FSMD Fibonacci

Last updated 1/28/21

FSMD - Fibonacci

- Fibonacci Sequence

$$fib(i) = \begin{cases} 0 & i = 0 \\ 1 & i = 1 \\ fib(i-1) + fib(i-2) & i > 1 \end{cases}$$

for i= 6

0-1-1-2-3-5-8

FSMD - Fibonacci

- FSMD

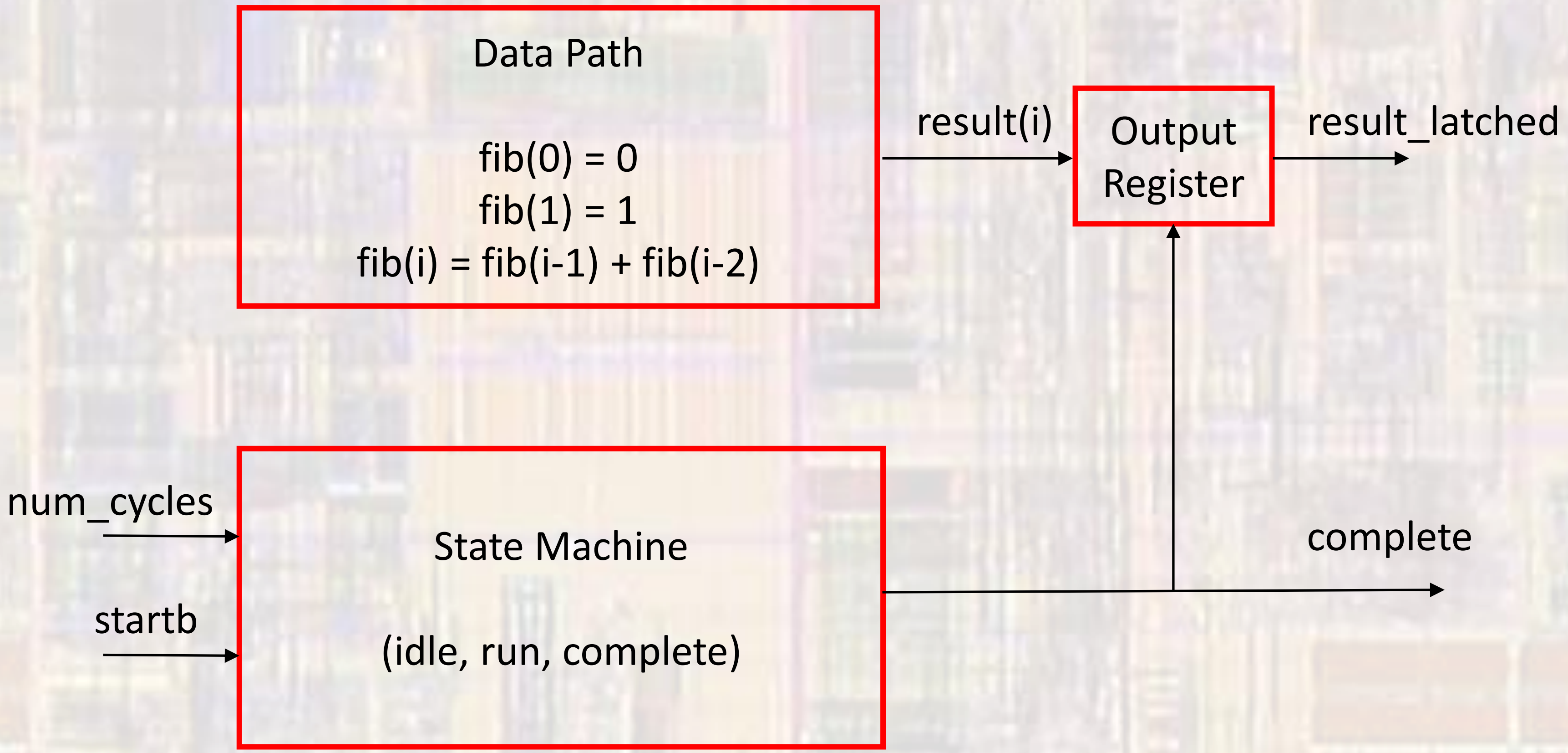
  - Input Signals
    - clk, rstb
    - num_cycles – how many numbers to generate - vector
    - startb – start generating (bar)

  - Output Signals
    - complete – generation complete
    - result_latched – Final Fibonacci value

  - States
    - Idle
    - Run
    - Complete

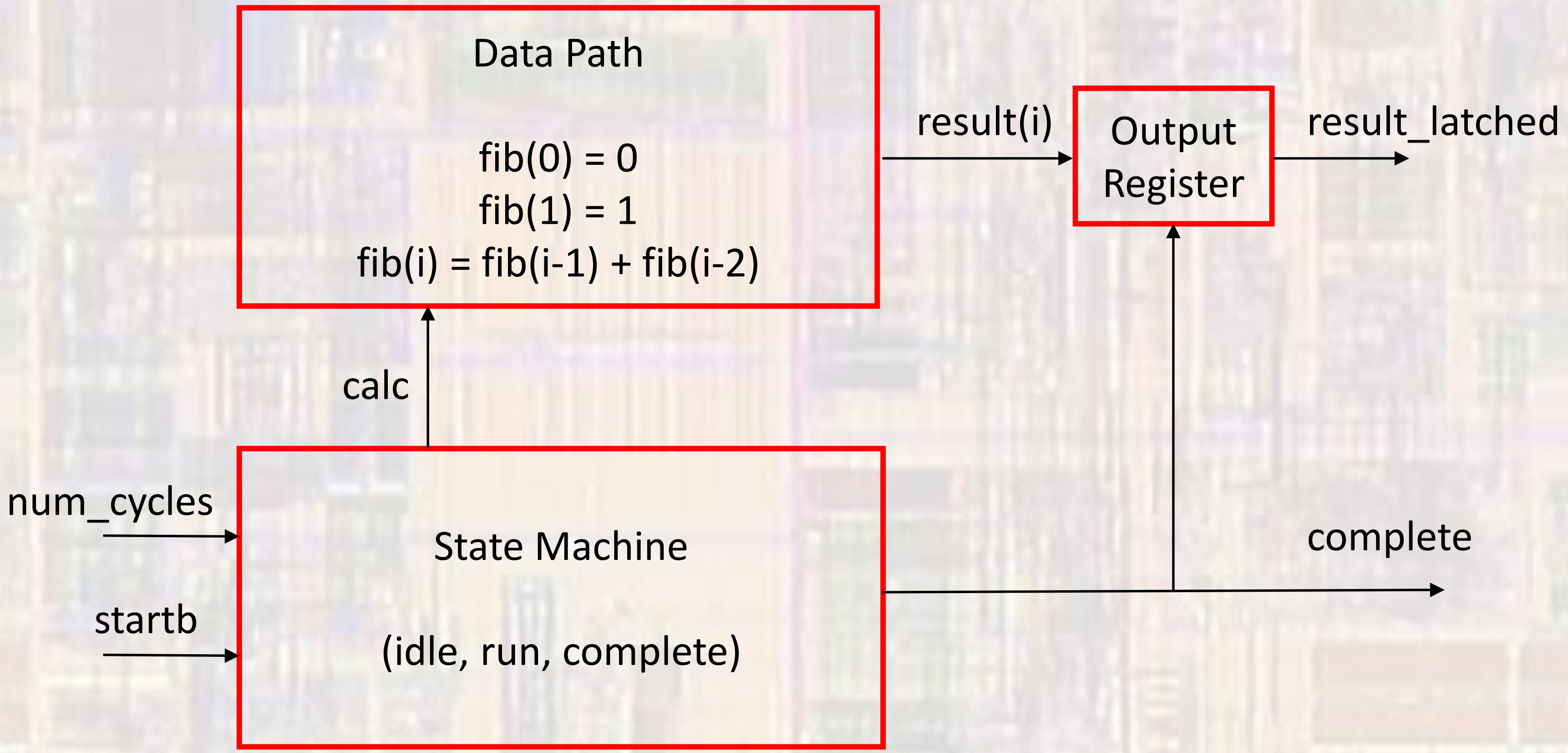
FSMD - Fibonacci

- FSMD



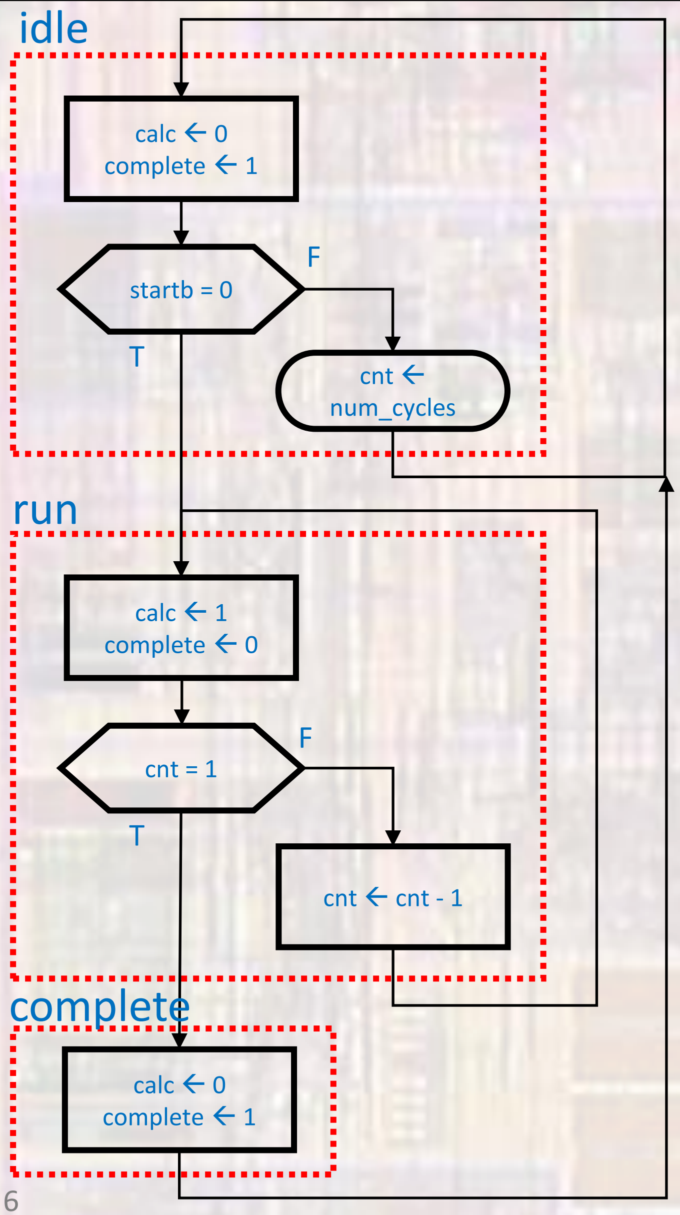
FSMD - Fibonacci

- FSMD



FSMD - Fibonacci

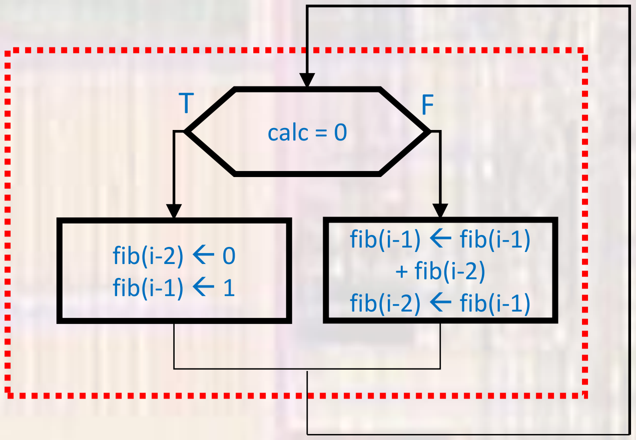
- ASM
  - Control Path



FSMD - Fibonacci

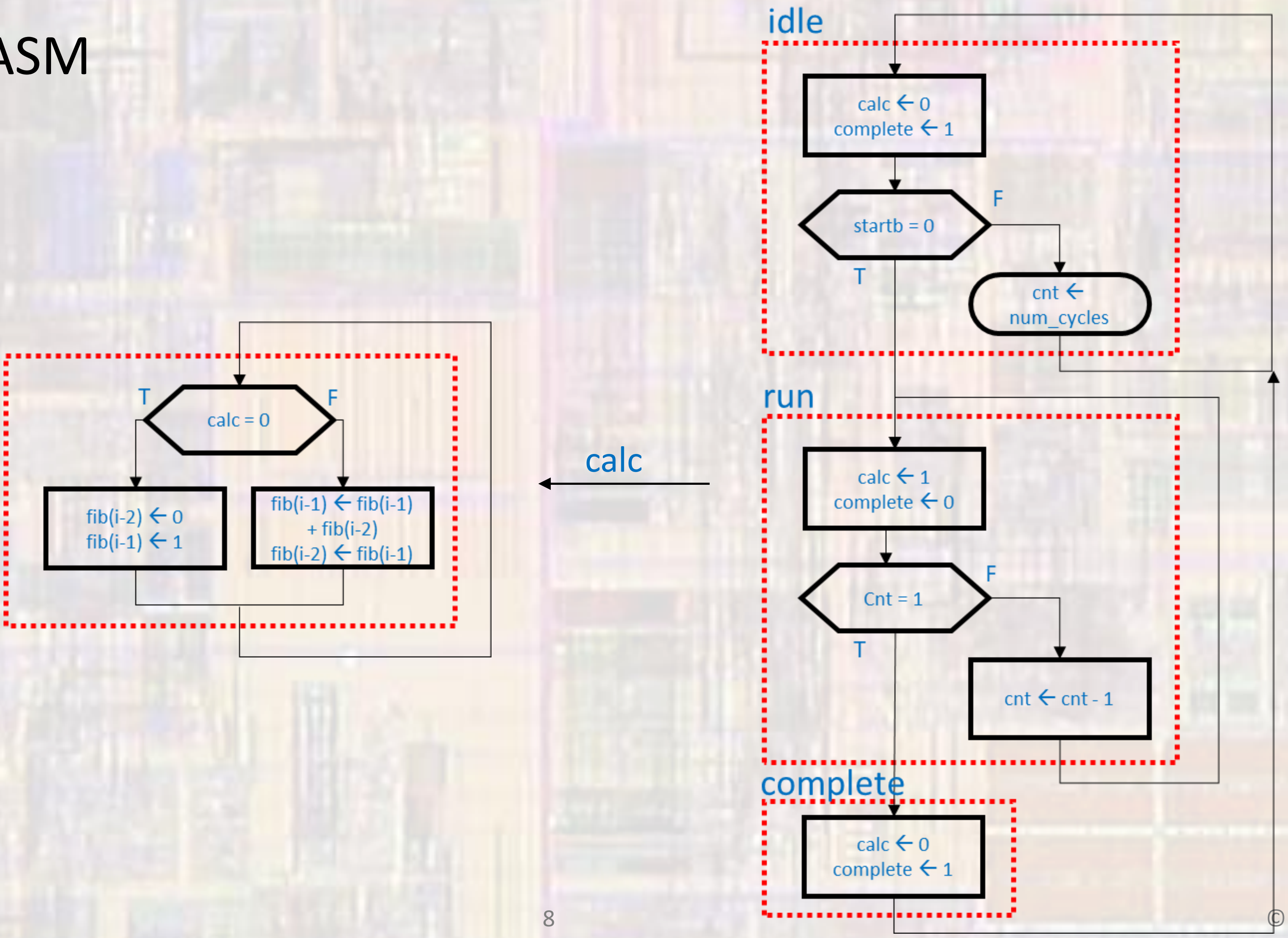
- ASM

  - Data Path



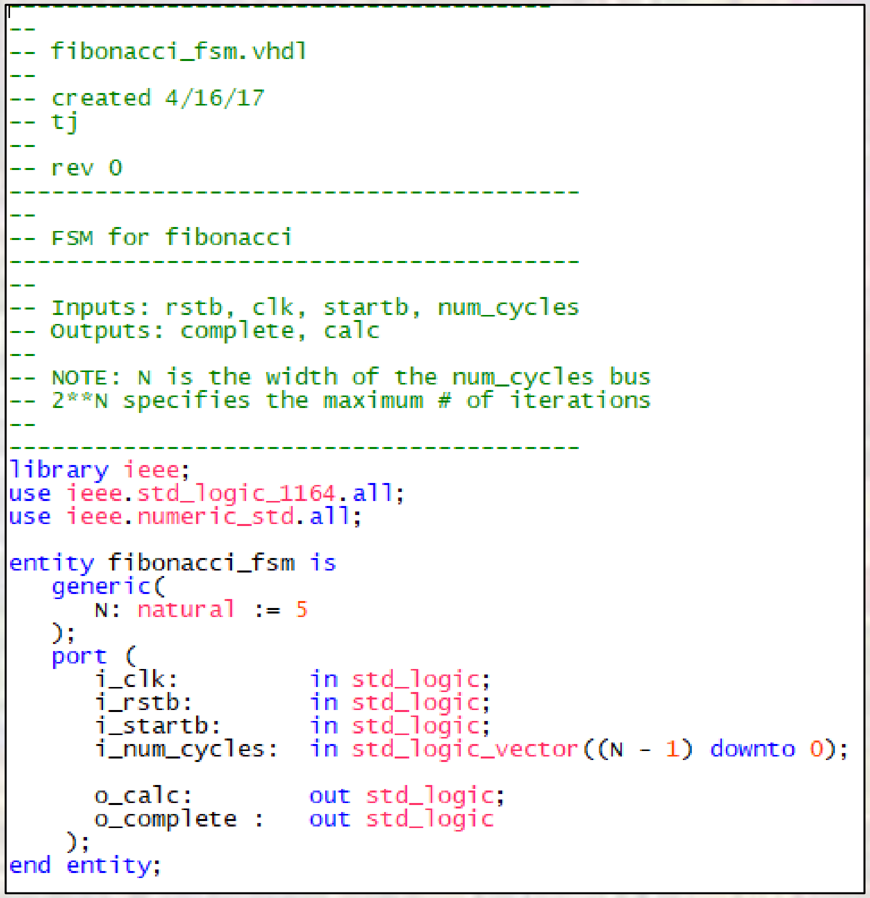
FSMD - Fibonacci

- ASM

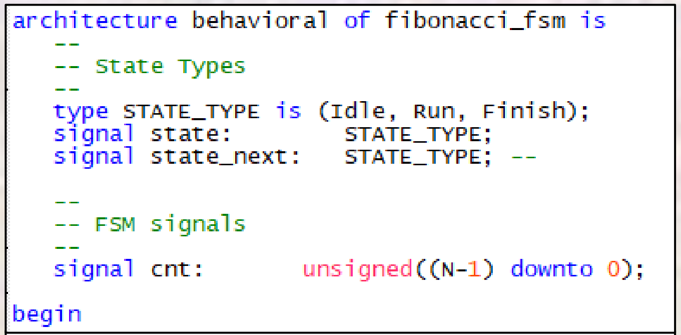


FSMD - Fibonacci

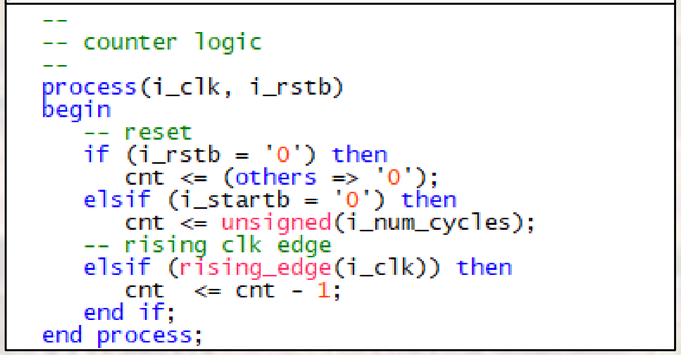
- HDL - FSM



```vhdl
-----------------------------------------
--
-- fibonacci_fsm.vhdl
--
-- created 4/16/17
-- tj
--
-- rev 0
-----------------------------------------
--
-- FSM for fibonacci
-----------------------------------------
--
-- Inputs: rstb, clk, startb, num_cycles
-- Outputs: complete, calc
--
-- NOTE: N is the width of the num_cycles bus
-- 2**N specifies the maximum # of iterations
--
-----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity fibonacci_fsm is
   generic(
      N: natural := 5
   );
   port (
      i_clk:         in std_logic;
      i_rstb:        in std_logic;
      i_startb:      in std_logic;
      i_num_cycles:  in std_logic_vector((N - 1) downto 0);

      o_calc:        out std_logic;
      o_complete :   out std_logic
    );
end entity;
```



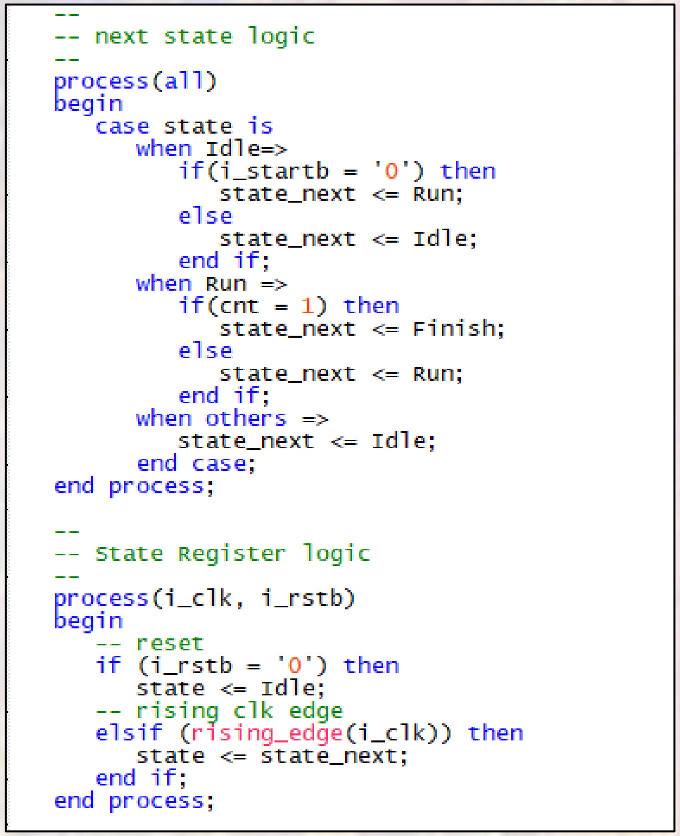
```vhdl
architecture behavioral of fibonacci_fsm is
   --
   -- State Types
   --
   type STATE_TYPE is (Idle, Run, Finish);
   signal state:       STATE_TYPE;
   signal state_next:  STATE_TYPE; --


   --
   -- FSM signals
   --
   signal cnt:        unsigned((N-1) downto 0);

begin
```



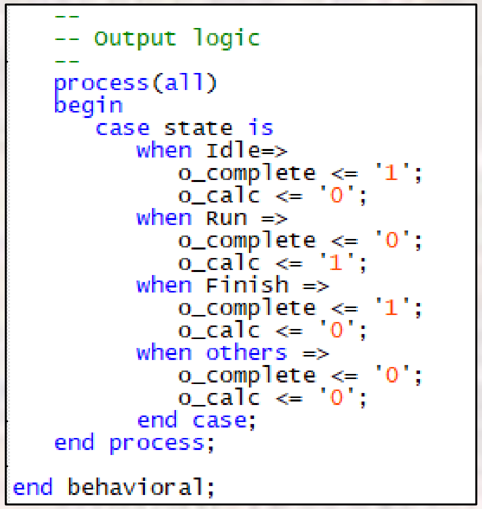
```vhdl
  --
  -- counter logic
  --
  process(i_clk, i_rstb)
  begin
    -- reset
    if (i_rstb = '0') then
       cnt <= (others => '0');
    elsif (i_startb = '0') then
       cnt <= unsigned(i_num_cycles);
    -- rising clk edge
    elsif (rising_edge(i_clk)) then
       cnt  <= cnt - 1;
    end if;
  end process;
```

FSMD - Fibonacci

- HDL - FSM



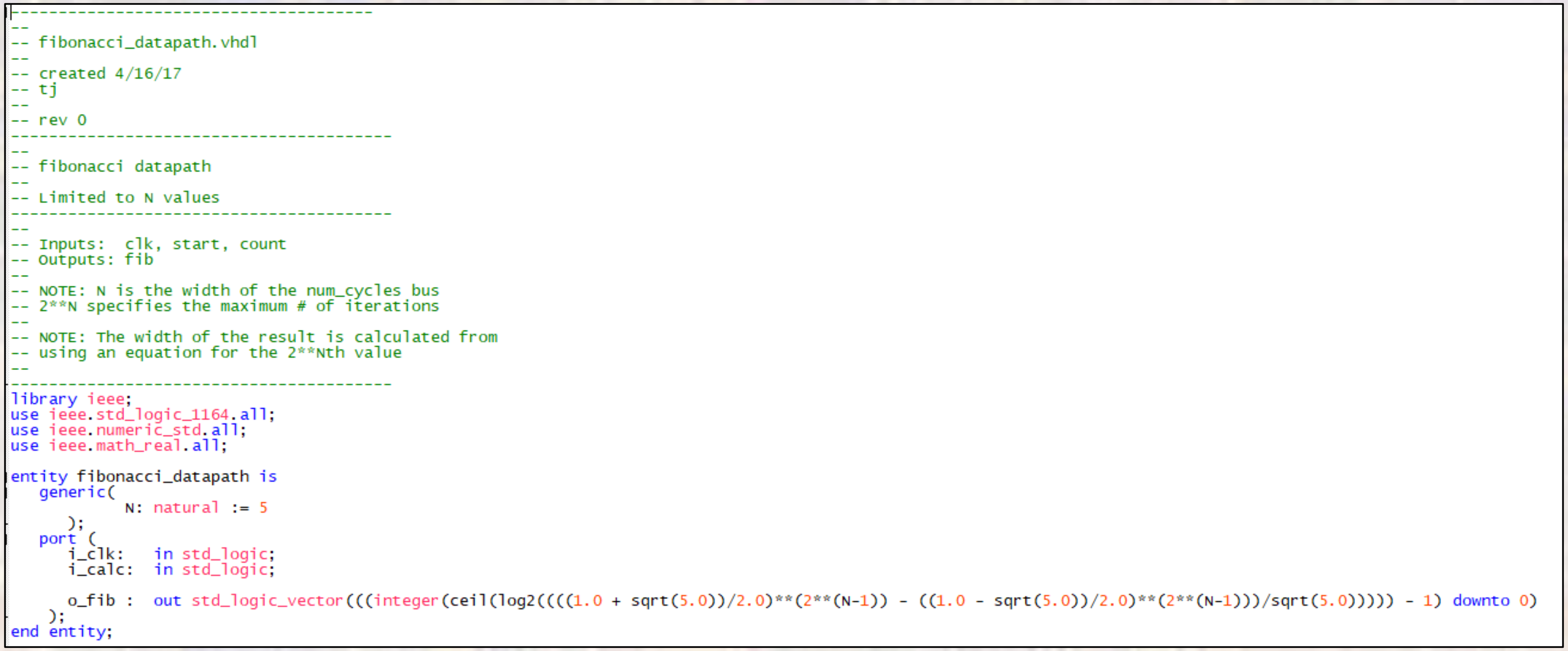
```vhdl
--
-- next state logic
--
process(all)
begin
   case state is
      when Idle=>
         if(i_startb = '0') then
            state_next <= Run;
         else
            state_next <= Idle;
         end if;
      when Run =>
         if(cnt = 1) then
            state_next <= Finish;
         else
            state_next <= Run;
         end if;
      when others =>
         state_next <= Idle;
      end case;
end process;

--
-- State Register logic
--
process(i_clk, i_rstb)
begin
   -- reset
   if (i_rstb = '0') then
      state <= Idle;
   -- rising clk edge
   elsif (rising_edge(i_clk)) then
      state <= state_next;
   end if;
end process;
```



```vhdl
   --
   -- Output logic
   --
   process(all)
   begin
      case state is
         when Idle=>
            o_complete <= '1';
            o_calc <= '0';
         when Run =>
            o_complete <= '0';
            o_calc <= '1';
         when Finish =>
            o_complete <= '1';
            o_calc <= '0';
         when others =>
            o_complete <= '0';
            o_calc <= '0';
         end case;
   end process;

end behavioral;
```

FSMD - Fibonacci

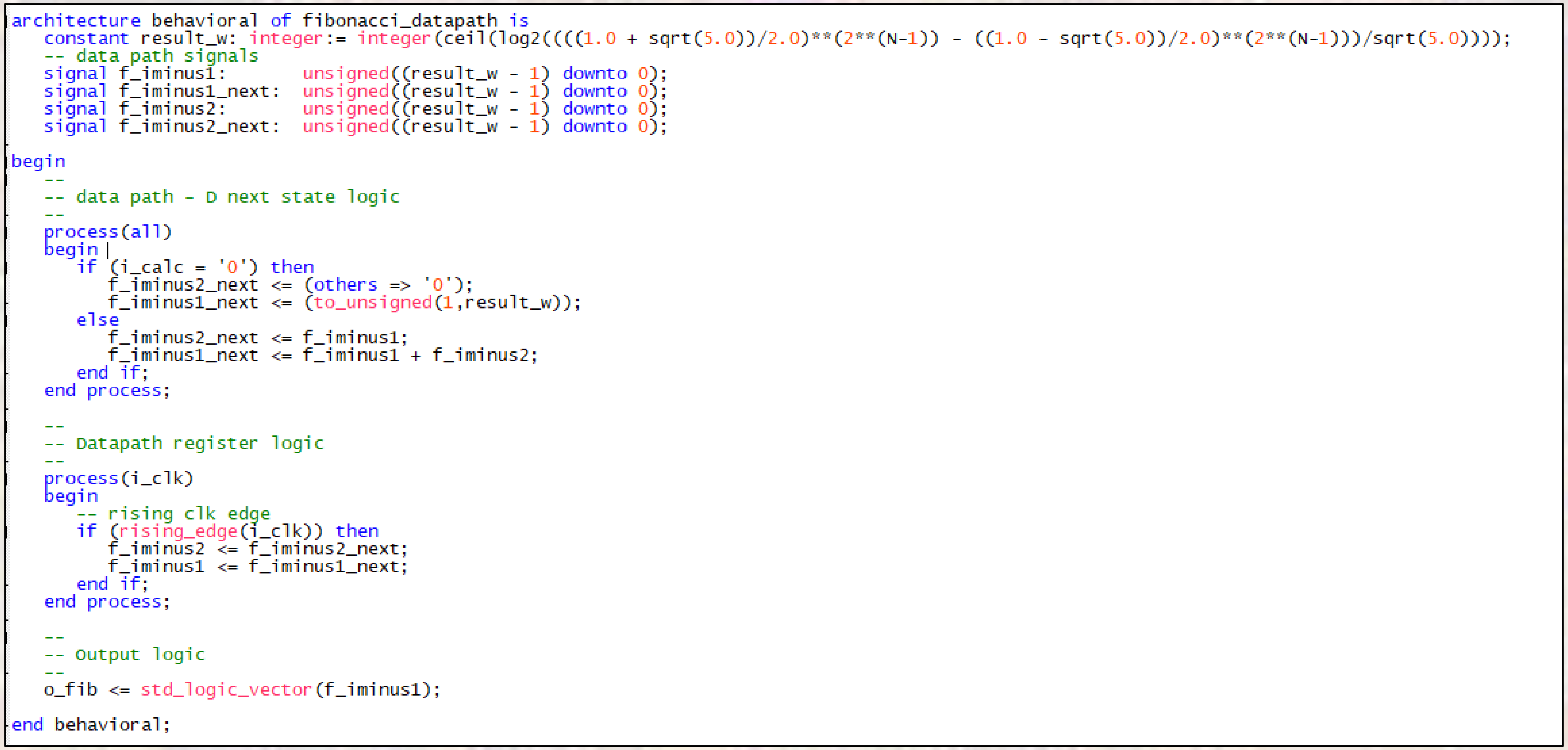
- HDL - data path



```vhdl
-----------------------------------------
--
-- fibonacci_datapath.vhdl
--
-- created 4/16/17
-- tj
--
-- rev 0
-----------------------------------------
--
-- fibonacci datapath
--
-- Limited to N values
-----------------------------------------
--
-- Inputs:  clk, start, count
-- Outputs: fib
--
-- NOTE: N is the width of the num_cycles bus
-- 2**N specifies the maximum # of iterations
--
-- NOTE: The width of the result is calculated from
-- using an equation for the 2**Nth value
--
-----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity fibonacci_datapath is
    generic(
              N: natural  := 5
         );
    port (
        i_clk:   in std_logic;
        i_calc:  in std_logic;

        o_fib :  out std_logic_vector(((integer(ceil(log2((((1.0 + sqrt(5.0))/2.0)**(2**(N-1)) - ((1.0 - sqrt(5.0))/2.0)**(2**(N-1)))/sqrt(5.0))))) - 1) downto 0)
      );
end entity;
```

FSMD - Fibonacci

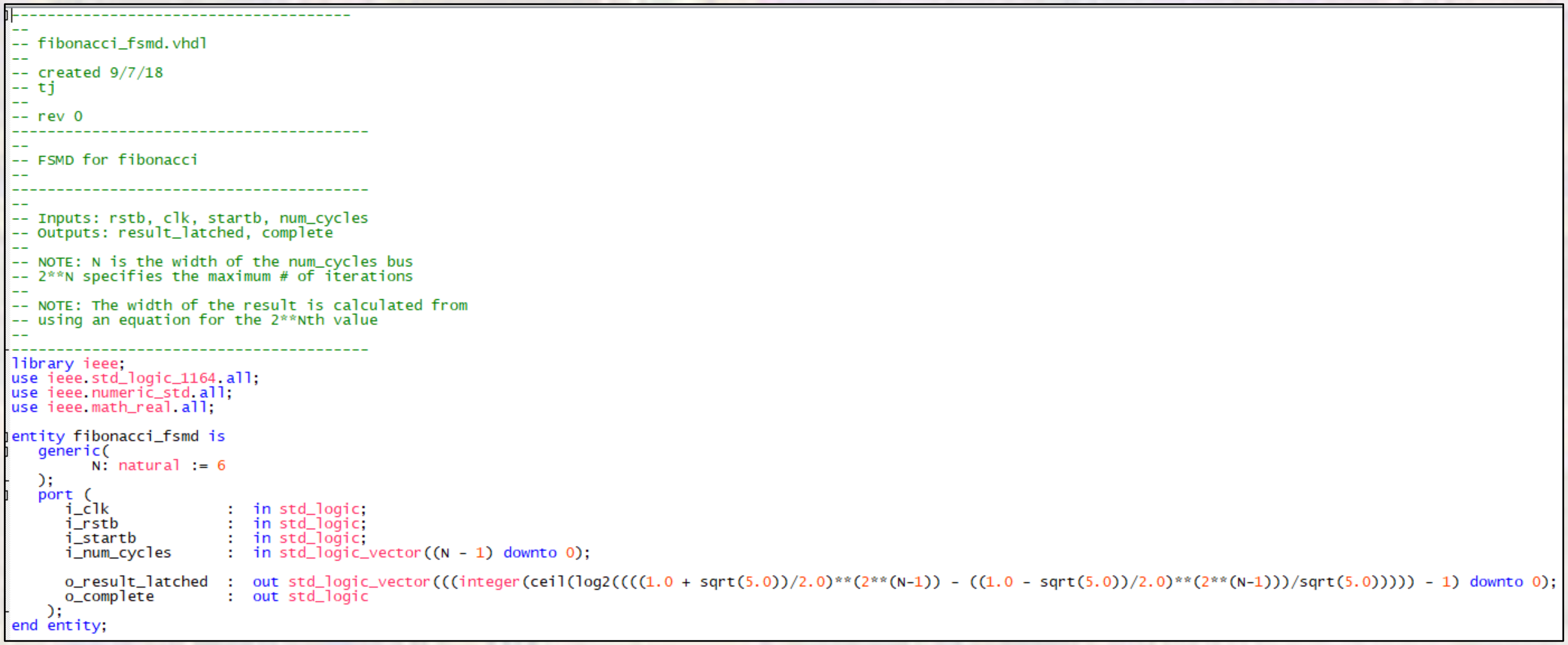
- HDL- data path



```vhdl
architecture behavioral of fibonacci_datapath is
   constant result_w: integer:= integer(ceil(log2((((1.0 + sqrt(5.0))/2.0)**(2**(N-1)) - ((1.0 - sqrt(5.0))/2.0)**(2**(N-1)))/sqrt(5.0))));
   -- data path signals
   signal f_iminus1:       unsigned((result_w - 1) downto 0);
   signal f_iminus1_next:  unsigned((result_w - 1) downto 0);
   signal f_iminus2:       unsigned((result_w - 1) downto 0);
   signal f_iminus2_next:  unsigned((result_w - 1) downto 0);

begin
   --
   -- data path - D next state logic
   --
   process(all)
   begin
      if (i_calc = '0') then
         f_iminus2_next <= (others => '0');
         f_iminus1_next <= (to_unsigned(1,result_w));
      else
         f_iminus2_next <= f_iminus1;
         f_iminus1_next <= f_iminus1 + f_iminus2;
      end if;
   end process;

   --
   -- Datapath register logic
   --
   process(i_clk)
   begin
      -- rising clk edge
      if (rising_edge(i_clk)) then
         f_iminus2 <= f_iminus2_next;
         f_iminus1 <= f_iminus1_next;
      end if;
   end process;

   --
   -- Output logic
   --
   o_fib <= std_logic_vector(f_iminus1);

end behavioral;
```

FSMD - Fibonacci

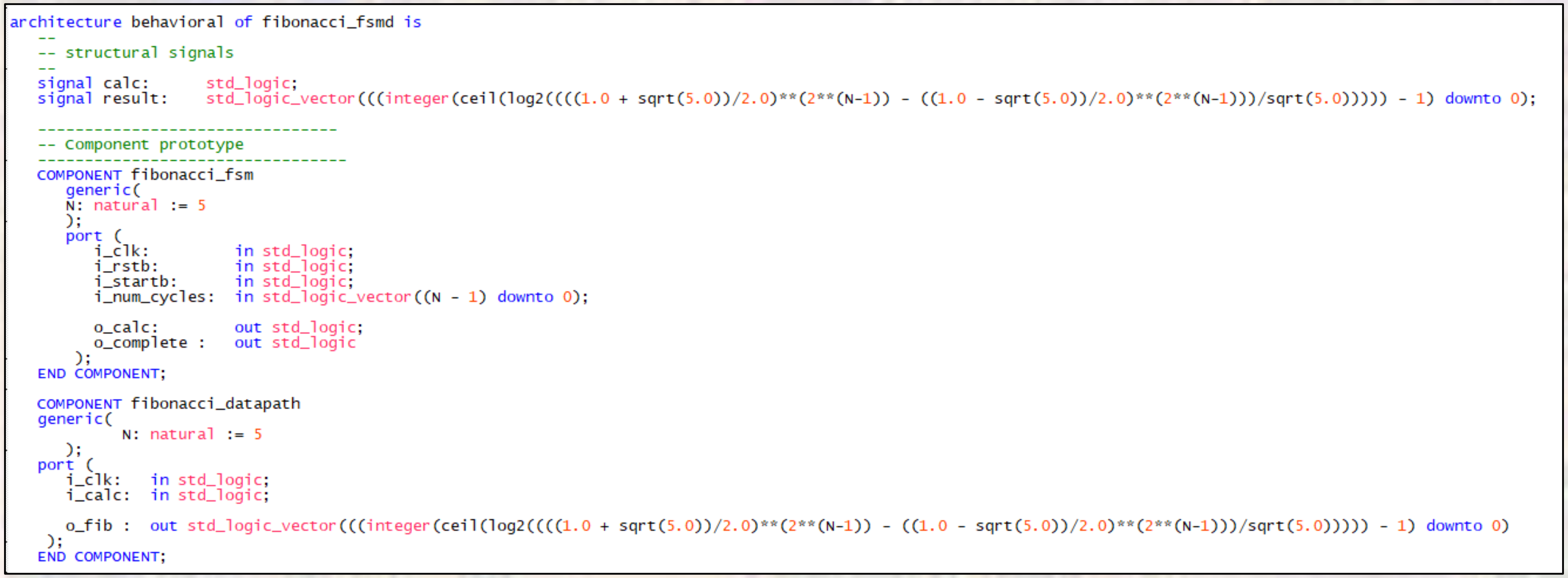
- HDL - FSMD



```
-----------------------------------------
--
-- fibonacci_fsmd.vhdl
--
-- created 9/7/18
-- tj
--
-- rev 0
-------------------------------------------
--
-- FSMD for fibonacci
--
-------------------------------------------
--
-- Inputs: rstb, clk, startb, num_cycles
-- Outputs: result_latched, complete
--
-- NOTE: N is the width of the num_cycles bus
-- 2**N specifies the maximum # of iterations
--
-- NOTE: The width of the result is calculated from
-- using an equation for the 2**Nth value
--
-------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.math_real.all;

entity fibonacci_fsmd is
   generic(
         N: natural  := 6
   );
   port (
      i_clk              :  in std_logic;
      i_rstb             :  in std_logic;
      i_startb           :  in std_logic;
      i_num_cycles       :  in std_logic_vector((N - 1) downto 0);

      o_result_latched   :  out std_logic_vector(((integer(ceil(log2((((1.0 + sqrt(5.0))/2.0)**(2**(N-1)) - ((1.0 - sqrt(5.0))/2.0)**(2**(N-1)))/sqrt(5.0))))) - 1) downto 0);
      o_complete         :  out std_logic
    );
end entity;
```

FSMD - Fibonacci

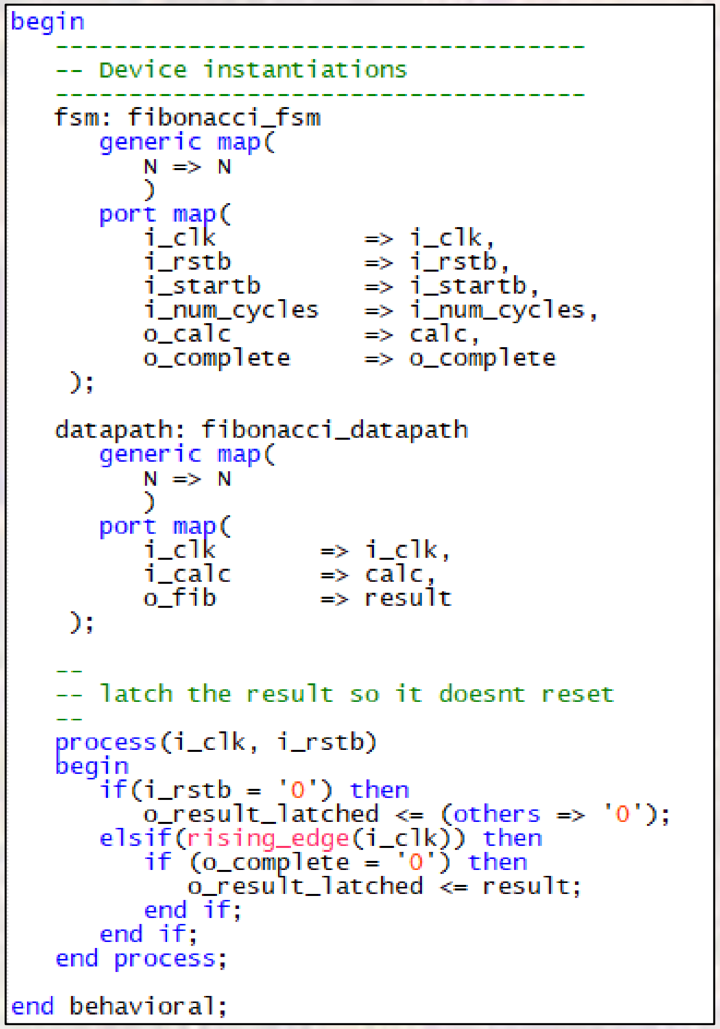
- HDL - FSMD



```vhdl
architecture behavioral of fibonacci_fsmd is
   --
   -- structural signals
   --
   signal calc:      std_logic;
   signal result:    std_logic_vector(((integer(ceil(log2((((1.0 + sqrt(5.0))/2.0)**(2**(N-1)) - ((1.0 - sqrt(5.0))/2.0)**(2**(N-1)))/sqrt(5.0))))) - 1) downto 0);

   --------------------------------
   -- Component prototype
   ---------------------------------
   COMPONENT fibonacci_fsm
      generic(
      N: natural  := 5
      );
      port (
         i_clk:          in std_logic;
         i_rstb:         in std_logic;
         i_startb:       in std_logic;
         i_num_cycles:   in std_logic_vector((N - 1) downto 0);

         o_calc:         out std_logic;
         o_complete :    out std_logic
       );
   END COMPONENT;

   COMPONENT fibonacci_datapath
   generic(
         N: natural  := 5
      );
   port (
      i_clk:   in std_logic;
      i_calc:  in std_logic;

      o_fib :  out std_logic_vector(((integer(ceil(log2((((1.0 + sqrt(5.0))/2.0)**(2**(N-1)) - ((1.0 - sqrt(5.0))/2.0)**(2**(N-1)))/sqrt(5.0))))) - 1) downto 0)
    );
   END COMPONENT;
```

FSMD - Fibonacci

- HDL - FSMD

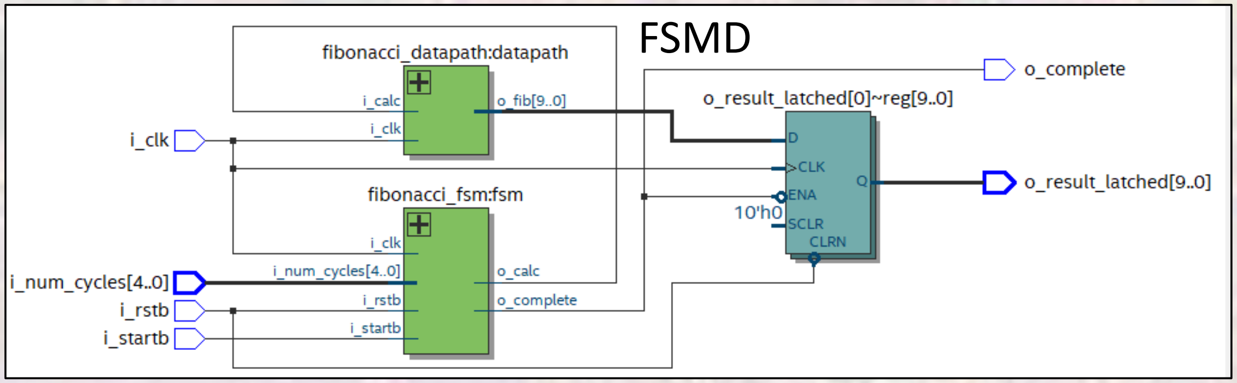


```
begin
   ------------------------------------
   -- Device instantiations
   ------------------------------------
   fsm: fibonacci_fsm
      generic map(
         N => N
         )
      port map(
         i_clk           => i_clk,
         i_rstb          => i_rstb,
         i_startb        => i_startb,
         i_num_cycles    => i_num_cycles,
         o_calc          => calc,
         o_complete      => o_complete
    );

   datapath: fibonacci_datapath
      generic map(
         N => N
         )
      port map(
         i_clk        => i_clk,
         i_calc       => calc,
         o_fib        => result
    );

   --
   -- latch the result so it doesnt reset
   --
   process(i_clk, i_rstb)
   begin
      if(i_rstb = '0') then
         o_result_latched <= (others => '0');
      elsif(rising_edge(i_clk)) then
         if (o_complete = '0') then
            o_result_latched <= result;
         end if;
      end if;
   end process;

end behavioral;
```

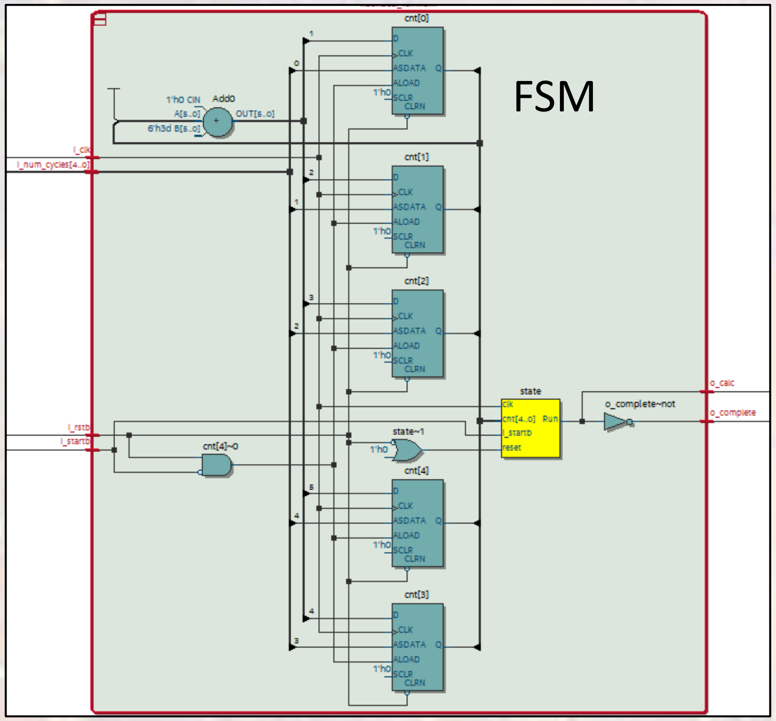
FSMD - Fibonacci

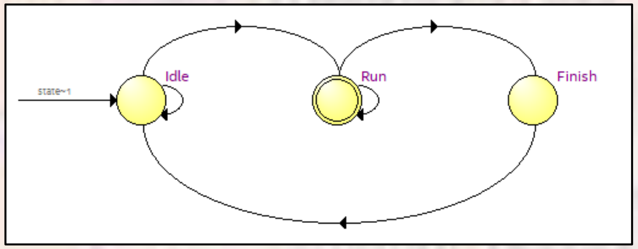
- RTL

FSMD

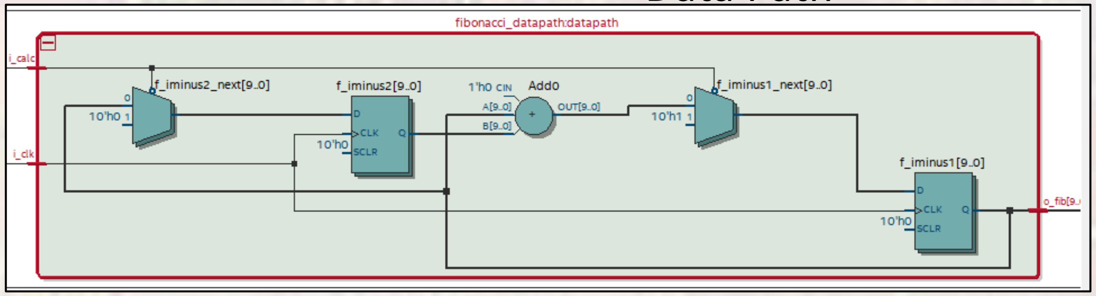


FSM





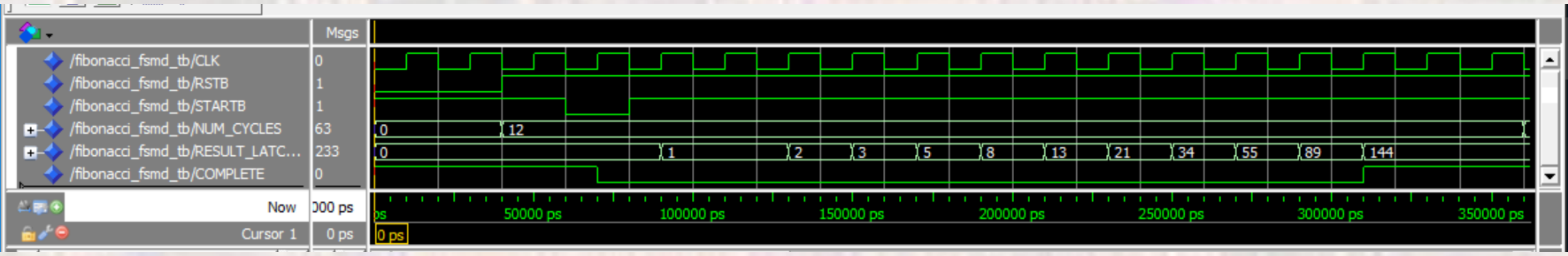
Data Path

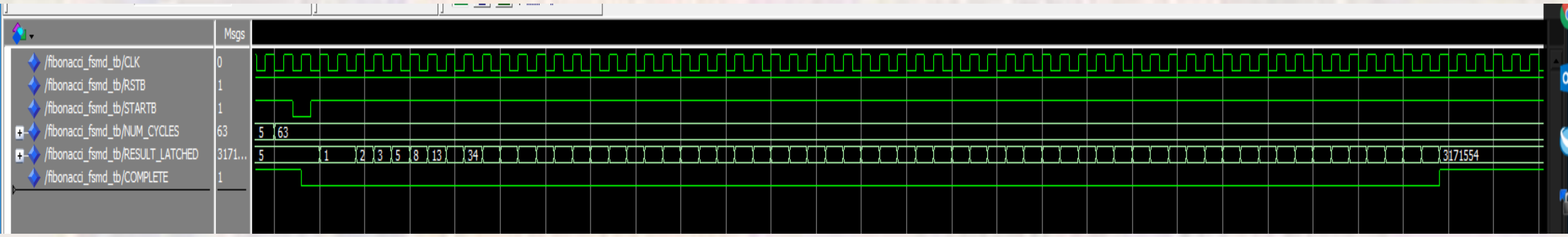


FSMD - Fibonacci

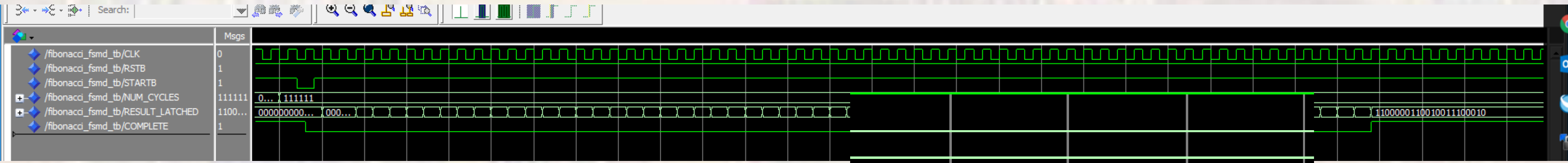
- Verification

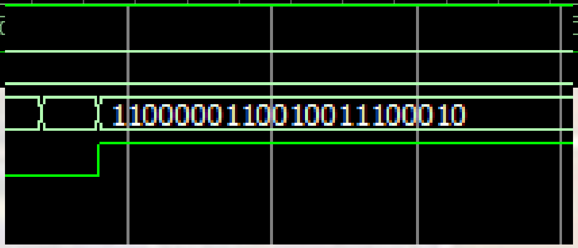
Results and intermediate values


Msgs
/fibonacci_fsmd_tb/CLK 0
/fibonacci_fsmd_tb/RSTB 1
/fibonacci_fsmd_tb/STARTB 1
/fibonacci_fsmd_tb/NUM_CYCLES 63
/fibonacci_fsmd_tb/RESULT_LATC... 233
/fibonacci_fsmd_tb/COMPLETE 0
0 12
0 1 2 3 5 8 13 21 34 55 89 144
Now 000 ps
Cursor 1 0 ps
ps 50000 ps 100000 ps 150000 ps 200000 ps 250000 ps 300000 ps 350000 ps
0 ps


Msgs
/fibonacci_fsmd_tb/CLK 0
/fibonacci_fsmd_tb/RSTB 1
/fibonacci_fsmd_tb/STARTB 1
/fibonacci_fsmd_tb/NUM_CYCLES 63
/fibonacci_fsmd_tb/RESULT_LATCHED 3171...
/fibonacci_fsmd_tb/COMPLETE 1
5 63
5 1 2 3 5 8 13 34 3171554

N = 6


Search:
Msgs
/fibonacci_fsmd_tb/CLK 0
/fibonacci_fsmd_tb/RSTB 1
/fibonacci_fsmd_tb/STARTB 1
/fibonacci_fsmd_tb/NUM_CYCLES 111111
/fibonacci_fsmd_tb/RESULT_LATCHED 1100...
/fibonacci_fsmd_tb/COMPLETE 1
0... 111111
000000000... 000...
1100000110010011100010



Indicates the correct vector size was calculated