const =0

Returns Transitions with rise and fall delay time values set to zero.

- virtual const Transitions< double > slew_at_fanout_node (const string fanout_node_name) const =0

  Returns Transitions with rise and fall slew time values set to zero.
- virtual void setFanoutPinCapacitance (const string fanoutNameAndPin, const double pinCapacitance)=0

  Increments lumped capacitance of the wire.
- virtual Transitions< double > root_delay (int arc_number)=0

  Virtual. Must be implemented by child class. Returns Transitions<double> representing delay of the wire.
- virtual Transitions< double > root_slew (int arc_number)=0

  Virtual. Must be implemented by child class. Returns Transitions<double> representing delay of the wire.
- virtual void clear ()=0

  Virtual. Must be implemented by child class. Clears attributes.
- double lumped_capacitance () const

  Returns lumped capacitance of the wire.

Protected Attributes

- double _lumped_capacitance

Static Protected Attributes

- static LinearLibertyLookupTableInterpolator interpolator

7.46.1 Detailed Description

Describes parasitc elements present in wires in a circuit which cause significant delay and are taken into account in static timing analysis.

7.46.2 Constructor & Destructor Documentation

7.46.2.1 WireDelayModel::WireDelayModel (const double & *lumped_capacitance*) [inline]

WireDelayModel default constructor.

Parameters

const	double & lumped_capacitance

7.46.2.2 virtual WireDelayModel::~WireDelayModel ( ) [inline], [virtual]

Empty WideDelayModel destructor.

7.46.3 Member Function Documentation

7.46.3.1 virtual void WireDelayModel::clear ( ) [pure virtual]

Virtual. Must be implemented by child class. Clears attributes.

Returns

void

Implemented in RCTreeWireDelayModel, and LumpedCapacitanceWireDelayModel.

7.46.3.2 virtual const Transitions<double> WireDelayModel::delay_at_fanout_node (const string *fanout_node_name*) const [pure virtual]

Returns Transitions with rise and fall delay time values set to zero.

Parameters

const	string fanout_node_name

Returns

Transitions<double>

Implemented in RCTreeWireDelayModel, and LumpedCapacitanceWireDelayModel.

7.46.3.3 double WireDelayModel::lumped_capacitance ( ) const

Returns lumped capacitance of the wire.

Returns

double

7.46.3.4 virtual Transitions<double> WireDelayModel::root_delay (int *arc_number*) [pure virtual]

Virtual. Must be implemented by child class. Returns Transitions<double> representing delay of the wire.

Parameters

int	arc_number

Returns

Transitions<double>

Implemented in RCTreeWireDelayModel, and LumpedCapacitanceWireDelayModel.

7.46.3.5 virtual Transitions<double> WireDelayModel::root_slew (int *arc_number*) [pure virtual]

Virtual. Must be implemented by child class. Returns Transitions<double> representing delay of the wire.

Parameters

int	arc_number

Returns

Transitions<double>

Implemented in RCTreeWireDelayModel, and LumpedCapacitanceWireDelayModel.

7.46.3.6 virtual void WireDelayModel::setFanoutPinCapacitance (const string *fanoutNameAndPin,* const double *pinCapacitance*) [pure virtual]

Increments lumped capacitance of the wire.

Parameters

const	string fanoutNameAndPin, const double capacitance

Returns

void

Implemented in RCTreeWireDelayModel, and LumpedCapacitanceWireDelayModel.

7.46.3.7 virtual const Transitions<double> WireDelayModel::simulate (const LibertyCellInfo & *cellInfo,* const int *input,* const Transitions< double > *slew,* bool *is_input_driver*) [pure virtual]

Parameters

const	LibertyCellInfo & cellInfo, const int input, const Transitions<double> slew, bool is_input_driver

Returns

Transitions<double>

Implemented in RCTreeWireDelayModel, and LumpedCapacitanceWireDelayModel.

7.46.3.8 virtual const Transitions<double> WireDelayModel::slew_at_fanout_node (const string *fanout_node_name*) const [pure virtual]

Returns Transitions with rise and fall slew time values set to zero.

Parameters

const	string fanout_node_name

Returns

Transitions<double>

Implemented in RCTreeWireDelayModel, and LumpedCapacitanceWireDelayModel.

7.46.4 Member Data Documentation

7.46.4.1 double WireDelayModel::_lumped_capacitance [protected]

7.46.4.2 LinearLibertyLookupTableInterpolator WireDelayModel::interpolator [static],[protected]

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

- TimingAnalysis/src/include/wire_delay_model.h

Chapter 8

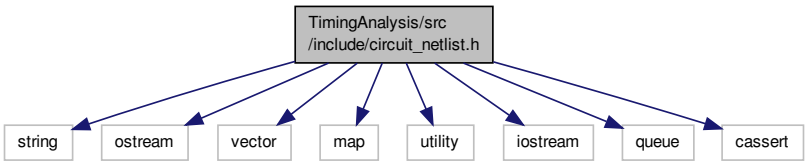
File Documentation

8.1 TimingAnalysis/doc/Index.dox File Reference

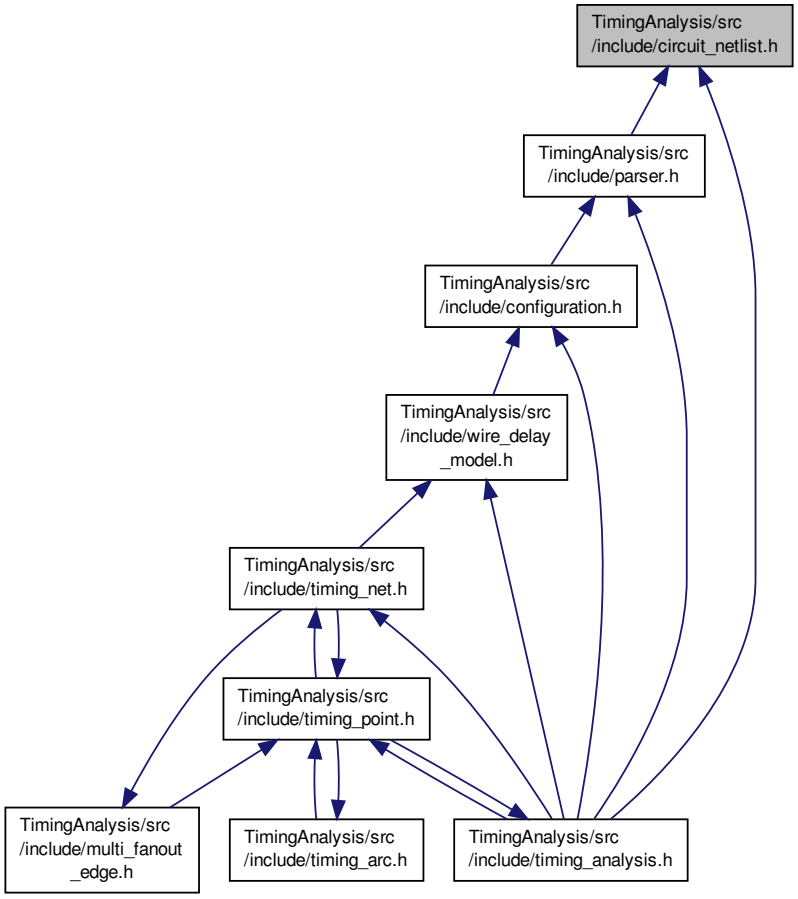
8.2 TimingAnalysis/src/include/circuit_netlist.h File Reference

```
#include <string>
#include <ostream>
#include <vector>
#include <map>
#include <utility>
#include <iostream>
#include <queue>
#include <cassert>
```

Include dependency graph for circuit_netlist.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Circuit_Netlist

  File which describe the set of elements of the circuit. Generated by the VerilogParser class.
- struct Circuit_Netlist::Logic_Gate

  Struct which represents a Logic_Gate.
- struct Circuit_Netlist::Sink

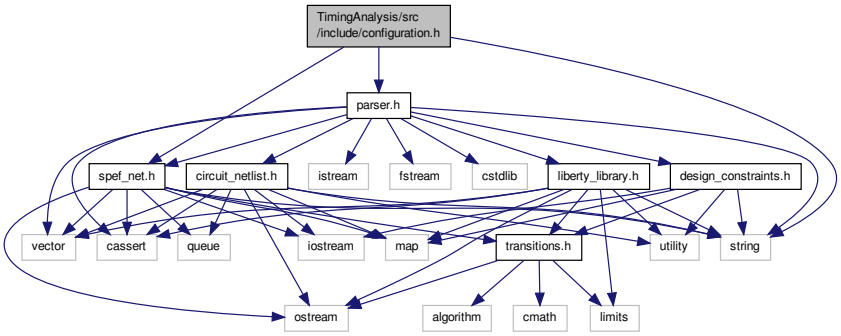
  Struct which represents a Sink. A sink is a vertex which has input Edge's, but no output Edge's (outdegree zero).
- struct Circuit_Netlist::Net

  Struct which represents a Net, which connect two or more circuit elements together.

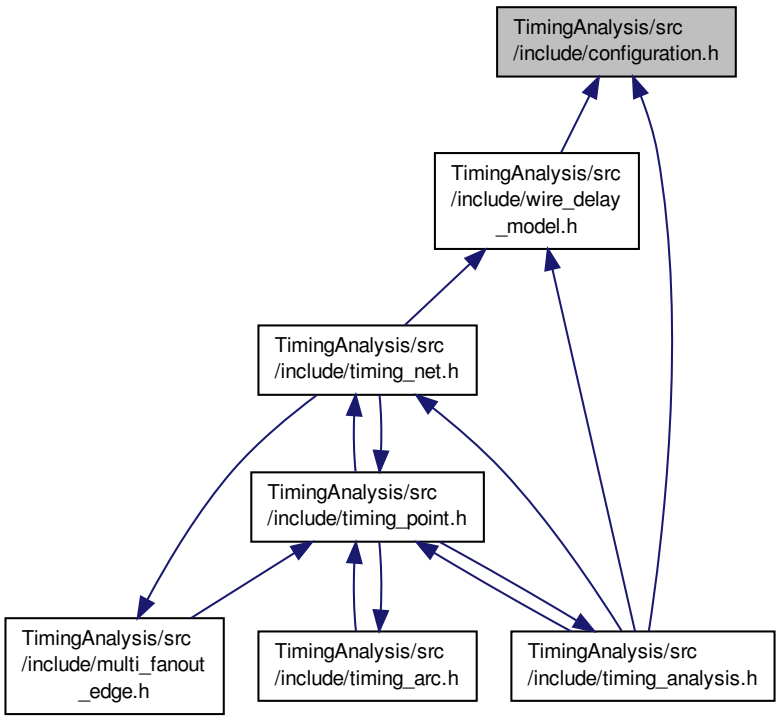
8.3 TimingAnalysis/src/include/configuration.h File Reference

```
#include "spef_net.h"
#include "parser.h"
#include <string>
```

Include dependency graph for configuration.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct IF< cond, ThenType, ElseType >

  This is a Configuration File To switch between ISPD2012 (Lumped Capacitance Wire Delay Model) or ISPD2013 (Distributed RC Wire Delay Model) SPEF format.
- struct IF< false, ThenType, ElseType >

  Metaprogrammed IF.
- class Traits

User configurable.

Typedefs

- typedef IF< Traits::ISPD_2012, SpefParserISPD2012, SpefParserISPD2013 >::RET SpefParser
- typedef IF< Traits::ISPD_2012, SpefNetISPD2012, SpefNetISPD2013 >::RET SpefNet
- typedef IF< Traits::ISPD_2012, Parasitics2012, Parasitics2013 > ::RET Parasitics

8.3.1 Typedef Documentation

8.3.1.1 typedef IF<Traits::ISPD_2012, Parasitics2012, Parasitics2013>::RET Parasitics

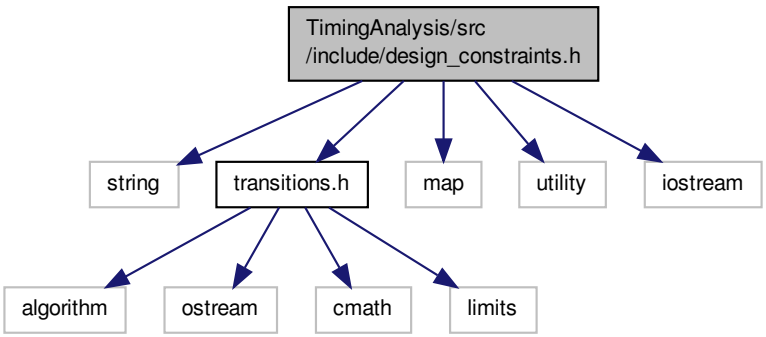
8.3.1.2 typedef IF<Traits::ISPD_2012, SpefNetISPD2012, SpefNetISPD2013>::RET SpefNet

8.3.1.3 typedef IF<Traits::ISPD_2012, SpefParserISPD2012, SpefParserISPD2013>::RET SpefParser

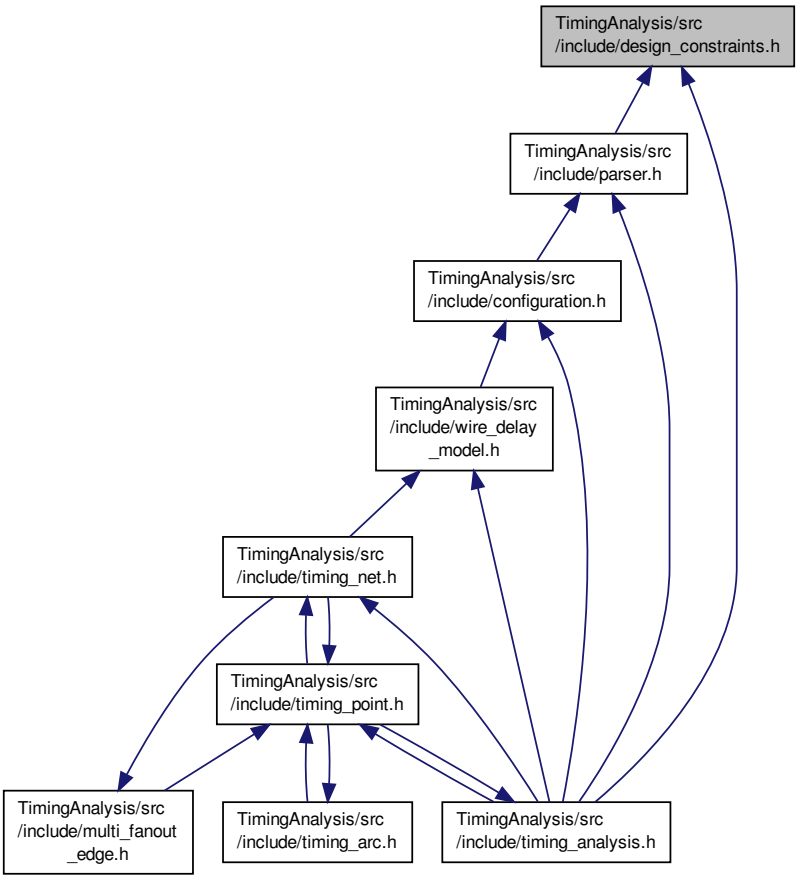
8.4 TimingAnalysis/src/include/design_constraints.h File Reference

```
#include <string>
#include "transitions.h"
#include <map>
#include <utility>
#include <iostream>
```

Include dependency graph for design_constraints.h:



This graph shows which files directly or indirectly include this file:



Classes

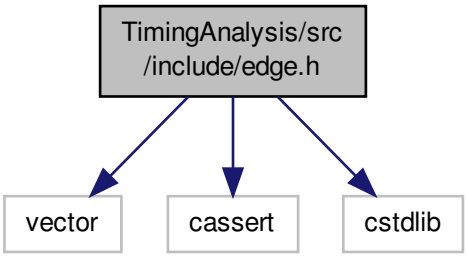
- class Design_Constraints

  Represents the design constraints of the case in study. Generated by the SDCParser class.

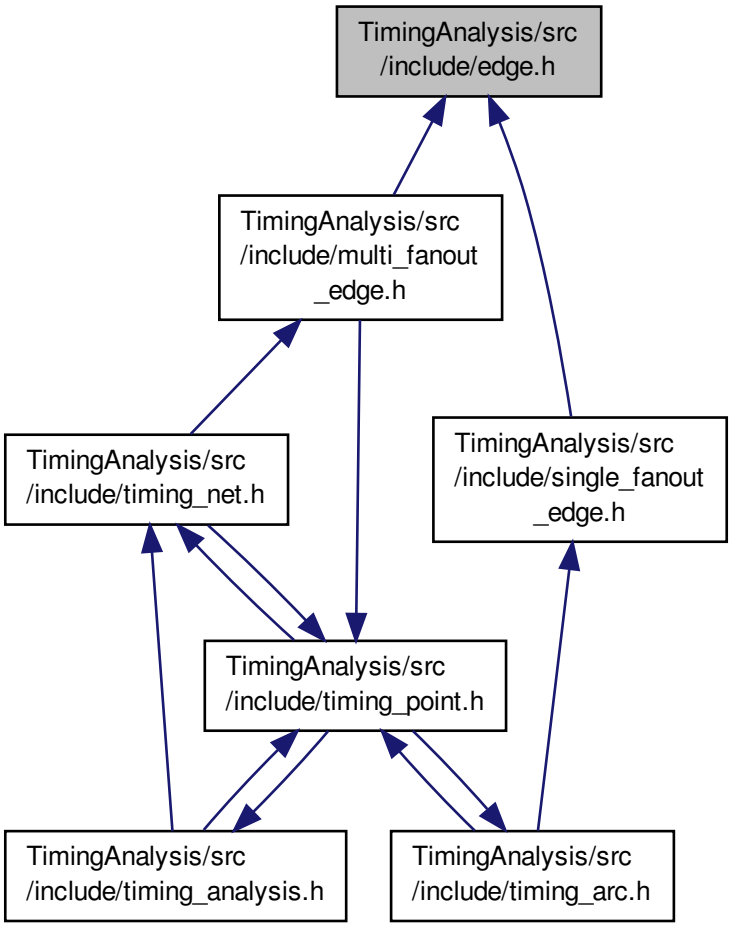
8.5 TimingAnalysis/src/include/edge.h File Reference

```
#include <vector>
#include <cassert>
#include <cstdlib>
```

Include dependency graph for edge.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timing_Analysis::Edge< T >

  Template class which represents an edge of the graph model. An edge in graph theory is a connection between vertices.

Namespaces

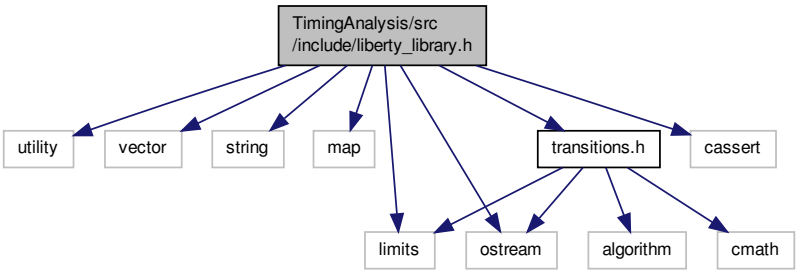
- Timing_Analysis

  Namespace which contains classes relative to timing analysis graph model.

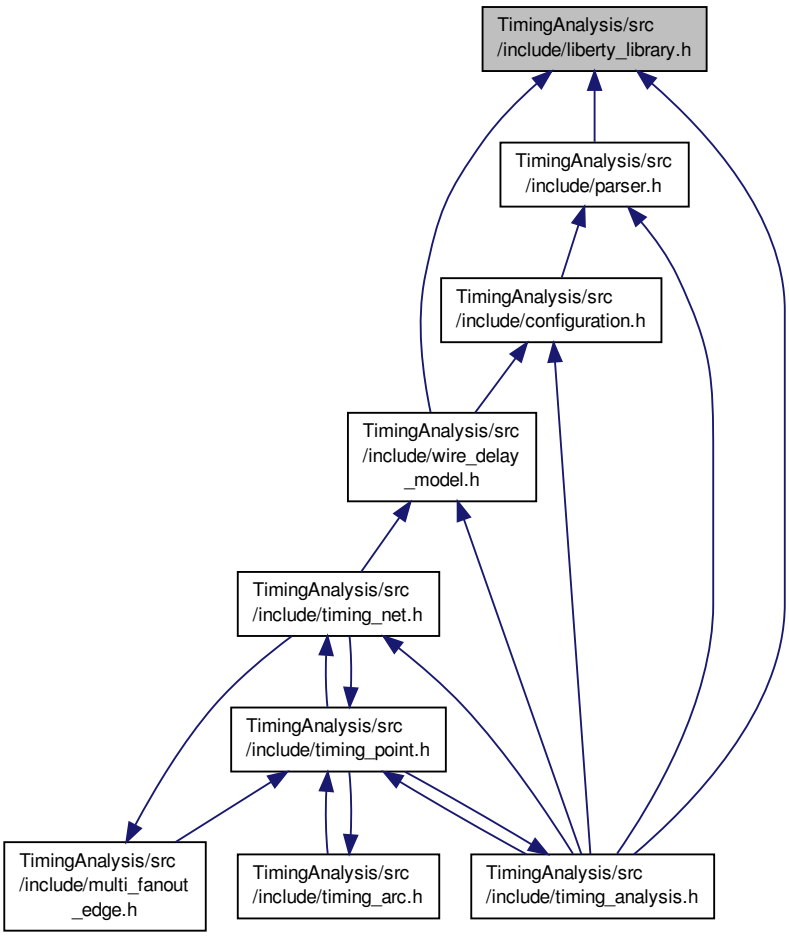
8.6 TimingAnalysis/src/include/liberty_library.h File Reference

```
#include <utility>
#include <vector>
#include <string>
#include <map>
#include <limits>
#include <ostream>
#include <cassert>
#include "transitions.h"
```

Include dependency graph for liberty_library.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct LibertyLookupTable

  Struct representing a lookup table to store delay or slew functions.
- struct LibertyTimingInfo

  Struct which has informations about timing.
- struct LibertyPinInfo

  Struct which has informations about a pin.
- struct LibertyCellInfo

  Struct which has informations about a cell.
- class LibertyLibrary

  Standard cell Library is a collection of logic functions info. Contains details about implementations of logic functions, within its layout, parasitic models, Verilog models, and more. Generated by the LibertyParser class.
- class LibertyLookupTableInterpolator

  Interpolation calculator.
- class LinearLibertyLookupTableInterpolator

  Linear interpolation calculator.

Enumerations

- enum Unateness { NEGATIVE_UNATE, POSITIVE_UNATE, NON_UNATE }

Functions

- ostream & operator<< (ostream &os, LibertyLookupTable &lut)

  Redefinition of <<s operator. Included for completeness, not implemented.
- ostream & operator<< (ostream &os, LibertyTimingInfo &timing)

  Redefinition of << operator. Included for completeness, not implemented.
- ostream & operator<< (ostream &os, LibertyPinInfo &pin)

  Redefinition of << operator. Included for completeness, not implemented.
- ostream & operator<< (ostream &os, LibertyCellInfo &cell)

  Redefinition of << operator. Included for completeness, not implemented.

8.6.1 Enumeration Type Documentation

8.6.1.1 enum Unateness

Enumerator

NEGATIVE_UNATE

POSITIVE_UNATE

NON_UNATE

8.6.2 Function Documentation

8.6.2.1 ostream& operator<< (ostream & *os*, LibertyLookupTable & *lut*)

Redefinition of <<s operator. Included for completeness, not implemented.

Parameters

ostream&	os, LibertyLookupTable& lut

8.6.2.2 ostream& operator<< (ostream & *os*, LibertyTimingInfo & *timing*)

Redefinition of << operator. Included for completeness, not implemented.

Parameters

ostream&	os, LibertyTimingInfo& timing

8.6.2.3 ostream& operator<< (ostream & *os*, LibertyPinInfo & *pin*)

Redefinition of << operator. Included for completeness, not implemented.

Parameters

ostream&	os, LibertyPinInfo& pin

8.6.2.4 ostream& operator<< (ostream & *os*, LibertyCellInfo & *cell*)

Redefinition of << operator. Included for completeness, not implemented.

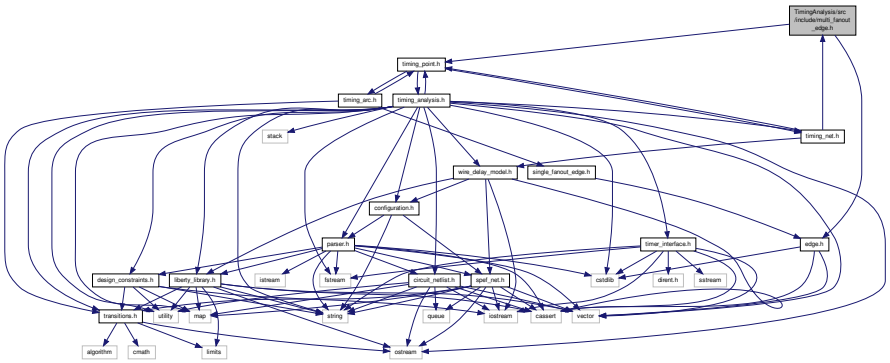
Parameters

ostream&	os, LibertyCellInfo& cell

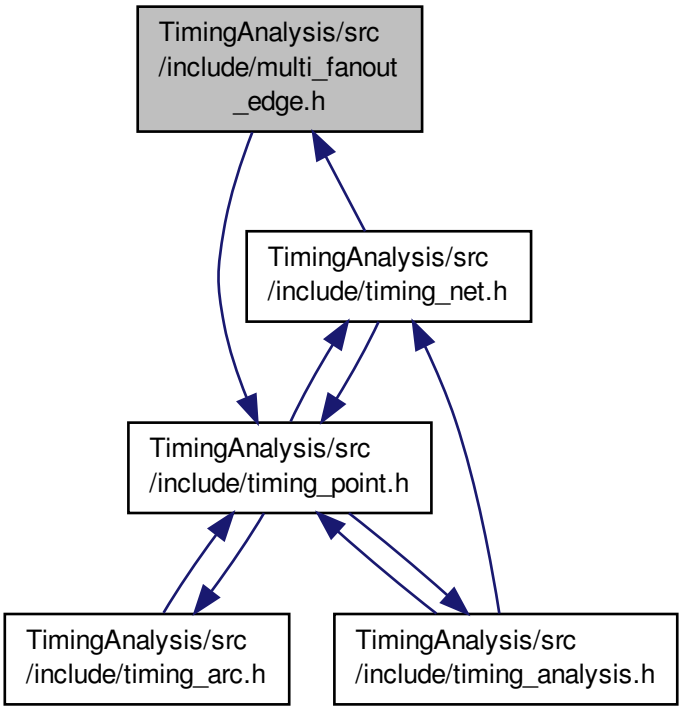
8.7 TimingAnalysis/src/include/multi_fanout_edge.h File Reference

```
#include "edge.h"
#include "timing_point.h"
```

Include dependency graph for multi_fanout_edge.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timing_Analysis::Multi_Fanout_Edge< T >

  Inherits from Edge. An multi fanout edge is a group of edges which connects a vertex to more than one vertexes.

Namespaces

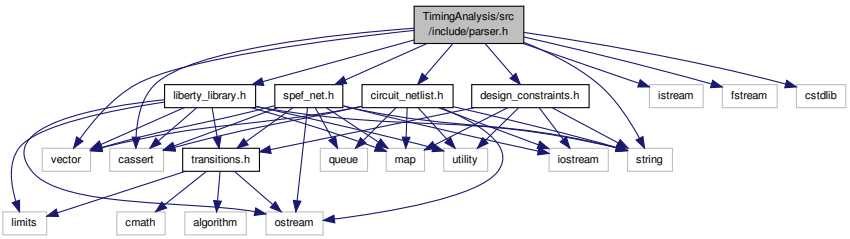
- Timing_Analysis

  Namespace which contains classes relative to timing analysis graph model.

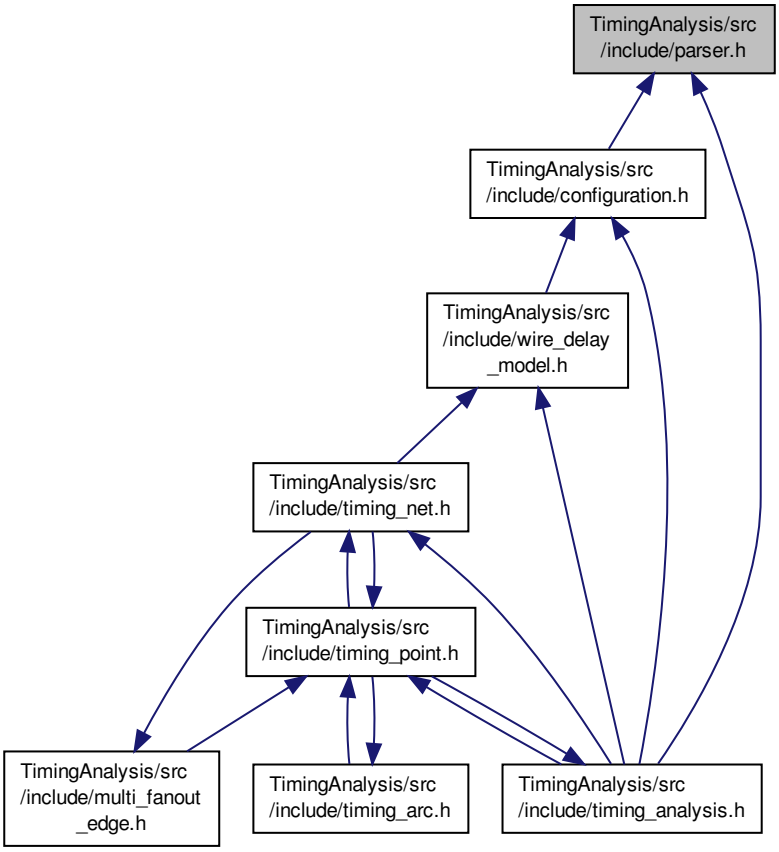
8.8 TimingAnalysis/src/include/parser.h File Reference

```
#include <string>
#include <vector>
#include <istream>
#include <fstream>
#include <cassert>
#include <cstdlib>
#include "circuit_netlist.h"
#include "liberty_library.h"
#include "spef_net.h"
#include "design_constraints.h"
```

Include dependency graph for parser.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Parser

  Superclass. A Parser is a program which reads an input and creates a data structure. In this case, the Parser will read a string, validate it and build essential files for use in the program using information read from the string.
- class Prime_Time_Output_Parser

  Inherits from Parser. Parse the Prime Time output. Used to generate a Prime_Time_Output object for later use.
- struct Prime_Time_Output_Parser::Pin_Timing

  Struct containing the timing values of a pin.
- struct Prime_Time_Output_Parser::Port_Timing

  Struct containing the timing values of a port.
- class Prime_Time_Output_Parser::Prime_Time_Output

  Represents Prime Time output. Provides access methods to it.
- class VerilogParser

  Inherits from Parser. Parse verilog files. Used to generate circuit netlists.
- class LibertyParser

  Inherits from Parser. Used to generate a LibertyLibrary object from a file.
- class SpefParserISPD2013

  Inherits from Parser. Used to generate SpefNetISPD2013 file from an input.
- class SpefParserISPD2012

Inherits from Parser. Used to generate SpefNetISPD2012 files from an input.

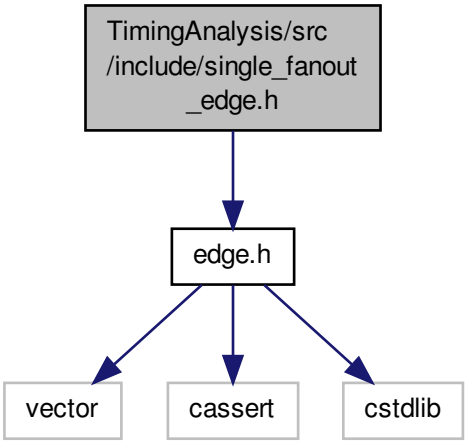
- class SDCParser

Inherits from Parser. SDC stands for Synopsys Design Constraints. Used to generate a Design_Constraints object from file.

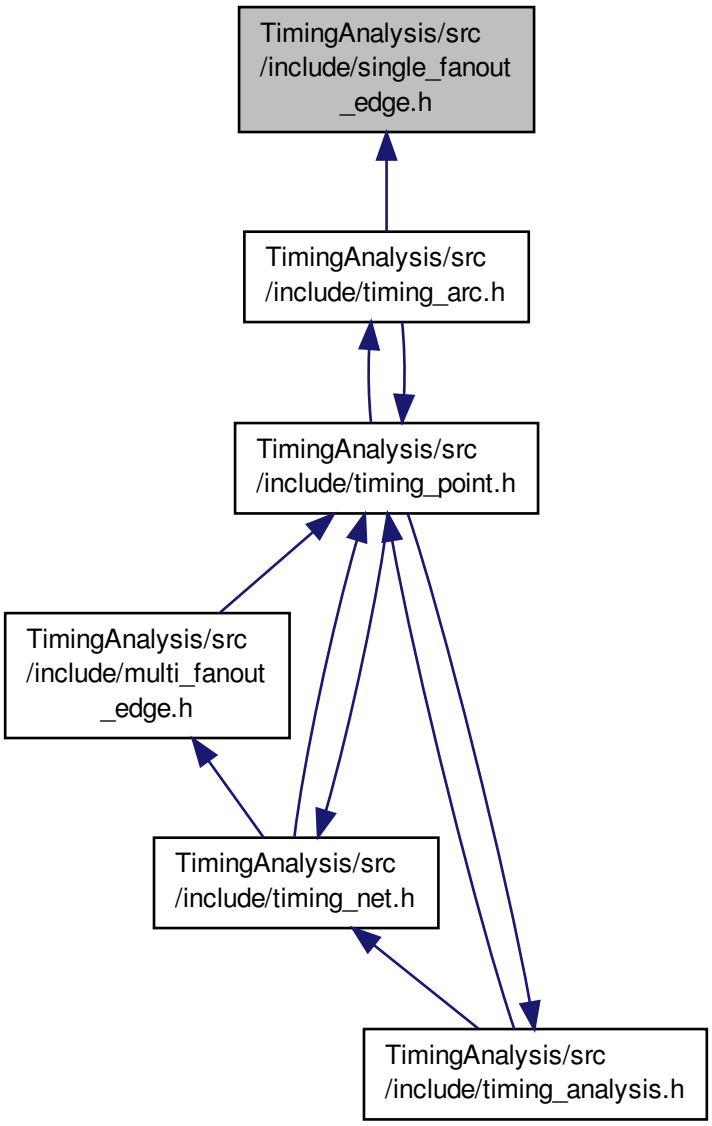
8.9 TimingAnalysis/src/include/single_fanout_edge.h File Reference

```
#include "edge.h"
```

Include dependency graph for single_fanout_edge.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timing_Analysis::Single_Fanout_Edge< T >

  Inherits from Edge. A single fanout edge is an edge which connects two vertex to each other.

Namespaces

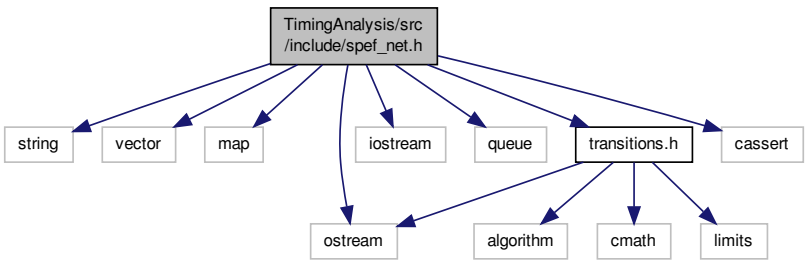
- Timing_Analysis

  Namespace which contains classes relative to timing analysis graph model.

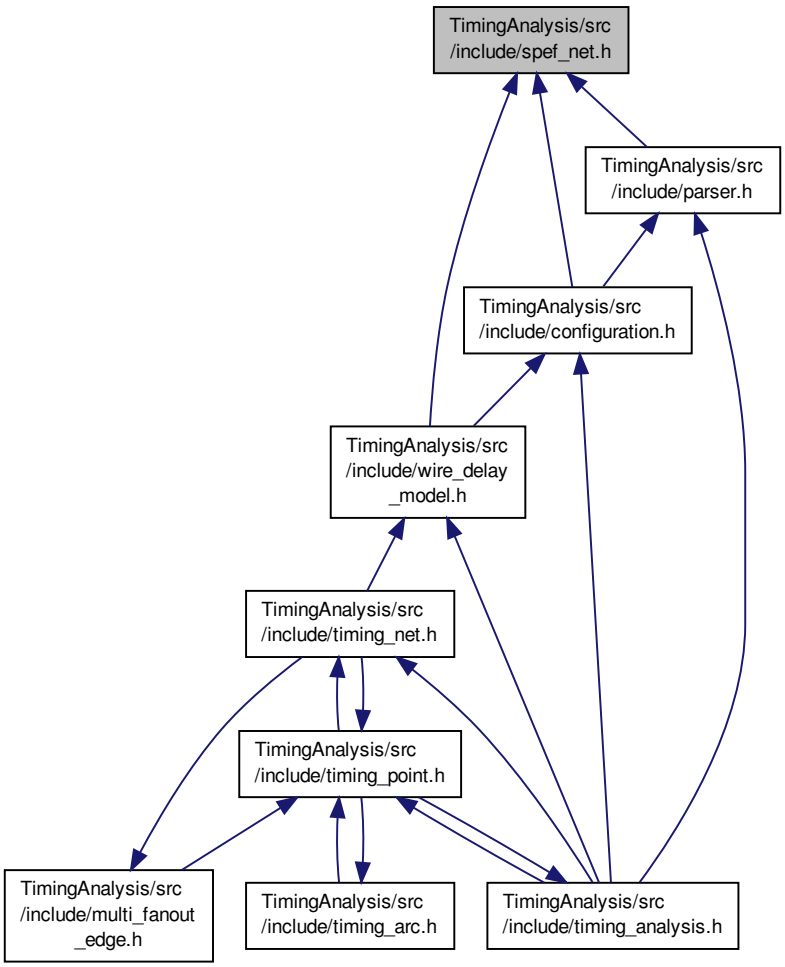
8.10 TimingAnalysis/src/include/spef_net.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <ostream>
#include <iostream>
#include <queue>
#include "transitions.h"
#include <cassert>
```

Include dependency graph for spef_net.h:



This graph shows which files directly or indirectly include this file:



Classes

- class SpefNetISPD2012

  SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2012.

- class SpefNetISPD2013

  SPEF stands for Standard Parasitic Exchange Format. Describes the set of parasitic elements of the wires in a circuit, within its descriptions and values. Generated by SpefParserISPD2013.

- struct SpefNetISPD2013::Resistor

  Struct which represents a resistor.

- struct SpefNetISPD2013::Capacitor

  Struct which represents a capacitor.

- struct SpefNetISPD2013::Node

  Struct which represents a node of the circuit. A node is a point of the circuit where two or more elements meet.

Typedefs

- typedef map< string, SpefNetISPD2012 > Parasitics2012

  Name redefinition: from map<string, SpefNetISPD2012> to SpefNetISPD2012>

- typedef map< string, SpefNetISPD2013 > Parasitics2013

  Name redefinition: from map<string, SpefNetISPD2013> to SpefNetISPD2013>

8.10.1 Typedef Documentation

8.10.1.1 typedef map<string, SpefNetISPD2012> Parasitics2012

Name redefinition: from map<string, SpefNetISPD2012> to SpefNetISPD2012>

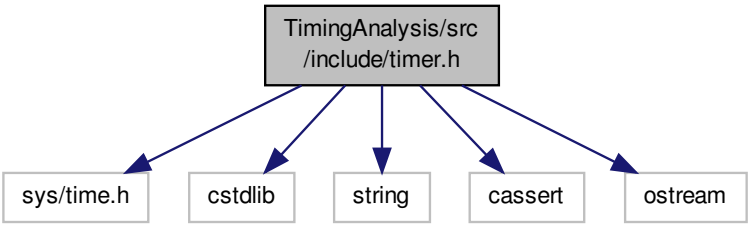
8.10.1.2 typedef map<string, SpefNetISPD2013> Parasitics2013

Name redefinition: from map<string, SpefNetISPD2013> to SpefNetISPD2013>

8.11 TimingAnalysis/src/include/timer.h File Reference

```
#include <sys/time.h>
#include <cstdlib>
#include <string>
#include <cassert>
#include <ostream>
```

Include dependency graph for timer.h:



Classes

- class Timer

  Represents a Timer.

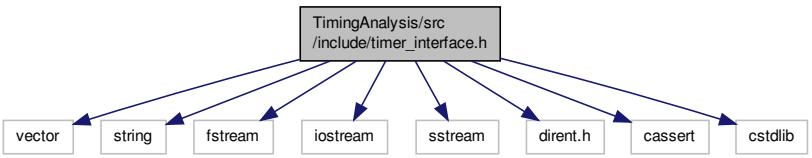
- class Timer::Result

  Represents the result from the Timer class.

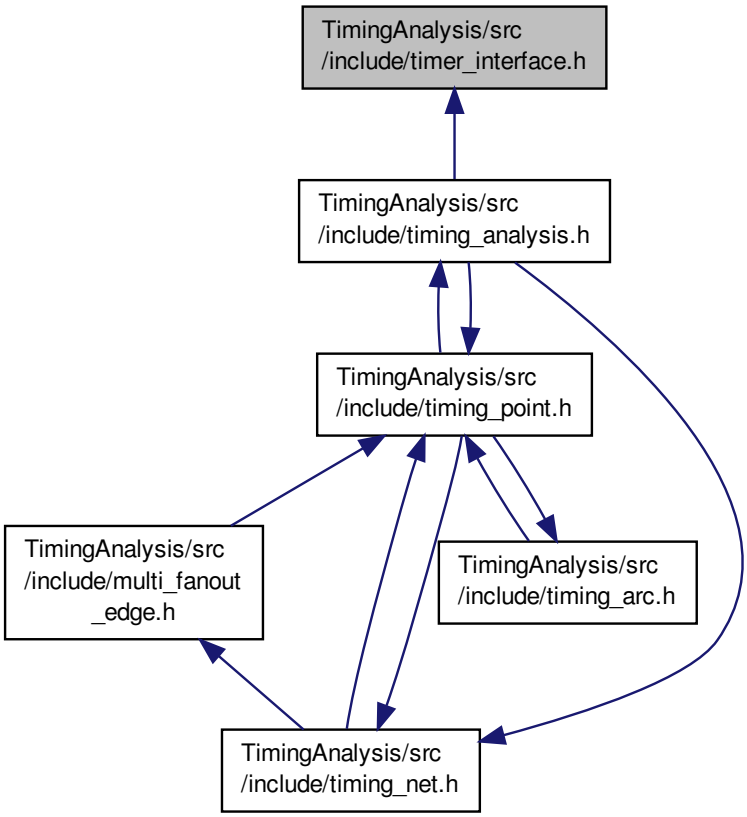
8.12 TimingAnalysis/src/include/timer_interface.h File Reference

```
#include <vector>
#include <string>
#include <fstream>
#include <iostream>
#include <sstream>
#include <dirent.h>
#include <cassert>
#include <cstdlib>
```

Include dependency graph for timer_interface.h:



This graph shows which files directly or indirectly include this file:



Classes

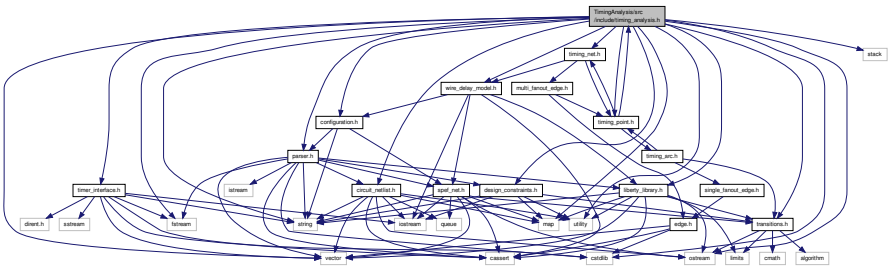
- class TimerInterface

This class contains functions for the timing analysis interface. To use any function belonging to this class, call TimerInterface::<function_name>(<argument_list>);.

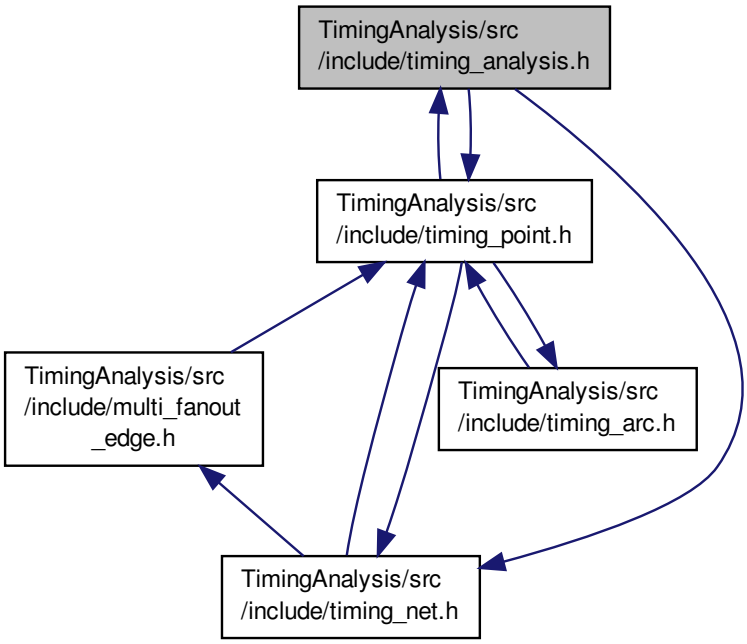
8.13 TimingAnalysis/src/include/timing_analysis.h File Reference

```
#include <vector>
#include <string>
#include <ostream>
#include <map>
#include <utility>
#include <fstream>
#include <stack>
#include <cstdlib>
#include "timer_interface.h"
#include "transitions.h"
#include "circuit_netlist.h"
#include "wire_delay_model.h"
#include "liberty_library.h"
#include "design_constraints.h"
#include "configuration.h"
#include "parser.h"
#include "timing_net.h"
#include "timing_point.h"
```

Include dependency graph for timing_analysis.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timing_Analysis::Option

  Refers to the implementation option of a logic gate.

- class Timing_Analysis::Timing_Analysis

  Class which performs the timing analysis.

Namespaces

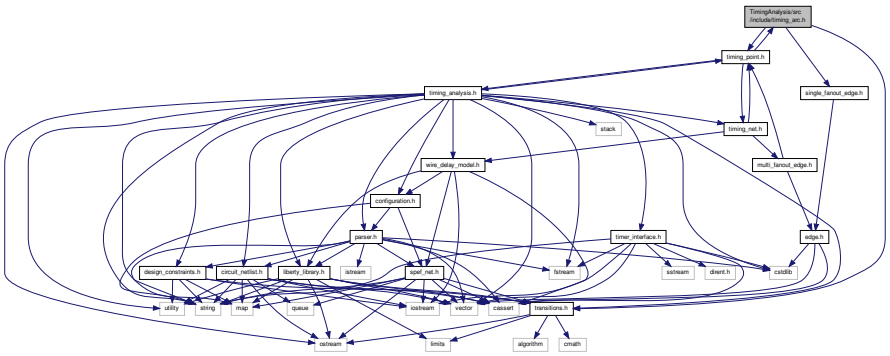
- Timing_Analysis

  Namespace which contains classes relative to timing analysis graph model.

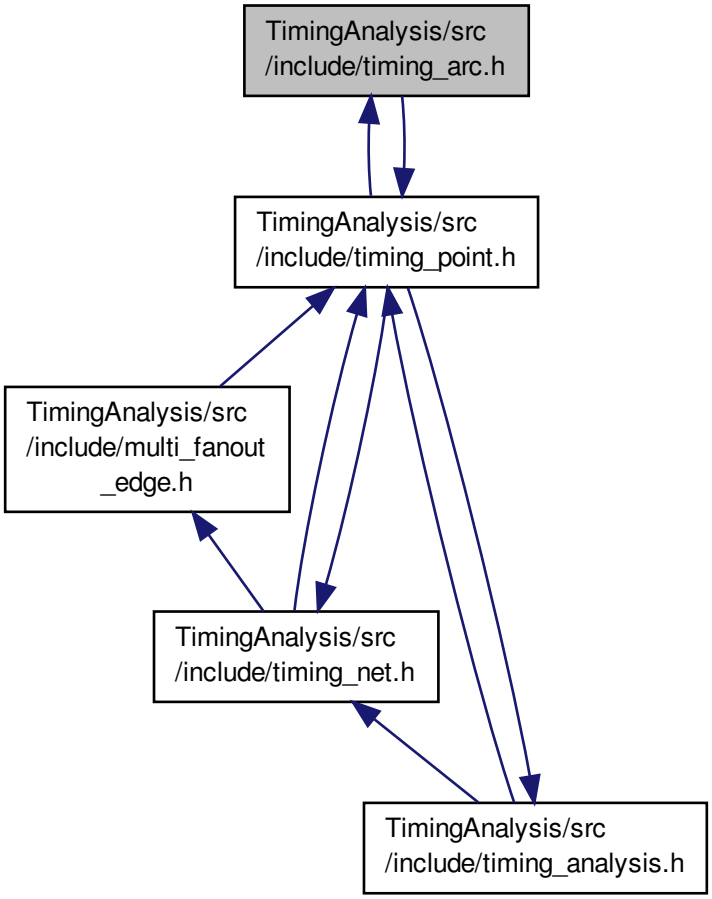
8.14 TimingAnalysis/src/include/timing_arc.h File Reference

```
#include "single_fanout_edge.h"
#include "transitions.h"
#include "timing_point.h"
```

Include dependency graph for timing_arc.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timing_Analysis::Timing_Arc

  Describes a Timing_Arc of the timing graph model. Vertex which connects one Timing_Point to another timing point only.

Namespaces

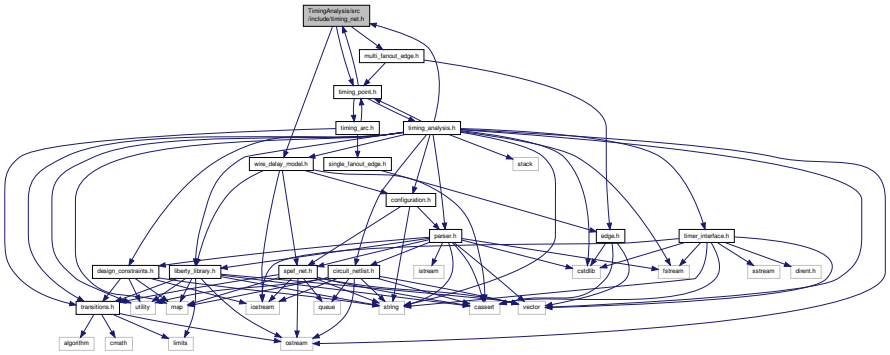
- Timing_Analysis

  Namespace which contains classes relative to timing analysis graph model.

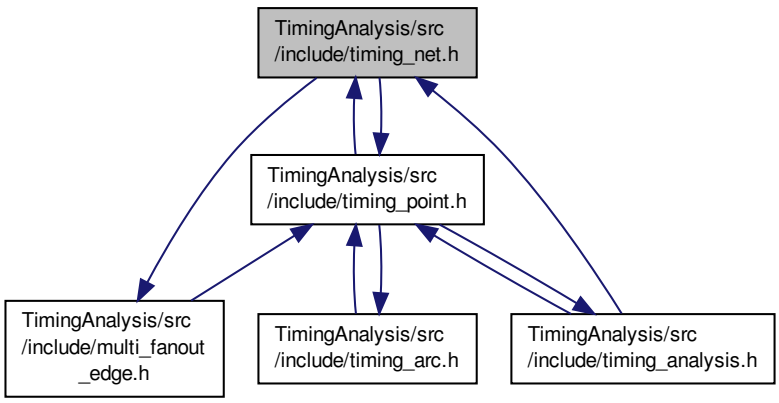
8.15 TimingAnalysis/src/include/timing_net.h File Reference

```
#include "wire_delay_model.h"
#include "multi_fanout_edge.h"
#include "timing_point.h"
```

Include dependency graph for timing_net.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timing_Analysis::Timing_Net

  Describes a Timing_Net of the timing graph model. Vertex which connects one Timing_Point to multiple Timing_Point's.

Namespaces

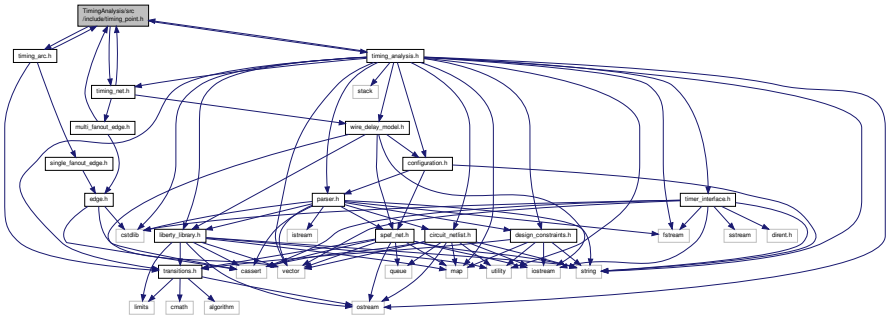
- Timing_Analysis

  Namespace which contains classes relative to timing analysis graph model.

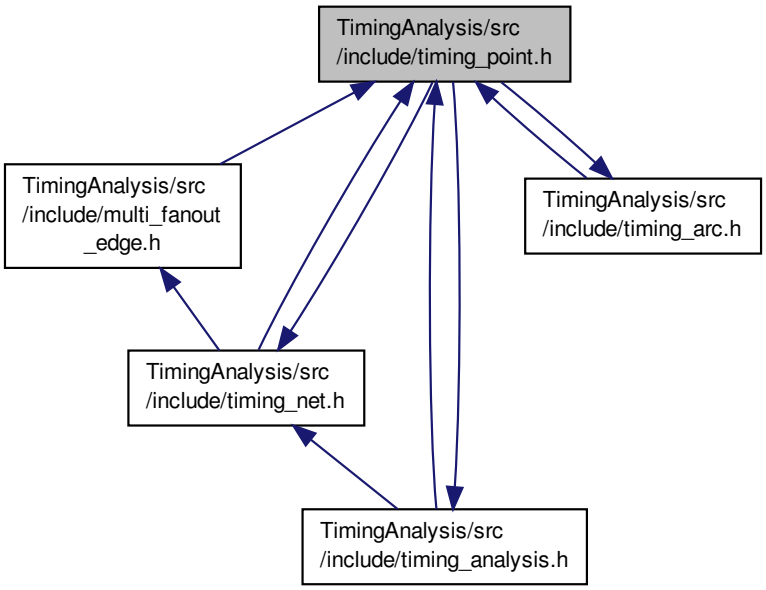
8.16 TimingAnalysis/src/include/timing_point.h File Reference

```
#include "timing_net.h"
#include "timing_arc.h"
#include "timing_analysis.h"
```

Include dependency graph for timing_point.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Timing_Analysis::Timing_Point

  Describes a Timing_Point of the timing graph model.

Namespaces

- Timing_Analysis

  Namespace which contains classes relative to timing analysis graph model.

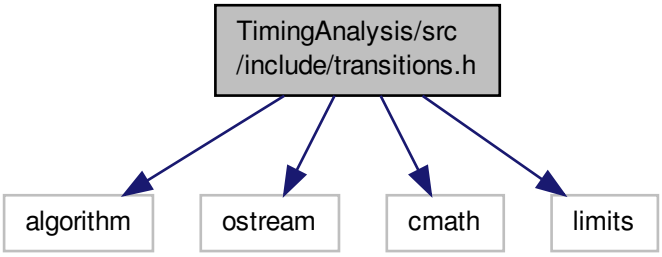
Enumerations

- enum Timing_Analysis::Timing_Point_Type {
  Timing_Analysis::INPUT, Timing_Analysis::OUTPUT, Timing_Analysis::PI_INPUT, Timing_Analysis::REGISTER_INPUT,
  Timing_Analysis::PI, Timing_Analysis::PO }

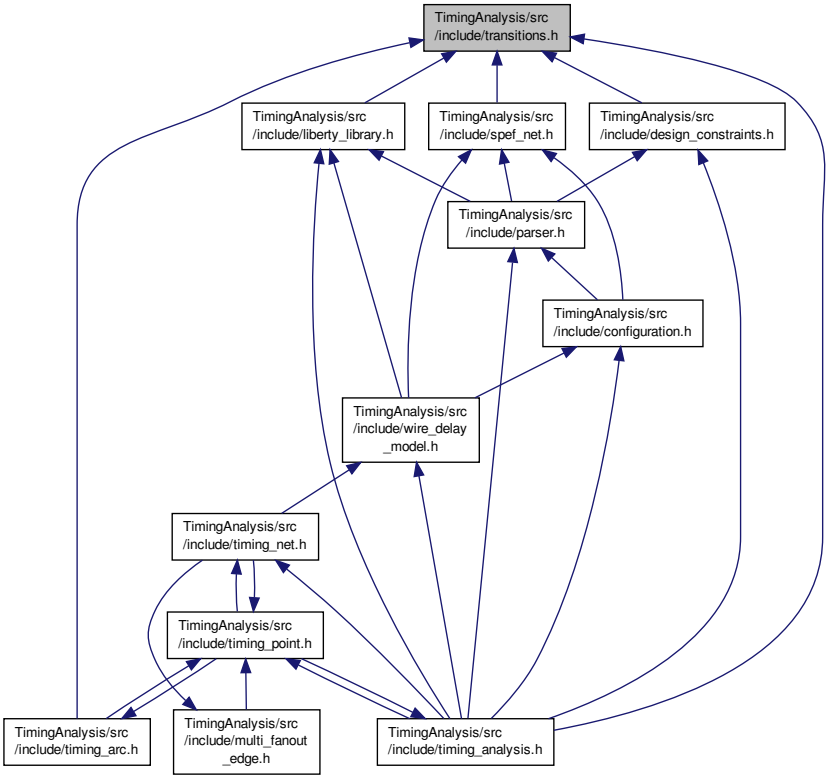
8.17 TimingAnalysis/src/include/transitions.h File Reference

```
#include <algorithm>
#include <ostream>
#include <cmath>
#include <limits>
```

Include dependency graph for transitions.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Transitions< T >

  Template class which encapsulates any [rise,fall] twosome values and provides methods to work with it.
- class std::numeric_limits< Transitions< double > >

  Class used to make easier to work with double variable numeric limits.

Namespaces

- std

Macros

- #define MAKE_SELF_OPERATOR(OP)
- #define MAKE_OPERATOR(OP)

Enumerations

- enum EdgeType { RISE = 0, FALL = 1 }

8.17.1 Macro Definition Documentation

8.17.1.1 #define MAKE_OPERATOR(*OP*)

Value:

```
friend Transitions<T> operator OP ( const Transitions<T> v0, const
      Transitions<T> v1 ) { return Transitions<T>(v0[RISE] OP v1[
      RISE], v0[FALL] OP v1[FALL]); } \
friend Transitions<T> operator OP ( const T            v0, const
      Transitions<T> v1 ) { return Transitions<T>(v0       OP v1[
      RISE], v0       OP v1[FALL]); } \
friend Transitions<T> operator OP ( const Transitions<T> v0, const T
       v1 ) { return Transitions<T>(v0[RISE] OP v1      , v0[FALL] OP v1      ); }
```

8.17.1.2 #define MAKE_SELF_OPERATOR(*OP*)

Value:

```
friend void operator OP ( Transitions<T> &v0, const Transitions<T> v1 ) { v0[
      RISE] OP v1[RISE], v0[FALL] OP v1[FALL]; } \
friend void operator OP ( Transitions<T> &v0, const T            v1 ) { v0[
      RISE] OP v1; v0[FALL] OP v1; }
```

8.17.2 Enumeration Type Documentation

8.17.2.1 enum EdgeType

Enumerator

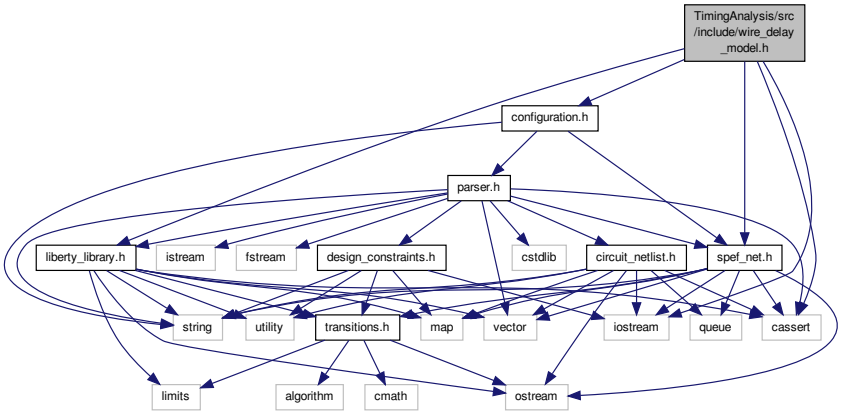
RISE

FALL

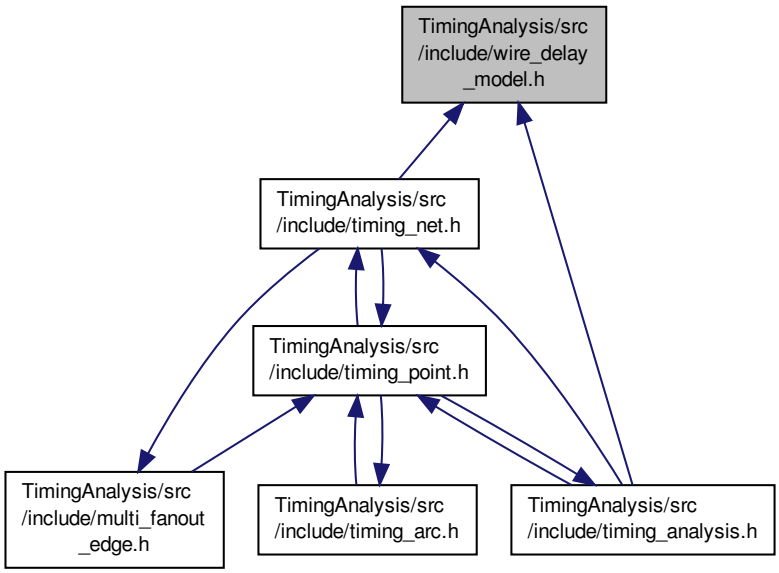
8.18 TimingAnalysis/src/include/wire_delay_model.h File Reference

```
#include <iostream>
#include "spef_net.h"
#include "liberty_library.h"
#include "configuration.h"
#include <cassert>
```

Include dependency graph for wire_delay_model.h:



This graph shows which files directly or indirectly include this file:



Classes

- class WireDelayModel

  Describes parasitc elements present in wires in a circuit which cause significant delay and are taken into account in static timing analysis.

- class LumpedCapacitanceWireDelayModel

  This model describes the parasitic capacitances distributed along a wire as a single circuit element. Does not take resistors into account.

- class RCTreeWireDelayModel

This model describes the parasitic elements(capacitors and resistors) distributed along a wire as a single circuit element.

Index

~Circuit_Netlist
  Circuit_Netlist, 19
~LibertyLibrary
  LibertyLibrary, 37
~Parser
  Parser, 69
~SpefNetISPD2013
  SpefNetISPD2013, 95
~Timer
  Timer, 102
~Timing_Analysis
  Timing_Analysis::Timing_Analysis, 106
~Timing_Arc
  Timing_Analysis::Timing_Arc, 113
~Timing_Net
  Timing_Analysis::Timing_Net, 119
~Timing_Point
  Timing_Analysis::Timing_Point, 122
~VerilogParser
  VerilogParser, 140
~WireDelayModel
  WireDelayModel, 144
_from
  Timing_Analysis::Edge, 32
_lumped_capacitance
  WireDelayModel, 146
_to
  Timing_Analysis::Edge, 32

abs
  Transitions, 137
add_fanout
  Timing_Analysis::Edge, 30
addCapacitor
  SpefNetISPD2013, 95
addCellInfo
  LibertyLibrary, 37
addCellInst
  Circuit_Netlist, 19
addResistor
  SpefNetISPD2013, 95
addSink
  Circuit_Netlist::Net, 62
aggregate
  Transitions, 135
arc
  Timing_Analysis::Timing_Point, 122
arc_number
  Timing_Analysis::Timing_Arc, 113
area
  LibertyCellInfo, 35
arrival_time
  Prime_Time_Output_Parser::Pin_Timing, 71
  Timing_Analysis::Timing_Point, 123

call_prime_time
  Timing_Analysis::Timing_Analysis, 107
capacitance
  LibertyPinInfo, 46
  SpefNetISPD2013::Node, 65
Capacitor
  SpefNetISPD2013::Capacitor, 15
ceff
  Timing_Analysis::Timing_Point, 124
cellType
  Circuit_Netlist::Logic_Gate, 52
Circuit_Netlist, 17
  ~Circuit_Netlist, 19
  addCellInst, 19
  Circuit_Netlist, 19
  Circuit_Netlist, 19
  get_net_topologic_index, 19
  getGate, 19
  getGateT, 20
  getGatesSize, 19
  getNet, 20
  getNetT, 20
  getNetsSize, 20
  getTopologicIndex, 20
  operator<<, 21, 22
  timing_arcs, 21
  timing_points, 21
  updateTopology, 21
  verilog, 21
Circuit_Netlist::Logic_Gate, 51
  cellType, 52
  fanoutNetIndex, 52
  inNets, 52
  inputDriver, 52
  Logic_Gate, 52
  name, 52
  primary_output, 52
  sequential, 52
Circuit_Netlist::Net, 60
  addSink, 62
  dummyNet, 62
  name, 62
  Net, 62
  sinks, 62
  sourceNode, 62

sourcePin, 62
Circuit_Netlist::Sink, 90
gate, 91
pinName, 91
Sink, 91
clear
LumpedCapacitanceWireDelayModel, 55
RCTreeWireDelayModel, 80
Timing_Analysis::Timing_Arc, 113
WireDelayModel, 144
clear_timing_info
Timing_Analysis::Timing_Point, 125
clock
Design_Constraints, 24
configuration.h
Parasitics, 150
SpefNet, 150
SpefParser, 150

delay
Timing_Analysis::Timing_Arc, 114
delay_at_fanout_node
LumpedCapacitanceWireDelayModel, 55
RCTreeWireDelayModel, 80
WireDelayModel, 144
Design_Constraints, 22
clock, 24
driving_cell, 24
input_delay, 24, 25
input_transition, 25
output_delay, 25
output_load, 27
output_loads_size, 27
dontTouch
LibertyCellInfo, 35
driving_cell
Design_Constraints, 24
dummyNet
Circuit_Netlist::Net, 62

Edge
Timing_Analysis::Edge, 30
EdgeType
transitions.h, 172
end
Timer, 102
exp
Transitions, 137

FALL
transitions.h, 172
fallDelay
LibertyTimingInfo, 48
fallTransition
LibertyTimingInfo, 48
fanoutNetIndex
Circuit_Netlist::Logic_Gate, 52
fanouts_size
Timing_Analysis::Edge, 30
first_PO_index
Timing_Analysis::Timing_Analysis, 107
footprint
LibertyCellInfo, 35
from
Timing_Analysis::Edge, 30
fromPin
LibertyTimingInfo, 48
full_timing_analysis
Timing_Analysis::Timing_Analysis, 107

gate
Circuit_Netlist::Sink, 91
gate_number
Timing_Analysis::Timing_Arc, 114
Timing_Analysis::Timing_Point, 125
get_net_topologic_index
Circuit_Netlist, 19
getCellIndex
LibertyLibrary, 37
getCellInfo
LibertyLibrary, 37, 38
getFall
Transitions, 135
getGate
Circuit_Netlist, 19
getGateT
Circuit_Netlist, 20
getGatesSize
Circuit_Netlist, 19
getMax
Transitions, 135
getMaxTransition
LibertyLibrary, 38
getMin
Transitions, 135
getNet
Circuit_Netlist, 20
getNetT
Circuit_Netlist, 20
getNetsSize
Circuit_Netlist, 20
getNode
SpefNetISPD2013, 95
getNodeIndex
SpefNetISPD2013, 95
getOtherNode
SpefNetISPD2013::Resistor, 83
getResistor
SpefNetISPD2013, 96
getReversed
Transitions, 136
getRise
Transitions, 136
getTimerStatus
TimerInterface, 104
getTopologicIndex
Circuit_Netlist, 20

INPUT
Timing_Analysis, 14
IF
RET, 33
IF< cond, ThenType, ElseType >, 32
IF< false, ThenType, ElseType >, 33
RET, 34
ISPD_2012
Traits, 130
inNets
Circuit_Netlist::Logic_Gate, 52
input_delay
Design_Constraints, 24, 25
input_transition
Design_Constraints, 25
inputDriver
Circuit_Netlist::Logic_Gate, 52
interpolate
LibertyLookupTableInterpolator, 41
LinearLibertyLookupTableInterpolator, 50
interpolator
WireDelayModel, 146
is
Parser, 69
is_PI
Timing_Analysis::Timing_Point, 125
is_PI_input
Timing_Analysis::Timing_Point, 125
is_PO
Timing_Analysis::Timing_Point, 126
is_input_pin
Timing_Analysis::Timing_Point, 125
is_output_pin
Timing_Analysis::Timing_Point, 125
is_reg_input
Timing_Analysis::Timing_Point, 126
isClock
LibertyPinInfo, 46
isInput
LibertyPinInfo, 46
isSequential
LibertyCellInfo, 35
isSpecialChar
Parser, 69
ispd_contest_benchmark
Traits, 130
ispd_contest_root
Traits, 130

leakagePower
LibertyCellInfo, 35
liberty_library.h
NEGATIVE_UNATE, 155
NON_UNATE, 155
POSITIVE_UNATE, 155
liberty_library.h
operator<<, 155
Unateness, 155
LibertyCellInfo, 34
area, 35
dontTouch, 35
footprint, 35
isSequential, 35
leakagePower, 35
LibertyCellInfo, 35
LibertyCellInfo, 35
name, 35
pins, 35
primaryOutput, 35
timingArcs, 35
LibertyLibrary, 35
~LibertyLibrary, 37
addCellInfo, 37
getCellIndex, 37
getCellInfo, 37, 38
getMaxTransition, 38
LibertyLibrary, 36
LibertyLibrary, 36
number_of_options, 38
LibertyLookupTable, 38
loadIndices, 39
tableVals, 39
transitionIndices, 39
LibertyLookupTableInterpolator, 40
interpolate, 41
round, 41
LibertyParser, 42
readFile, 43
LibertyPinInfo, 45
capacitance, 46
isClock, 46
isInput, 46
LibertyPinInfo, 46
LibertyPinInfo, 46
maxCapacitance, 46
name, 46
LibertyTimingInfo, 46
fallDelay, 48
fallTransition, 48
fromPin, 48
riseDelay, 48
riseTransition, 48
timingSense, 48
toPin, 48
LinearLibertyLookupTableInterpolator, 48
interpolate, 50
load
Timing_Analysis::Timing_Point, 126
loadIndices
LibertyLookupTable, 39
Logic_Gate
Circuit_Netlist::Logic_Gate, 52
lumped_capacitance
WireDelayModel, 145
LumpedCapacitanceWireDelayModel, 52
clear, 55
delay_at_fanout_node, 55

LumpedCapacitanceWireDelayModel, 55
LumpedCapacitanceWireDelayModel, 55
root_delay, 56
root_slew, 56
setFanoutPinCapacitance, 56
simulate, 56
slew_at_fanout_node, 57

MAKE_OPERATOR
transitions.h, 172
MAKE_SELF_OPERATOR
transitions.h, 172
MICRO
Timer, 103
MILI
Timer, 103
max
std::numeric_limits< Transitions< double > >, 66
Transitions, 138
maxCapacitance
LibertyPinInfo, 46
min
std::numeric_limits< Transitions< double > >, 66
Transitions, 138
Multi_Fanout_Edge
Timing_Analysis::Multi_Fanout_Edge, 60

NEGATIVE_UNATE
liberty_library.h, 155
NON_UNATE
liberty_library.h, 155
name
Circuit_Netlist::Logic_Gate, 52
Circuit_Netlist::Net, 62
LibertyCellInfo, 35
LibertyPinInfo, 46
SpefNetISPD2013::Node, 65
Timing_Analysis::Timing_Net, 119
Timing_Analysis::Timing_Point, 126
Net
Circuit_Netlist::Net, 62
net
Timing_Analysis::Timing_Point, 126, 127
netLumpedCap
SpefNetISPD2012, 93
SpefNetISPD2013, 97
netName
SpefNetISPD2012, 93
SpefNetISPD2013, 97
Node
SpefNetISPD2013::Node, 63
node
SpefNetISPD2013::Capacitor, 17
node1
SpefNetISPD2013::Resistor, 83
node2
SpefNetISPD2013::Resistor, 83
nodeIndex
SpefNetISPD2013::Node, 65
nodesSize
SpefNetISPD2013, 96
number_of_options
LibertyLibrary, 38
Timing_Analysis::Timing_Analysis, 107

OUTPUT
Timing_Analysis, 14
operator<<
Circuit_Netlist, 21, 22
liberty_library.h, 155
Prime_Time_Output_Parser::Pin_Timing, 71
Prime_Time_Output_Parser::Port_Timing, 73
SpefNetISPD2013, 96
Timer::Result, 84
Timing_Analysis::Timing_Arc, 116
Timing_Analysis::Timing_Net, 120
Timing_Analysis::Timing_Point, 129
Transitions, 138
operator=
Transitions, 136
Option
Timing_Analysis::Option, 67
option
Timing_Analysis::Timing_Analysis, 107
output_delay
Design_Constraints, 25
output_load
Design_Constraints, 27
output_loads_size
Design_Constraints, 27

PI
Timing_Analysis, 14
PI_INPUT
Timing_Analysis, 14
PO
Timing_Analysis, 14
POSITIVE_UNATE
liberty_library.h, 155
Parasitics
configuration.h, 150
Parasitics2012
spef_net.h, 163
Parasitics2013
spef_net.h, 163
parse_prime_time_output_file
Prime_Time_Output_Parser, 77
Parser, 68
~Parser, 69
is, 69
isSpecialChar, 69
Parser, 69
readLineAsTokens, 69
pin
Prime_Time_Output_Parser::Prime_Time_Output, 74
pin_capacitance
Timing_Analysis::Timing_Analysis, 107

pin_load
Timing_Analysis::Timing_Analysis, 108
pin_name
Prime_Time_Output_Parser::Pin_Timing, 71
pinName
Circuit_Netlist::Sink, 91
pins
LibertyCellInfo, 35
pins_size
Prime_Time_Output_Parser::Prime_Time_Output, 74
port
Prime_Time_Output_Parser::Prime_Time_Output, 74
port_name
Prime_Time_Output_Parser::Port_Timing, 73
ports_size
Prime_Time_Output_Parser::Prime_Time_Output, 74
pow
Transitions, 138
primary_output
Circuit_Netlist::Logic_Gate, 52
primaryOutput
LibertyCellInfo, 35
Prime_Time_Output_Parser, 75
parse_prime_time_output_file, 77
Prime_Time_Output_Parser, 77
Prime_Time_Output_Parser, 77
Prime_Time_Output_Parser::Prime_Time_Output, 75
Prime_Time_Output_Parser::Pin_Timing, 69
arrival_time, 71
operator<<, 71
pin_name, 71
slack, 71
slew, 71
Prime_Time_Output_Parser::Port_Timing, 71
operator<<, 73
port_name, 73
slack, 73
slew, 73
Prime_Time_Output_Parser::Prime_Time_Output, 73
pin, 74
pins_size, 74
port, 74
ports_size, 74
print_circuit_info
Timing_Analysis::Timing_Analysis, 108
print_info
Timing_Analysis::Timing_Analysis, 108

REGISTER_INPUT
Timing_Analysis, 14
RISE
transitions.h, 172
RCTreeWireDelayModel, 77
clear, 80
delay_at_fanout_node, 80
RCTreeWireDelayModel, 80
RCTreeWireDelayModel, 80
root_delay, 81
root_slew, 81
setFanoutPinCapacitance, 81
simulate, 81
slew_at_fanout_node, 82
RET
IF, 33
IF< false, ThenType, ElseType >, 34
readFile
LibertyParser, 43
SDCParser, 86
SpefParserISPD2012, 98
SpefParserISPD2013, 100
VerilogParser, 141
readLineAsTokens
Parser, 69
report_timing
Timing_Analysis::Timing_Analysis, 108
required_time
Timing_Analysis::Timing_Point, 127
Resistor
SpefNetISPD2013::Resistor, 83
resistors
SpefNetISPD2013::Node, 65
resistorsSize
SpefNetISPD2013, 96
reverse
Transitions, 137
riseDelay
LibertyTimingInfo, 48
riseTransition
LibertyTimingInfo, 48
root_delay
LumpedCapacitanceWireDelayModel, 56
RCTreeWireDelayModel, 81
WireDelayModel, 145
root_slew
LumpedCapacitanceWireDelayModel, 56
RCTreeWireDelayModel, 81
WireDelayModel, 145
round
LibertyLookupTableInterpolator, 41
runTimingAnalysisBlocking
TimerInterface, 104

SDCParser, 85
readFile, 86
STD_THRESHOLD
Traits, 130
sequential
Circuit_Netlist::Logic_Gate, 52
set
SpefNetISPD2013, 96
Timer::Result, 84
Transitions, 137
set_fanout
Timing_Analysis::Edge, 30

setFanoutPinCapacitance
LumpedCapacitanceWireDelayModel, 56
RCTreeWireDelayModel, 81
WireDelayModel, 145
simulate
LumpedCapacitanceWireDelayModel, 56
RCTreeWireDelayModel, 81
WireDelayModel, 146
Single_Fanout_Edge
Timing_Analysis::Single_Fanout_Edge, 90
Sink
Circuit_Netlist::Sink, 91
sinks
Circuit_Netlist::Net, 62
slack
Prime_Time_Output_Parser::Pin_Timing, 71
Prime_Time_Output_Parser::Port_Timing, 73
Timing_Analysis::Timing_Point, 127, 128
slew
Prime_Time_Output_Parser::Pin_Timing, 71
Prime_Time_Output_Parser::Port_Timing, 73
Timing_Analysis::Timing_Arc, 115
Timing_Analysis::Timing_Point, 128
slew_at_fanout_node
LumpedCapacitanceWireDelayModel, 57
RCTreeWireDelayModel, 82
WireDelayModel, 146
sourceNode
Circuit_Netlist::Net, 62
sourcePin
Circuit_Netlist::Net, 62
spef_net.h
Parasitics2012, 163
Parasitics2013, 163
SpefNet
configuration.h, 150
SpefNetISPD2012, 92
netLumpedCap, 93
netName, 93
SpefNetISPD2012, 93
SpefNetISPD2012, 93
SpefNetISPD2013, 93
~SpefNetISPD2013, 95
addCapacitor, 95
addResistor, 95
getNode, 95
getNodeIndex, 95
getResistor, 96
netLumpedCap, 97
netName, 97
nodesSize, 96
operator<<, 96
resistorsSize, 96
set, 96
SpefNetISPD2013, 95
SpefNetISPD2013, 95
SpefNetISPD2013::Capacitor, 15
Capacitor, 15
node, 17
value, 17
SpefNetISPD2013::Node, 62
capacitance, 65
name, 65
Node, 63
nodeIndex, 65
resistors, 65
SpefNetISPD2013::Resistor, 82
getOtherNode, 83
node1, 83
node2, 83
Resistor, 83
value, 83
SpefParser
configuration.h, 150
SpefParserISPD2012, 97
readFile, 98
SpefParserISPD2013, 99
readFile, 100
sqrt
Transitions, 138
start
Timer, 102
startTimingAnalysisNonBlocking
TimerInterface, 104
Status
TimerInterface, 104
std, 13
std::numeric_limits< Transitions< double > >, 65
max, 66
min, 66
zero, 66

TIMER_BUSY
TimerInterface, 104
TIMER_FINISHED_ERROR
TimerInterface, 104
TIMER_FINISHED_SUCCESS
TimerInterface, 104
TIMER_INTERFACEERROR
TimerInterface, 104
TIMER_NOT_STARTED
TimerInterface, 104
tableVals
LibertyLookupTable, 39
Timer, 101
~Timer, 102
end, 102
MICRO, 103
MILI, 103
start, 102
Timer, 102
Timer::Result, 85
value, 102
TimerInterface
TIMER_BUSY, 104
TIMER_FINISHED_ERROR, 104
TIMER_FINISHED_SUCCESS, 104

TIMER_INTERFACEERROR, 104
TIMER_NOT_STARTED, 104
Timer::Result, 84
operator<<, 84
set, 84
Timer, 85
TimerInterface, 103
getTimerStatus, 104
runTimingAnalysisBlocking, 104
startTimingAnalysisNonBlocking, 104
Status, 104
wait, 104
Timing_Analysis
INPUT, 14
OUTPUT, 14
PI, 14
PI_INPUT, 14
PO, 14
REGISTER_INPUT, 14
Timing_Analysis, 13
Timing_Point_Type, 14
Timing_Analysis::Option, 67
Timing_Analysis::Timing_Analysis, 106
Timing_Analysis::Timing_Net, 120
Timing_Analysis::Edge
_from, 32
_to, 32
add_fanout, 30
Edge, 30
fanouts_size, 30
from, 30
set_fanout, 30
Timing_Analysis::Edge< T >, 27
Timing_Analysis::Multi_Fanout_Edge
Multi_Fanout_Edge, 60
to, 60
Timing_Analysis::Multi_Fanout_Edge< T >, 57
Timing_Analysis::Option, 66
Option, 67
Timing_Analysis, 67
Timing_Analysis::Single_Fanout_Edge
Single_Fanout_Edge, 90
to, 90
Timing_Analysis::Single_Fanout_Edge< T >, 87
Timing_Analysis::Timing_Analysis, 105
~Timing_Analysis, 106
call_prime_time, 107
first_PO_index, 107
full_timing_analysis, 107
number_of_options, 107
option, 107
pin_capacitance, 107
pin_load, 108
print_circuit_info, 108
print_info, 108
report_timing, 108
Timing_Analysis, 106
timing_arc, 108
timing_arcs_size, 109
timing_net, 109
timing_nets_size, 109
timing_point, 109
timing_points_size, 109
validate_with_prime_time, 109
Timing_Analysis::Timing_Arc, 110
~Timing_Arc, 113
arc_number, 113
clear, 113
delay, 114
gate_number, 114
operator<<, 116
slew, 115
Timing_Arc, 113
Timing_Analysis::Timing_Net, 116
~Timing_Net, 119
name, 119
operator<<, 120
Timing_Analysis, 120
Timing_Net, 119
Timing_Point, 120
wire_delay_model, 119
Timing_Analysis::Timing_Point, 120
~Timing_Point, 122
arc, 122
arrival_time, 123
ceff, 124
clear_timing_info, 125
gate_number, 125
is_PI, 125
is_PI_input, 125
is_PO, 126
is_input_pin, 125
is_output_pin, 125
is_reg_input, 126
load, 126
name, 126
net, 126, 127
operator<<, 129
required_time, 127
slack, 127, 128
slew, 128
Timing_Net, 129
Timing_Point, 122
update_slack, 129
Timing_Arc
Timing_Analysis::Timing_Arc, 113
Timing_Net
Timing_Analysis::Timing_Net, 119
Timing_Analysis::Timing_Point, 129
Timing_Point
Timing_Analysis::Timing_Net, 120
Timing_Analysis::Timing_Point, 122
Timing_Point_Type
Timing_Analysis, 14
timing_arc
Timing_Analysis::Timing_Analysis, 108

timing_arcs
Circuit_Netlist, 21
timing_arcs_size
Timing_Analysis::Timing_Analysis, 109
timing_net
Timing_Analysis::Timing_Analysis, 109
timing_nets_size
Timing_Analysis::Timing_Analysis, 109
timing_point
Timing_Analysis::Timing_Analysis, 109
timing_points
Circuit_Netlist, 21
timing_points_size
Timing_Analysis::Timing_Analysis, 109
TimingAnalysis/doc/Index.dox, 147
TimingAnalysis/src/include/circuit_netlist.h, 147
TimingAnalysis/src/include/configuration.h, 148
TimingAnalysis/src/include/design_constraints.h, 150
TimingAnalysis/src/include/edge.h, 151
TimingAnalysis/src/include/liberty_library.h, 153
TimingAnalysis/src/include/multi_fanout_edge.h, 156
TimingAnalysis/src/include/parser.h, 157
TimingAnalysis/src/include/single_fanout_edge.h, 159
TimingAnalysis/src/include/spef_net.h, 161
TimingAnalysis/src/include/timer.h, 163
TimingAnalysis/src/include/timer_interface.h, 164
TimingAnalysis/src/include/timing_analysis.h, 165
TimingAnalysis/src/include/timing_arc.h, 166
TimingAnalysis/src/include/timing_net.h, 168
TimingAnalysis/src/include/timing_point.h, 169
TimingAnalysis/src/include/transitions.h, 170
TimingAnalysis/src/include/wire_delay_model.h, 172
timingArcs
LibertyCellInfo, 35
timingSense
LibertyTimingInfo, 48
to
Timing_Analysis::Multi_Fanout_Edge, 60
Timing_Analysis::Single_Fanout_Edge, 90
toPin
LibertyTimingInfo, 48
Traits, 130
ISPD_2012, 130
ispd_contest_benchmark, 130
ispd_contest_root, 130
STD_THRESHOLD, 130
transitionIndices
LibertyLookupTable, 39
Transitions
abs, 137
aggregate, 135
exp, 137
getFall, 135
getMax, 135
getMin, 135
getReversed, 136
getRise, 136
max, 138
min, 138
operator<<, 138
operator=, 136
pow, 138
reverse, 137
set, 137
sqrt, 138
Transitions, 134
Transitions< T >, 131
transitions.h
FALL, 172
RISE, 172
transitions.h
EdgeType, 172
MAKE_OPERATOR, 172
MAKE_SELF_OPERATOR, 172

Unateness
liberty_library.h, 155
update_slack
Timing_Analysis::Timing_Point, 129
updateTopology
Circuit_Netlist, 21

validate_with_prime_time
Timing_Analysis::Timing_Analysis, 109
value
SpefNetISPD2013::Capacitor, 17
SpefNetISPD2013::Resistor, 83
Timer, 102
verilog
Circuit_Netlist, 21
VerilogParser, 139
~VerilogParser, 140
readFile, 141

wait
TimerInterface, 104
wire_delay_model
Timing_Analysis::Timing_Net, 119
WireDelayModel, 141
~WireDelayModel, 144
_lumped_capacitance, 146
clear, 144
delay_at_fanout_node, 144
interpolator, 146
lumped_capacitance, 145
root_delay, 145
root_slew, 145
setFanoutPinCapacitance, 145
simulate, 146
slew_at_fanout_node, 146
WireDelayModel, 144
WireDelayModel, 144

zero
std::numeric_limits< Transitions< double > >, 66