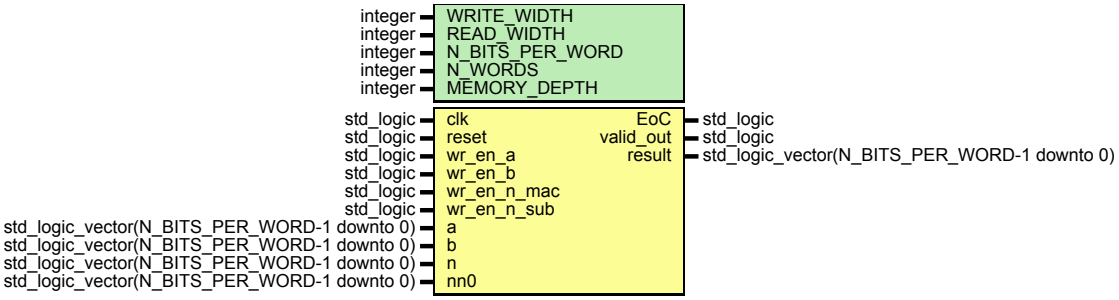
Entity: monmult_module

- **File**: monmult_module.vhd

Diagram



Description

this is the module that contains all the others, and wires them together

Generics

Generic name	Type	Value	Description
WRITE_WIDTH	integer	16	MUST BE SET THE SAME AS NUMBER OF BITS PER WORD
READ_WIDTH	integer	16	MUST BE SET THE SAME AS NUMBER OF BITS PER WORD
N_BITS_PER_WORD	integer	16	
N_WORDS	integer	16	IS THE RESULT OF TOTAL_BITS/N_BITS_PER_WORD
MEMORY_DEPTH	integer	16	IT IS THE SAME OF N_WORDS

Ports

Port name	Direction	Type	Description
clk	in	std_logic	
reset	in	std_logic	
wr_en_a	in	std_logic	will be rised by the testbench when the data are ready to be loaded in the input memories
wr_en_b	in	std_logic	will be rised by the testbench when the data

			are ready to be loaded in the input memories
wr_en_n_mac	in	std_logic	will be rised by the testbench when the data are ready to be loaded in the input memories
wr_en_n_sub	in	std_logic	will be rised by the testbench when the data are ready to be loaded in the input memories
a	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	data inputs to be loaded to input memories will be fed one word at a time
b	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	data inputs to be loaded to input memories will be fed one word at a time
n	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	data inputs to be loaded to input memories will be fed one word at a time
nn0	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	data inputs to be loaded to input memories will be fed one word at a time
EoC	out	std_logic	End Of Conversion, is high on the last word of the valid result
valid_out	out	std_logic	Is high only when the subtractor is giving out the correct result
result	out	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	

Signals

Name	Type	Description
a_mem	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
b_mem	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
n_mac_mem	std_logic_vector(N_BITS_PER_WORD-	

	1 downto 0)	
n_sub_mem	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
start_a	std_logic	
start_b	std_logic	
start_n_mac	std_logic	
start_n_sub	std_logic	
EoC_sig	std_logic	
EoC_reg	std_logic	
memory_full	std_logic	
start_mem	std_logic	signals that memories can begin exposing data at their outputs
start_modules	std_logic	signal that modules can start to wait for their respective latencies

Constants

Name	Type	Value	Description
LATENCY_AB	integer	1	this constant controls after how many cycles startig from the start signal is mac_ab going to start reading
LATENCY_N_SUB	integer	LATENCY_AB +N_WORDS* (N_WORDS-1)+4	this constant controls after how many cycles startig from the start signal is sub going to start reading
LATENCY_N_MAC	integer	LATENCY_AB+2	this constant controls after how many cycles startig from the start signal is mac_mn going to start reading

Processes

- unnamed: (clk)

Instantiations

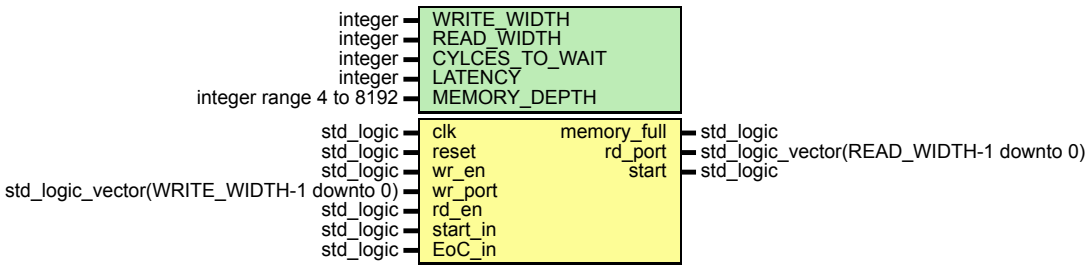
- inst_cios_1w: cios_top_1w
- mem_a_inst: input_mem_abn
- mem_b_inst: input_mem_abn
- mem_n_mac_inst: input_mem_abn

- mem_n_sub_inst: input_mem_abn
- regulator_inst: start_regulator

Entity: input_mem_abn

- **File**: input_mem_abn.vhd

Diagram



Description

this module takes care of storing the entire input vector, and exposes the rigth word at the output at each cycle this module will be istantiated three times:

- for the input a, which exposes one word per cycle
- for the input b, which expses one word every N_WORDS cycles
- for the input n, which exposes one word per cycle but with an initial delay

Generics

Generic name	Type	Value	Description
WRITE_WIDTH	integer	8	
READ_WIDTH	integer	8	In this implementation, READ_WIDTH=WRITE_WIDTH
CYLCES_TO_WAIT	integer	4	after the first word has been exposed, after how many cycles the next word will be exposed
LATENCY	integer	4	Initial time to wait after receving start_in before exposing the first word
MEMORY_DEPTH	integer range 4 to 8192	16	in this implementation, is equal to TOT_BITS/WRITE_WIDTH

Ports

Port name	Direction	Type	Description
clk	in	std_logic	
reset	in	std_logic	
memory_full	out	std_logic	unused in this implementation

wr_en	in	std_logic	has to be kept high while writing
wr_port	in	std_logic_vector(WRITE_WIDTH-1 downto 0)	accepts one word at a time, loaded by the testbench
rd_en	in	std_logic	has to be kept high while reading
rd_port	out	std_logic_vector(READ_WIDTH-1 downto 0)	exposes one word at a time, at the right cycle
start	out	std_logic	used to notify start_regulator that reading phase is over
start_in	in	std_logic	this notifies the memory that all memories are full and ready to start providing data
EoC_in	in	std_logic	End of Conversion input to notify the memory that the content of the memory can be reset on order to be ready for another computation

Signals

Name	Type	Description
memory	memory_type	
write_counter	integer range 0 to MEMORY_DEPTH	counts the times a writing in the memory is done, does only one full cycle
read_counter	integer range 0 to MEMORY_DEPTH	counts the times a reading from the memory is done, does as many full cycles as necessary
memory_full_int	std_logic	used to tell if reading phase can start
cycle_counter	integer	counts the cycle between one reading and the next(i.e 1 for a, N_WORDS for b)
initial_counter	integer	counts the cycle to wait before the beginning of the reading phase
begin_reading	std_logic	flag, to 1 when writing phase is finished
start_flag	std_logic	used to manage the start signal, which may be high for one or more cycles
start_int	std_logic	

Types

Name	Type	Description

memory_type		

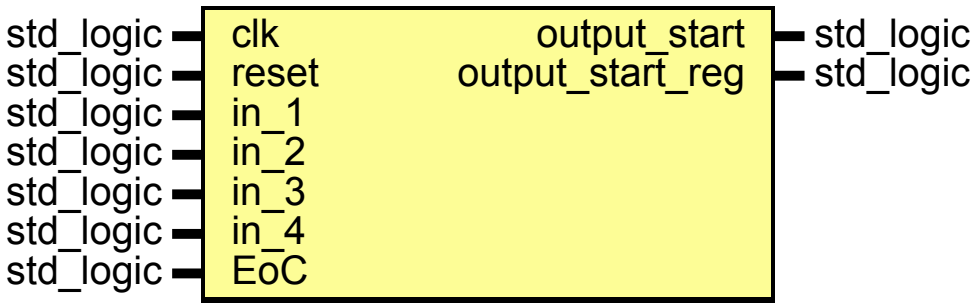
Processes

- unnamed: (clk,reset, EoC_in)

Entity: start_regulator

- **File**: start_regulator.vhd

Diagram



Description

takes as input the start output of the memories, and feeds to the modules and the memories a start signal, and it is useful if the memories are not filled at the same time

Ports

Port name	Direction	Type	Description
clk	in	std_logic	
reset	in	std_logic	
in_1	in	std_logic	outputs of the memories, goes high when memory full, can be one cycle or more cycle long
in_2	in	std_logic	outputs of the memories, goes high when memory full, can be one cycle or more cycle long
in_3	in	std_logic	outputs of the memories, goes high when memory full, can be one cycle or more cycle long
in_4	in	std_logic	outputs of the memories, goes high when memory full, can be one cycle or more cycle long
EoC	in	std_logic	End of Conversion, resets the module when computation is over
output_start	out	std_logic	notifies the memories that the can start exposing on the rd_port
output_start_reg	out	std_logic	notifies the computation modules that they can start reading

Signals

Name	Type	Description
flag_1	std_logic	flags to hold the value '1' of the input signal
flag_2	std_logic	flags to hold the value '1' of the input signal
flag_3	std_logic	flags to hold the value '1' of the input signal
flag_4	std_logic	flags to hold the value '1' of the input signal
accept_1	std_logic	is '1' when the module has not yes stored a previous value '1'
accept_2	std_logic	is '1' when the module has not yes stored a previous value '1'
accept_3	std_logic	is '1' when the module has not yes stored a previous value '1'
accept_4	std_logic	is '1' when the module has not yes stored a previous value '1'
output_start_int	std_logic	replica of output_start, needed in order for the output port to be readable
output_start_reg_int	std_logic	replica of output_start delayed by one cylce
output_start_reg_reg_int	std_logic	

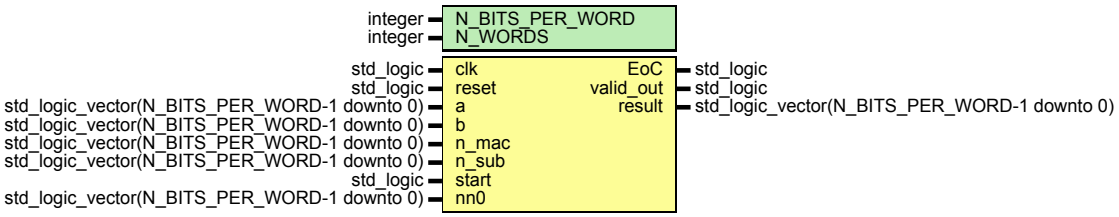
Processes

- unnamed: (clk, reset)

Entity: cios_top_1w

- **File**: cios_top_1w.vhd

Diagram



Description

This module includes all the computational blocks. It's embedded inside monmult_module, together with the memories.

Generics

Generic name	Type	Value	Description
N_BITS_PER_WORD	integer	8	
N_WORDS	integer	4	

Ports

Port name	Direction	Type	Description
clk	in	std_logic	
reset	in	std_logic	
a	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
b	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
n_mac	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
n_sub	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
start	in	std_logic	indicates that memories are full and computation is going to start, can be one or more cycles long

nn0	in	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	this is the least significant word of the inverse of N modulo R, which is computed by the PC and fed to the module
EoC	out	std_logic	End Of Conversion, is high on the last word of the valid result
valid_out	out	std_logic	Is high only when the subtractor is giving out the correct result
result	out	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	

Signals

Name	Type	Description
t_out_ab	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
c_out_ab	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	
m	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_mac_out_mn	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
c_out_mn	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_out_mn	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
c_in_ab	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
c_in_mn	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
c_out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
n_in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_adder	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_out_mult	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	

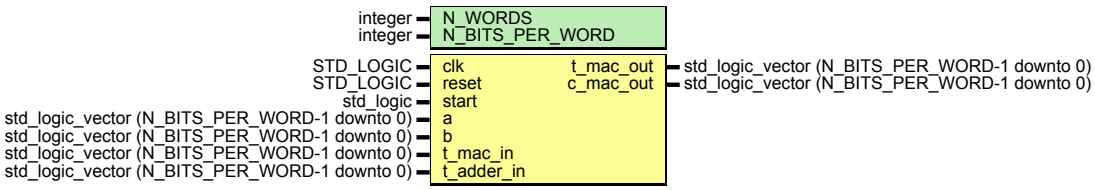
Instantiations

- mac_ab_inst: FSM_mac_ab
- mac_mn_inst: FSM_mac_mn
- mult_inst: FSM_mult
- add_inst: FSM_add
- sub_inst: FSM_sub_v2

Entity: FSM_mac_ab

- **File**: FSM_mac_ab.vhd

Diagram



Description

this FSM uses two counters i and j both going from 0 to N_WORDS-1 and are used to control what to expose at the output at every cycle

This FSM:

starts at cycle 0,

- reads a every cycle,
- reads b every N_WORDS cycles,
- reads t = 0 for i=0,
- reads t = t_mac_in for i>=1, j<N_WORDS,
- reads t = t_adder_in for i>=1, j=N_WORDS,
- if N_WORDS>4, a shift register is added in order to take into account the delay between the clock mac_mn exposes its t output and mac_ab reads it
- Cout is brought to the output everytime, adder has to sample the correct one

Generics

Generic name	Type	Value	Description
N_WORDS	integer	4	
N_BITS_PER_WORD	integer	8	

Ports

Port name	Direction	Type	Description
clk	in	STD_LOGIC	
reset	in	STD_LOGIC	
start	in	std_logic	
a	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
b	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	

t_mac_in	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_adder_in	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_mac_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
c_mac_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	

Signals

Name	Type	Description
i	integer	coarse counter
j	integer	fine counter
sr_in	std_logic_vector(t_adder_in'range)	
a_dut	std_logic_vector(a'range)	wrapper signal for the combinatorial mac module
b_dut	std_logic_vector(a'range)	wrapper signal for the combinatorial mac module
t_in_dut	std_logic_vector(a'range)	wrapper signal for the combinatorial mac module
c_in_dut	std_logic_vector(a'range)	wrapper signal for the combinatorial mac module
s_out_dut	std_logic_vector(a'range)	wrapper signal for the combinatorial mac module
c_out_dut	std_logic_vector(a'range)	wrapper signal for the combinatorial mac module
din_dut	std_logic_vector(t_mac_in'range)	wrapper signal for the sr
dout_dut	std_logic_vector(t_mac_in'range)	wrapper signal for the sr
counter	integer	
start_reg	std_logic	
send_t_mac_in	std_logic	controls when the sr needs to store the mac_mn output
send_t_adder	std_logic	controls when the sr needs to store the adder output
counter_mac	integer	

Processes

- FSM_process: (clk,reset)

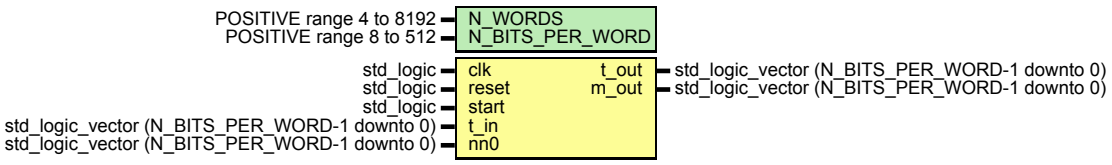
Instantiations

- mac_inst: simple_1w_mac

Entity: FSM_mult

- **File**: FSM_mult.vhd

Diagram



Description

This block implements an FSM controlling the inputs of a combinatorial 1w-multiplier. It also functions as a register for t_out_ab. It starts after 1 clock cycle w.r.t. the start signal, and performs a multiplication every s cycles. The t_out port copies and syncronizes the t_out data from mac_ab.

Generics

Generic name	Type	Value	Description
N_WORDS	POSITIVE range 4 to 8192	4	Number of words per each operand
N_BITS_PER_WORD	POSITIVE range 8 to 512	32	Number of bits per each word

Ports

Port name	Direction	Type	Description
clk	in	std_logic	clock signal
reset	in	std_logic	asyncronous reset signal
start	in	std_logic	start signal from outside
t_in	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	input word from mac_ab
nn0	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	input n'(0)
t_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	output t (input t delayed by 1 cycle)
m_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	output product

Signals

Name	Type	Description
i_counter	natural range 0 to N_WORDS	signal to count i cycle
j_counter	natural range 0 to N_WORDS	signal to count words of an i_cycle
a_sig	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	signals to sample input data
start_reg	std_logic	

Processes

- Mult_FSM: (clk,reset)
  - **Description** This FSM controls the inputs of the combinatorial multiplier

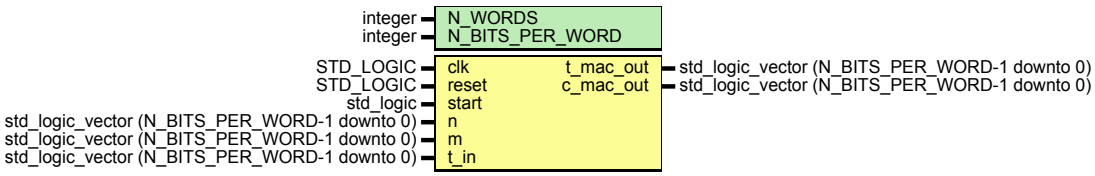
Instantiations

- mult_1w: simple_1w_mult

Entity: FSM_mac_mn

- **File**: FSM_mac_mn.vhd

Diagram



Description

This FSM uses two counters i and j both going from 0 to N_WORDS-1 and are used to control what to expose at the output at every cycle.N_BITS_PER_WORD

this FSM:

- starts at cycle 0
- reads n every cycle (n is equivalent to a in mac_ab)
- reads m every cycle with j=0 (m is equivalent to b in mac_ab)
- reads t from multiplier. the multiplier has registered the t value of mac_ab, to give a 1 cycle delay
- Cout is brought to the output everytime, adder has to sample it at the correct clock cycle

Generics

Generic name	Type	Value	Description
N_WORDS	integer	4	
N_BITS_PER_WORD	integer	8	

Ports

Port name	Direction	Type	Description
clk	in	STD_LOGIC	
reset	in	STD_LOGIC	
start	in	std_logic	indicates that memories are full and computation is going to start, can be one or more cycles long
n	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
m	in	std_logic_vector	

		(N_BITS_PER_WORD-1 downto 0)	
t_in	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
t_mac_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	
c_mac_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	

Signals

Name	Type	Description
i	integer	counter, cfr mac_ab
j	integer	counter, cfr mac_ab
n_dut	std_logic_vector(n'range)	wrapper signal for the combinatorial mac module
m_dut	std_logic_vector(n'range)	wrapper signal for the combinatorial mac module
t_in_dut	std_logic_vector(n'range)	wrapper signal for the combinatorial mac module
c_in_dut	std_logic_vector(n'range)	wrapper signal for the combinatorial mac module
s_out_dut	std_logic_vector(n'range)	wrapper signal for the combinatorial mac module
c_out_dut	std_logic_vector(n'range)	wrapper signal for the combinatorial mac module
start_reg	std_logic	
finished	std_logic	unused in this implementation
start_counter	integer	counts up to LATENCY, and measures time before computation begins
start_comp	std_logic	

Constants

Name	Type	Value	Description
LATENCY	integer	1	clock cycles to wait after having received start

Processes

- FSM_process: (clk,reset)

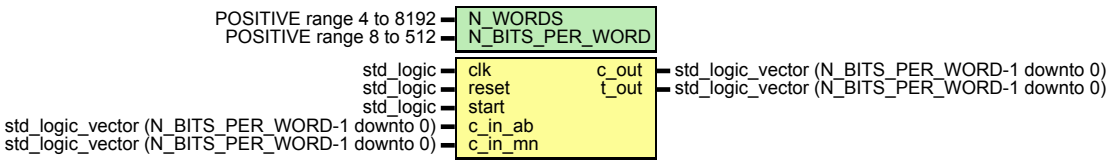
Instantiations

- mac_inst: simple_1w_mac

Entity: FSM_add

- **File**: FSM_add.vhd

Diagram



Description

This FSM module controls the inputs of the 1W_ADDER combinatorial block. The adder should start s clock cycles after the "start" signal arrives. After starting, it has to compute 3 sums:

- Sum1: 'Carry from fsm_mac_ab' + 'Previous computation of the adder'
- Sum2: 'Result of sum1 ' + 'Carry from fsm_mac_mn'
- Sum3: 'Carry of sum1' + 'Carry of sum2'

After finishing it has to wait for (s-4) clocks (SUM_3_1) before accepting a new value from mac_ab. When all the computations for the current multiplication are done, it returns in the IDLE state, waiting for a new 'start' signal.

Generics

Generic name	Type	Value	Description
N_WORDS	POSITIVE range 4 to 8192	4	Number of words per each operand
N_BITS_PER_WORD	POSITIVE range 8 to 512	32	Number of bits per each word

Ports

Port name	Direction	Type	Description
clk	in	std_logic	clock signal
reset	in	std_logic	asyncronous reset signal
start	in	std_logic	start signal, tells the fsm when a new mult is started
c_in_ab	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	carry in from mac_ab
c_in_mn	in	std_logic_vector (N_BITS_PER_WORD-1 downto	carry in from mac_mn

		0)	
c_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	carry out from combinatorial adder
t_out	out	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	result of combinatorial adder

Signals

Name	Type	Description
state	state_type	FSM state signal
a_sig	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	mainly used to load carry from outside
b_sig	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	mainly used to load internal values
start_reg	std_logic	start signal flipflop '1' -> a conversion is in progress.
c_out_reg	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	signal to store c_out for Sum3
t_out_sig	std_logic_vector(t_out'range)	signal linked to combinatorial adder t_out
c_out_sig	std_logic_vector(c_out'range)	signal linked to combinatorial adder c_out
delay_counter	unsigned (9 downto 0)	counter to track cycles to wait before starting a new sum
i_counter	unsigned (9 downto 0)	counter to keep track of current i_loop cycle (and return to idle when finished)

Constants

Name	Type	Value	Description
DELAY_SUM_3_1	UNSIGNED (9 downto 0)	to_unsigned((N_WORDS - 4), 10)	

Types

Name	Type	Description
state_type	(IDLE, SUM_1,	FSM state type

	SUM_1B, SUM_2, SUM_3)	

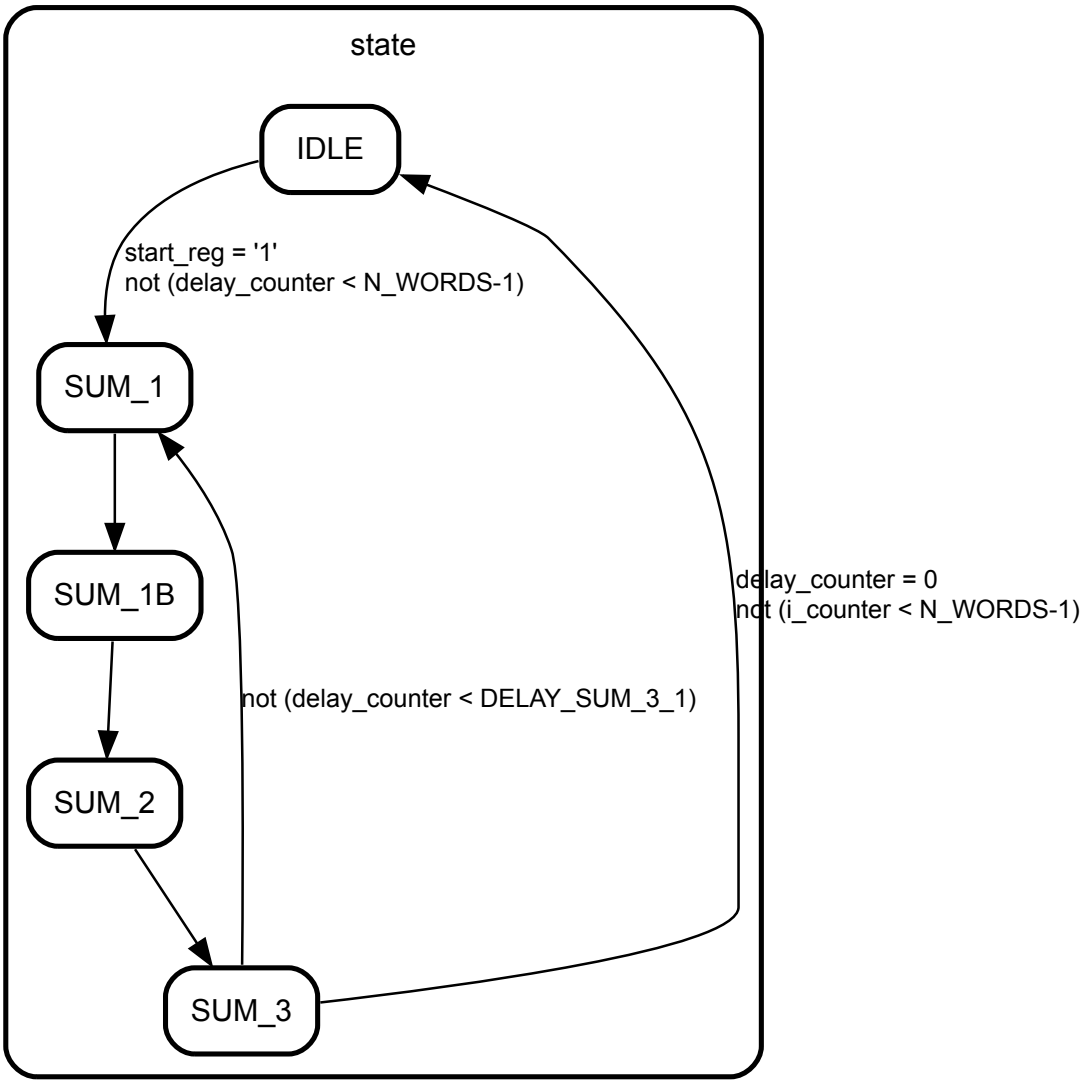
Processes

- FSM: (clk,reset)

Instantiations

- add_1w: simple_1w_adder

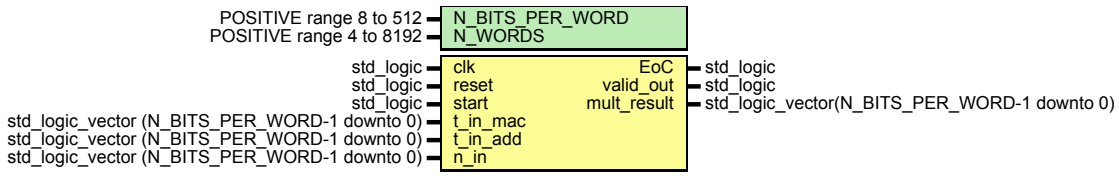
State machines



Entity: FSM_sub_v2

- **File**: FSM_sub_v2.vhd

Diagram



Description

This module describes an FSM controlling the inputs of a combinatorial 1word subtractor. It takes t and n respectively from the mac_mn/adder and from the n_memory, and it subtract the two word by word. It also stores both t and the result of the difference (t-n). At the end of the difference:

- If t > n (b_out = 0) => mult_result = t-n
- If t <= n (b_out = 1) => mult_result = t

In total, the block has to perform s+1 subtractions:

- First s subtractions: t[i] - n[i] -> out_borrows are brought to input_borrow port again
- Last subtraction: adder third sum result (t[s]) - prev_b_out At the end of the conversion an EoC signal is asserted for 1 clock cycle. When the output port is outputting valid data, the valid_out port is '1'

Generics

Generic name	Type	Value	Description
N_BITS_PER_WORD	POSITIVE range 8 to 512	32	Number of bits per word
N_WORDS	POSITIVE range 4 to 8192	4	Number of words per operand

Ports

Port name	Direction	Type	Description
clk	in	std_logic	clock signal
reset	in	std_logic	asyncronous reset signal
start	in	std_logic	start signal (when '1', a new mult has started)
EoC	out	std_logic	End of Conversion signal. High for 1 clock when mult finished
valid_out	out	std_logic	high when rersult is being

			written on output port
t_in_mac	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	t data coming from mac_mn
t_in_add	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	t data coming from adder
n_in	in	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	n coming from n memory
mult_result	out	std_logic_vector(N_BITS_PER_WORD-1 downto 0)	final result of monmult

Signals

Name	Type	Description
state	state_type	FSM state signal
diff_out_temp	out_temp_type	memory to store difference before comparison
t_out_temp	out_temp_type	memory to store t_in_mac before comparison
start_reg	std_logic	flag signal: '1' means a multiplication is in progress
read_counter	natural range 0 to N_WORDS	counter for reads from mac, adder and n memory
write_counter	natural range 0 to N_WORDS-1	counter for output writes
wait_counter	natural range 0 to CLK_TO_WAIT-1	counter for waiting before starting subtraction
t_in_sig	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	signal linked to first input of combinatorial subtractor
n_in_sig	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	signal linked to second input of combinatorial subtractor
b_in_sig	std_logic_vector (0 downto 0)	signal linked to input borrow of combinatorial subtractor
diff_out_sig	std_logic_vector (N_BITS_PER_WORD-1 downto 0)	signal linked to out result of combinatorial subtractor
b_out_sig	std_logic_vector (0 downto 0)	signal linked to out borrow of combinatorial subtractor
b_out_reg	std_logic_vector (0 downto 0)	

Constants

Name	Type	Value	Description
CLK_TO_WAIT	positive	N_WORDS * (N_WORDS-1) + 3	clock cycles to wait before starting comparison

Types

Name	Type	Description
state_type	(IDLE, SUB_STATE, COMPARE_STATE)	FSM state type
out_temp_type		memory type to store words before comparison

Processes

- FSM: (clk,reset)
  - **Description** FSM process

Instantiations

- sub_1w_inst: simple_1w_sub

State machines

- FSM process