FSMD
Multiplier

Last updated 1/28/21

FSMD Multiplier

- Multiplication
  - Want a process that is repetitive
    - Repetitive addition multiplication

5 x 4 = 5 + 5 + 5 + 5   i.e. the multiplicand added the multiplier times

    - Requires integers – or non-fractional binary numbers for the multiplier

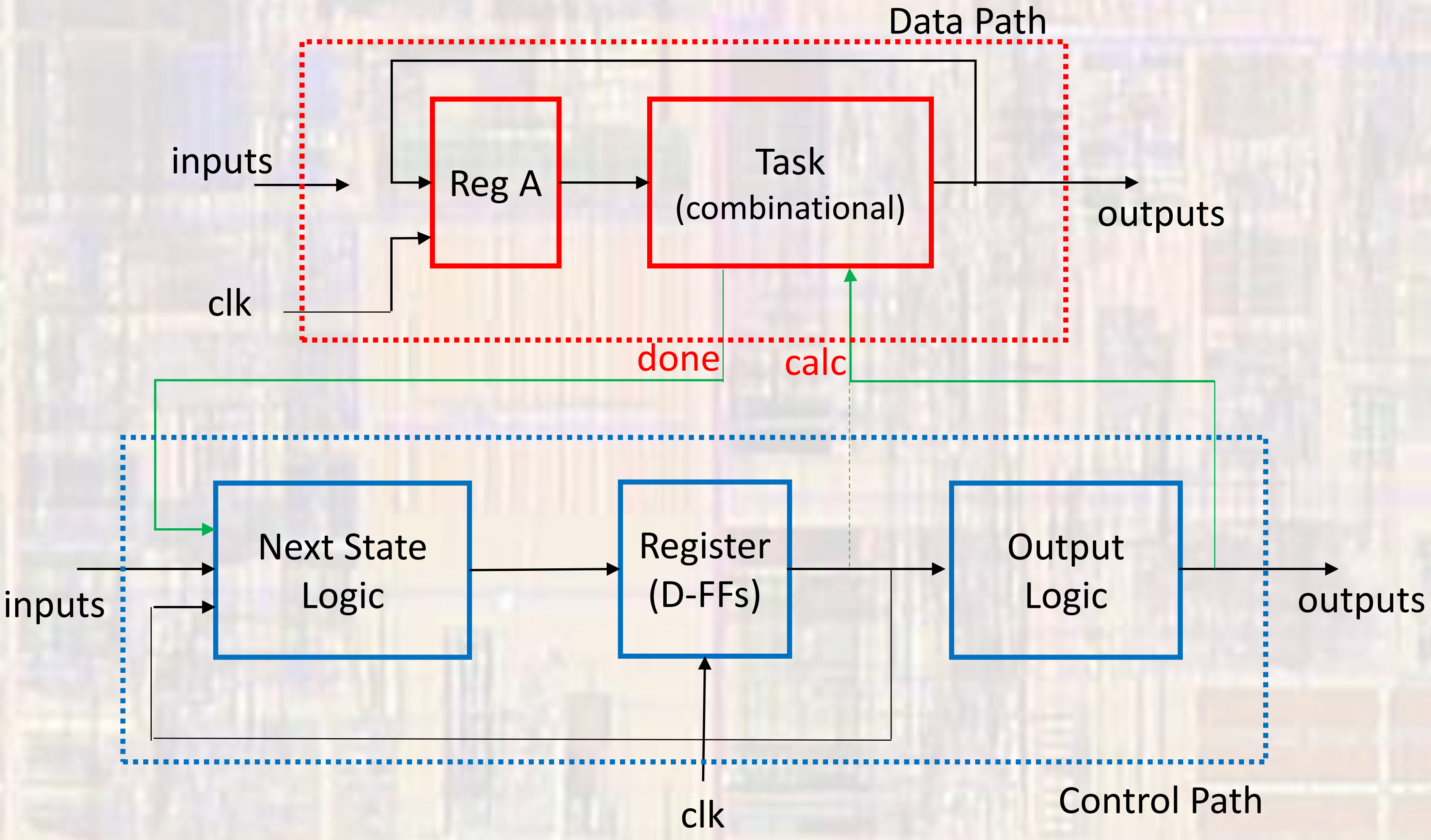
FSMD Multiplier

- Multiplication
  - Must sign extend to sum of # bits

```
  1110  ⇨    11111110           11111110
x 0011     x 00000011         x 00000011
                                11111110
                               11111110
                              00000000
                             00000000
                            00000000
                           00000000
                          00000000
                         00000000
                                11111010
```

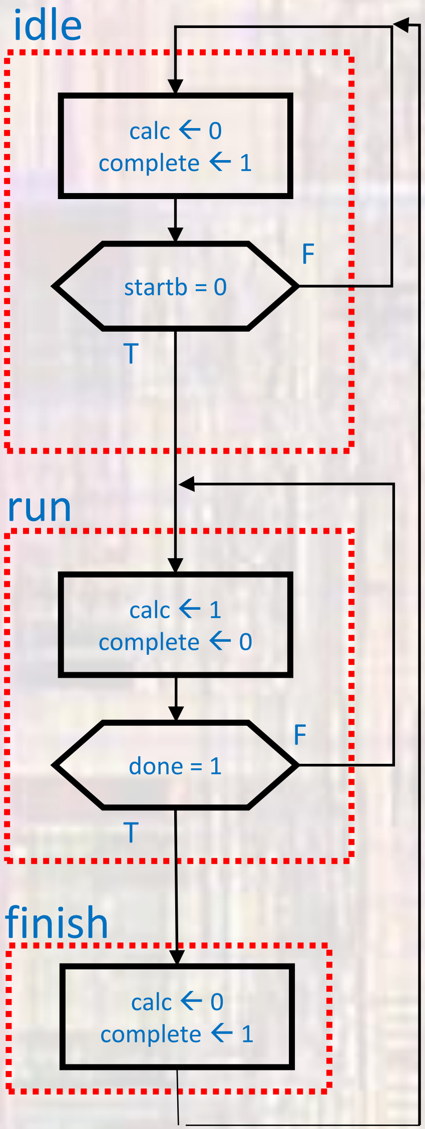
FSMD Multiplier

- Multiplier



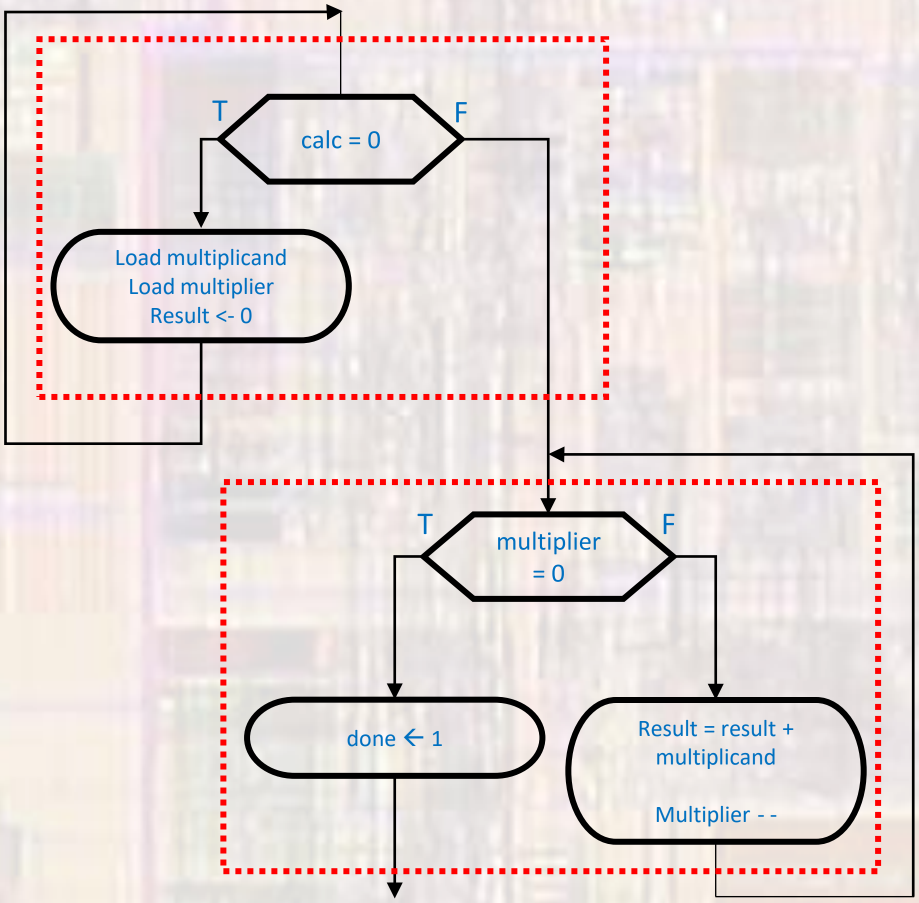
FSMD Multiplier

- Multiplier – ASM
  - Control Path



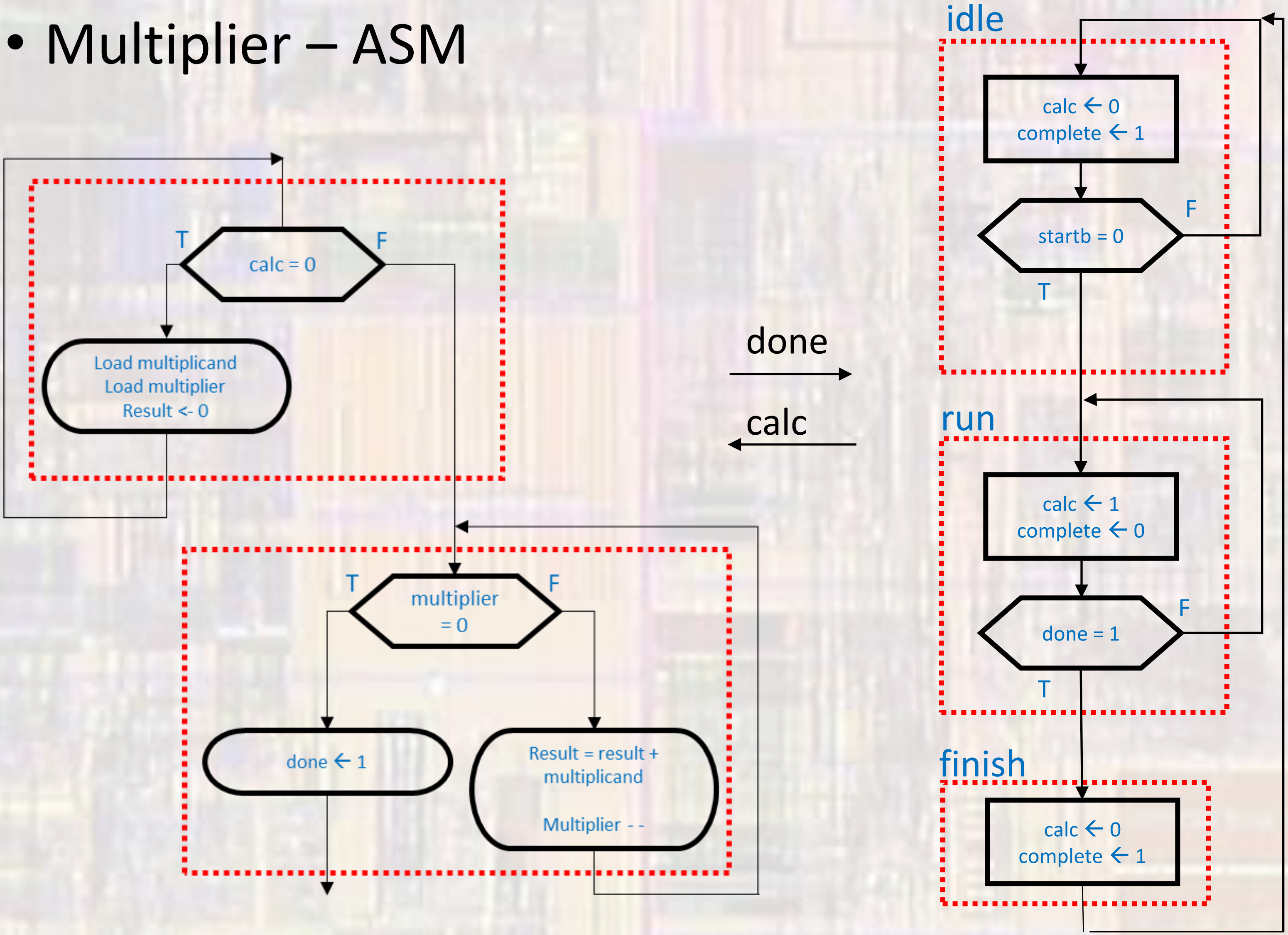
FSMD Multiplier

- Multiplier – ASM
  - Data Path



FSMD Multiplier

- Multiplier – ASM



calc = 0

T

F

Load multiplicand
Load multiplier
Result <- 0

multiplier = 0

T

F

done ← 1

Result = result + multiplicand

Multiplier - -

done

calc

idle

calc ← 0
complete ← 1

startb = 0

F

T

run

calc ← 1
complete ← 0

done = 1

F

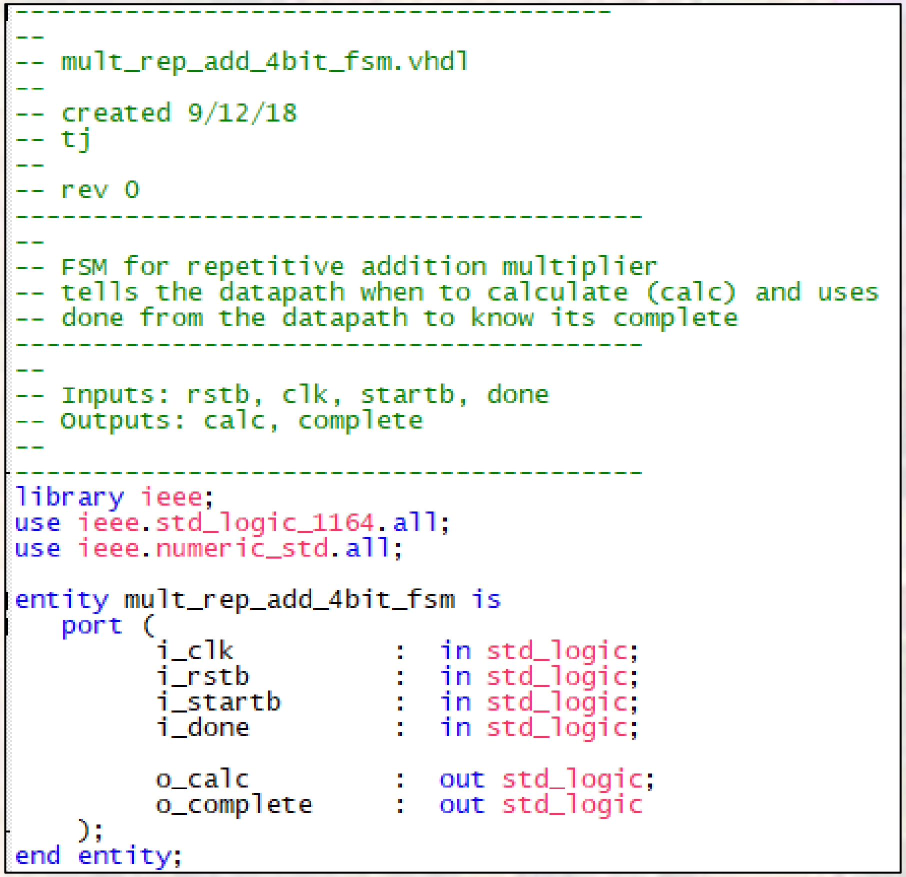
T

finish

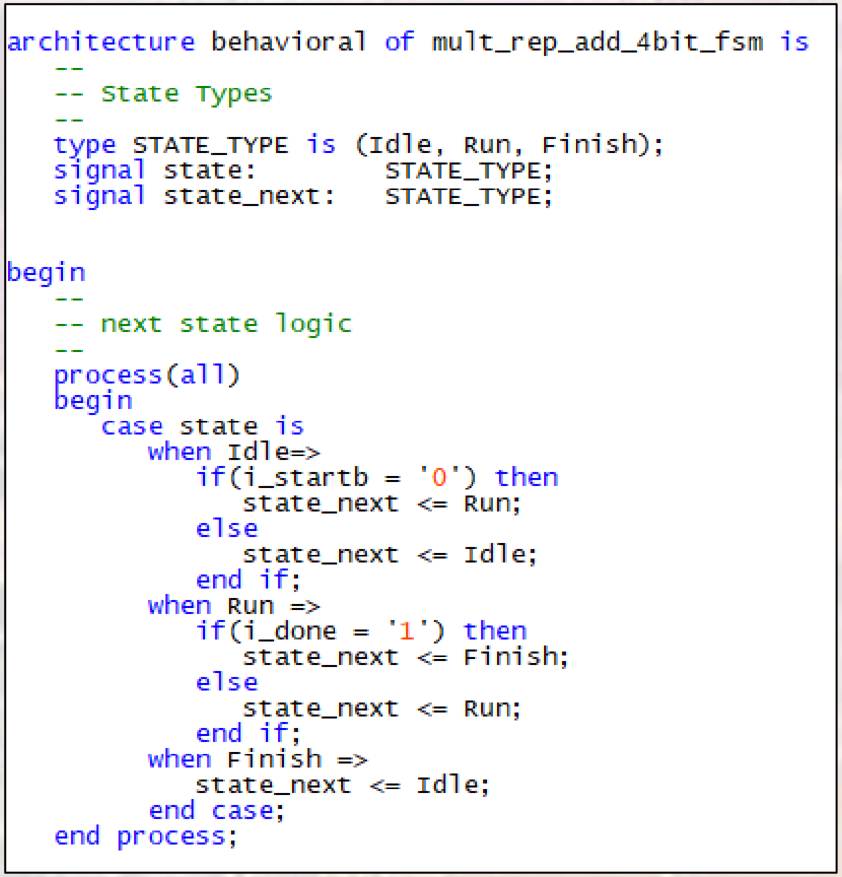
calc ← 0
complete ← 1

FSMD Multiplier

- Multiplier - FSM



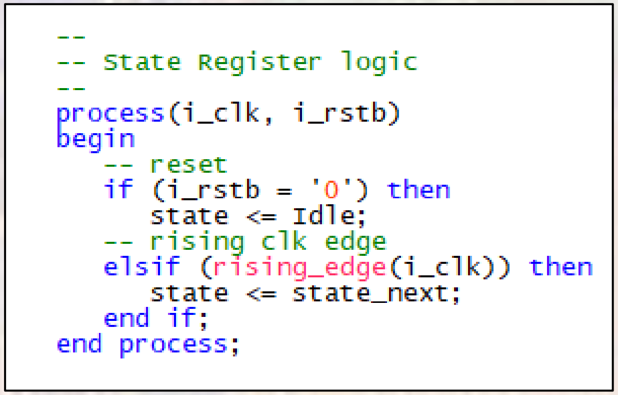
```vhdl
--------------------------------------
--
-- mult_rep_add_4bit_fsm.vhdl
--
-- created 9/12/18
-- tj
--
-- rev 0
----------------------------------------
--
-- FSM for repetitive addition multiplier
-- tells the datapath when to calculate (calc) and uses
-- done from the datapath to know its complete
----------------------------------------
--
-- Inputs: rstb, clk, startb, done
-- Outputs: calc, complete
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mult_rep_add_4bit_fsm is
   port (
         i_clk          :  in std_logic;
         i_rstb         :  in std_logic;
         i_startb       :  in std_logic;
         i_done         :  in std_logic;

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



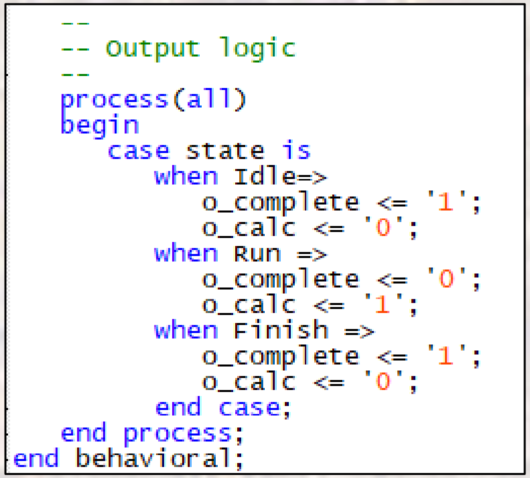
```vhdl
architecture behavioral of mult_rep_add_4bit_fsm is
   --
   -- State Types
   --
   type STATE_TYPE is (Idle, Run, Finish);
   signal state:       STATE_TYPE;
   signal state_next:  STATE_TYPE;


begin
   --
   -- 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(i_done = '1') then
               state_next <= Finish;
            else
               state_next <= Run;
            end if;
         when Finish =>
            state_next <= Idle;
      end case;
   end process;
```

FSMD Multiplier

- Multiplier - FSM



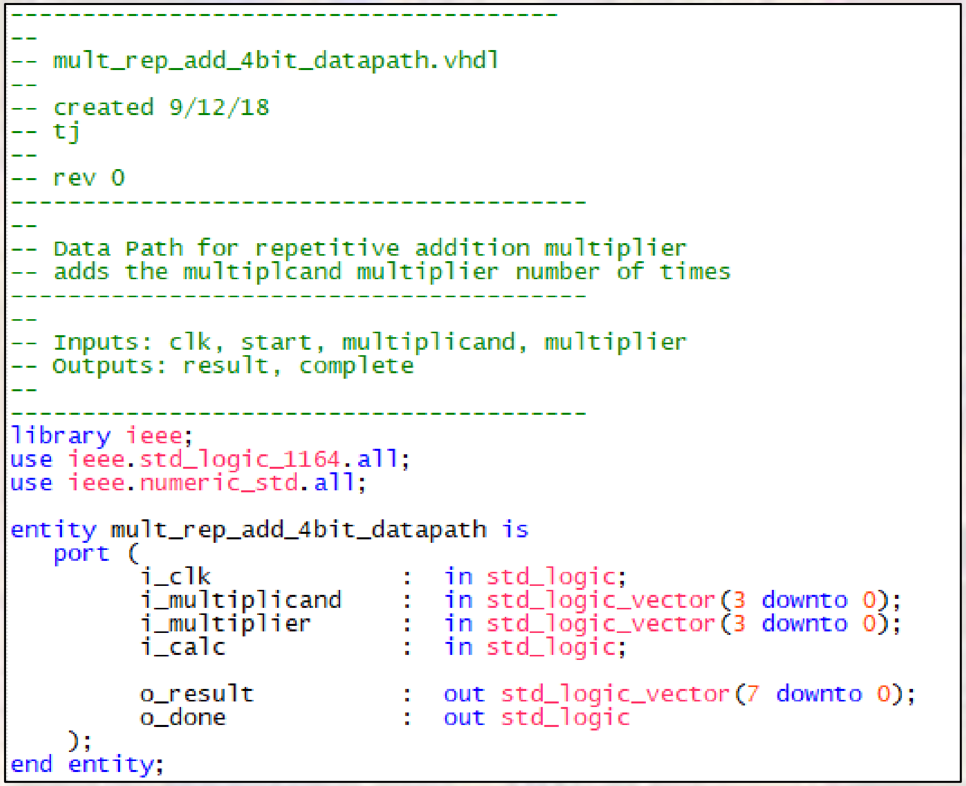
```
--
-- State Register logic
--
process(i_clk, i_rstb)
begin
   -- reset
   if (i_rstb = '0') then
      state <= Idle;
   -- rising clk edge
   elsif (rising_edge(i_clk)) then
      state <= state_next;
   end if;
end process;
```



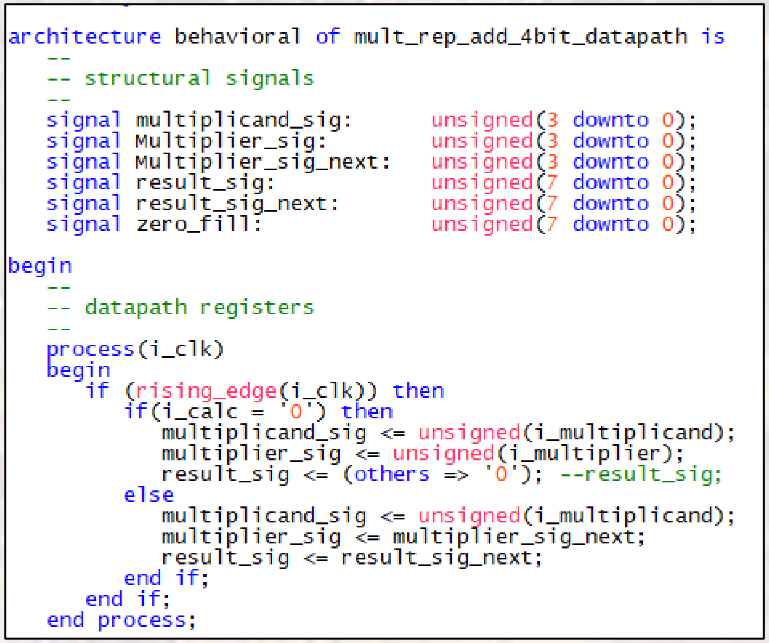
```
    --
    -- Output logic
    --
    process(all)
    begin
       case state is
          when Idle=>
             o_complete <= '1';
             o_calc <= '0';
          when Run =>
             o_complete <= '0';
             o_calc <= '1';
          when Finish =>
             o_complete <=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

FSMD Multiplier

- Multiplier – Data path



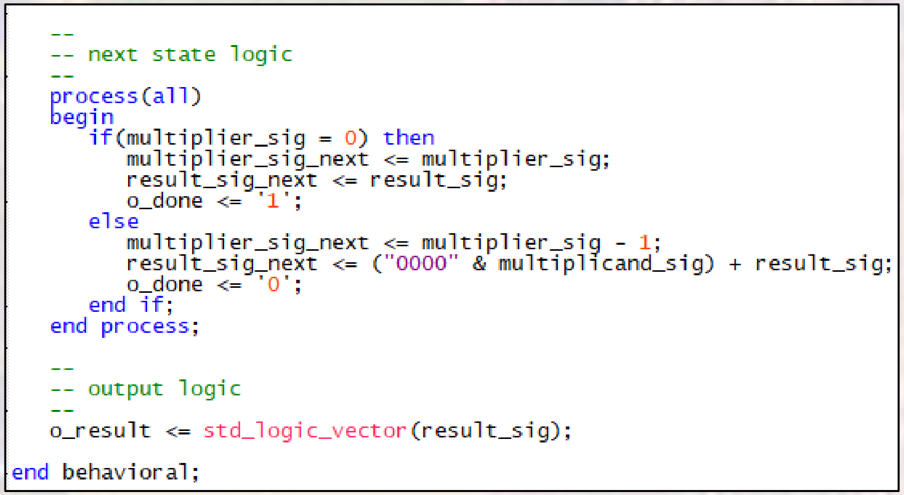
```vhdl
--------------------------------------
--
-- mult_rep_add_4bit_datapath.vhdl
--
-- created 9/12/18
-- tj
--
-- rev 0
----------------------------------------
--
-- Data Path for repetitive addition multiplier
-- adds the multiplcand multiplier number of times
----------------------------------------
--
-- Inputs: clk, start, multiplicand, multiplier
-- Outputs: result, complete
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mult_rep_add_4bit_datapath is
   port (
         i_clk             :  in std_logic;
         i_multiplicand    :  in std_logic_vector(3 downto 0);
         i_multiplier      :  in std_logic_vector(3 downto 0);
         i_calc            :  in std_logic;

         o_result          :  out std_logic_vector(7 downto 0);
         o_done            :  out std_logic
    );
end entity;
```



```vhdl
architecture behavioral of mult_rep_add_4bit_datapath is
   --
   -- structural signals
   --
   signal multiplicand_sig:      unsigned(3 downto 0);
   signal Multiplier_sig:        unsigned(3 downto 0);
   signal Multiplier_sig_next:   unsigned(3 downto 0);
   signal result_sig:            unsigned(7 downto 0);
   signal result_sig_next:       unsigned(7 downto 0);
   signal zero_fill:             unsigned(7 downto 0);

begin
   --
   -- datapath registers
   --
   process(i_clk)
   begin
      if (rising_edge(i_clk)) then
         if(i_calc = '0') then
            multiplicand_sig <= unsigned(i_multiplicand);
            multiplier_sig <= unsigned(i_multiplier);
            result_sig <= (others => '0'); --result_sig;
         else
            multiplicand_sig <= unsigned(i_multiplicand);
            multiplier_sig <= multiplier_sig_next;
            result_sig <= result_sig_next;
         end if;
      end if;
   end process;
```

FSMD Multiplier

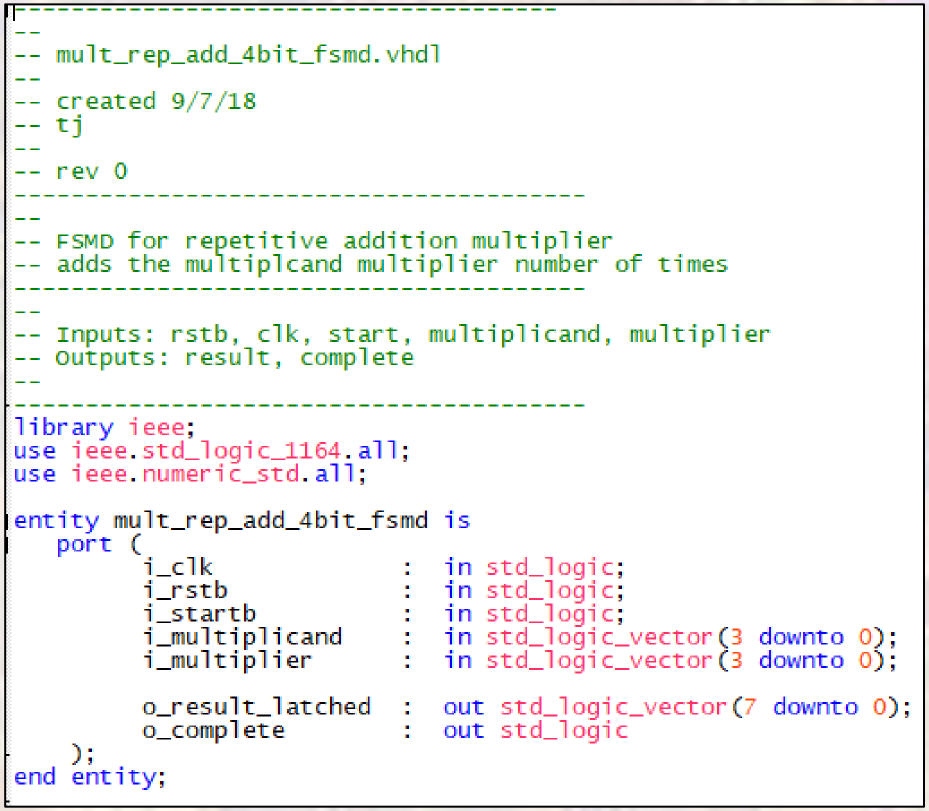
- Multiplier – Data path



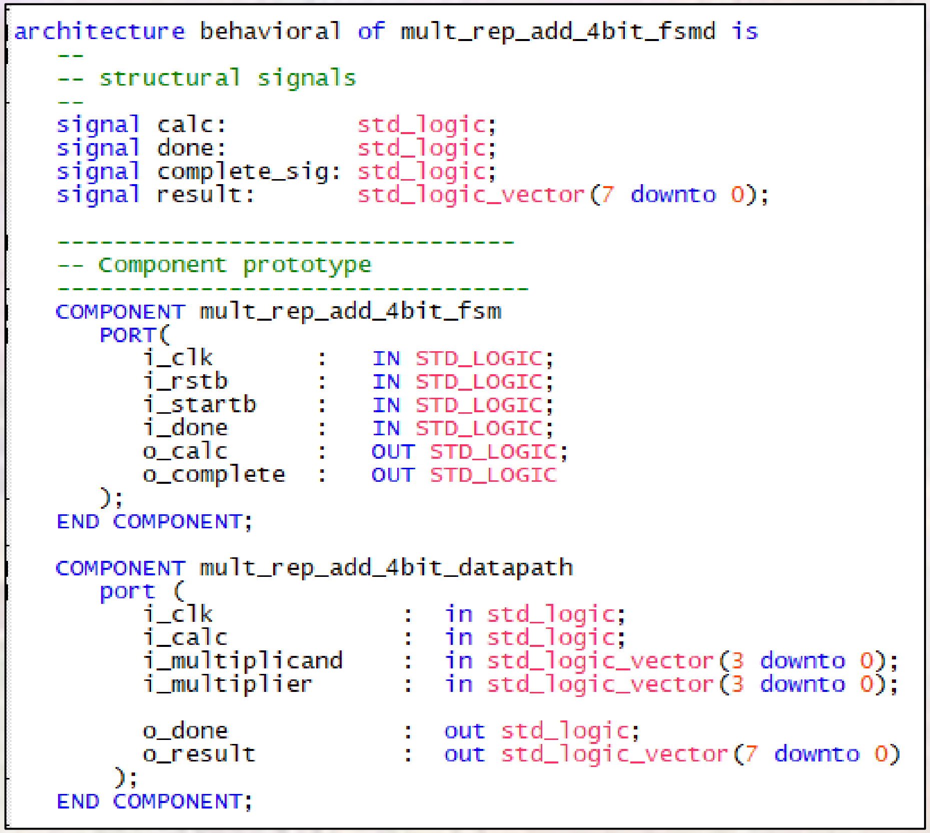
```
   --
   -- next state logic
   --
   process(all)
   begin
      if(multiplier_sig = 0) then
         multiplier_sig_next <= multiplier_sig;
         result_sig_next <= result_sig;
         o_done <= '1';
      else
         multiplier_sig_next <= multiplier_sig - 1;
         result_sig_next <= ("0000" & multiplicand_sig) + result_sig;
         o_done <= '0';
      end if;
   end process;

   --
   -- output logic
   --
   o_result <= std_logic_vector(result_sig);

end behavioral;
```

FSMD Multiplier

- Multiplier – FSMD



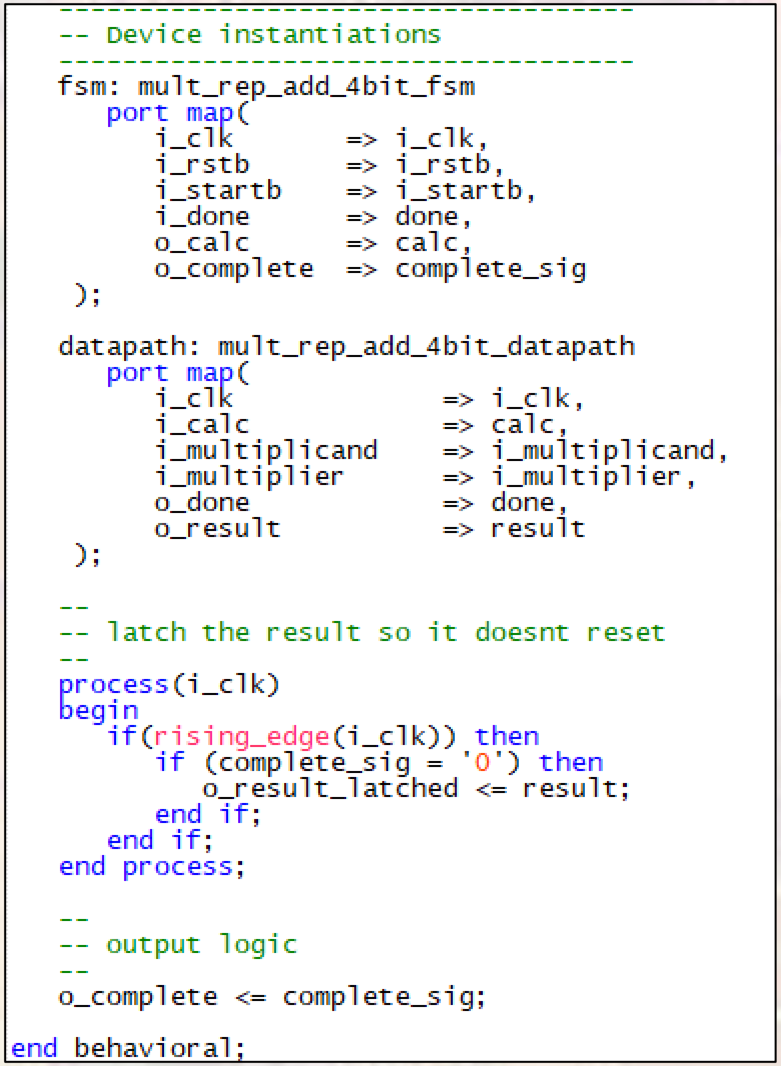
```
--------------------------------------
--
-- mult_rep_add_4bit_fsmd.vhdl
--
-- created 9/7/18
-- tj
--
-- rev 0
----------------------------------------
--
-- FSMD for repetitive addition multiplier
-- adds the multiplcand multiplier number of times
----------------------------------------
--
-- Inputs: rstb, clk, start, multiplicand, multiplier
-- Outputs: result, complete
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity mult_rep_add_4bit_fsmd is
   port (
         i_clk              :  in std_logic;
         i_rstb             :  in std_logic;
         i_startb           :  in std_logic;
         i_multiplicand     :  in std_logic_vector(3 downto 0);
         i_multiplier       :  in std_logic_vector(3 downto 0);

         o_result_latched   :  out std_logic_vector(7 downto 0);
         o_complete         :  out std_logic
    );
end entity;
```



```
architecture behavioral of mult_rep_add_4bit_fsmd is
   --
   -- structural signals
   --
   signal calc:          std_logic;
   signal done:          std_logic;
   signal complete_sig:  std_logic;
   signal result:        std_logic_vector(7 downto 0);

   --------------------------------
   -- Component prototype
   ---------------------------------
   COMPONENT mult_rep_add_4bit_fsm
      PORT(
         i_clk       :   IN STD_LOGIC;
         i_rstb      :   IN STD_LOGIC;
         i_startb    :   IN STD_LOGIC;
         i_done      :   IN STD_LOGIC;
         o_calc      :   OUT STD_LOGIC;
         o_complete  :   OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT mult_rep_add_4bit_datapath
      port (
         i_clk             :  in std_logic;
         i_calc            :  in std_logic;
         i_multiplicand    :  in std_logic_vector(3 downto 0);
         i_multiplier      :  in std_logic_vector(3 downto 0);

         o_done            :  out std_logic;
         o_result          :  out std_logic_vector(7 downto 0)
       );
   END COMPONENT;
```

FSMD Multiplier

- Multiplier – FSMD

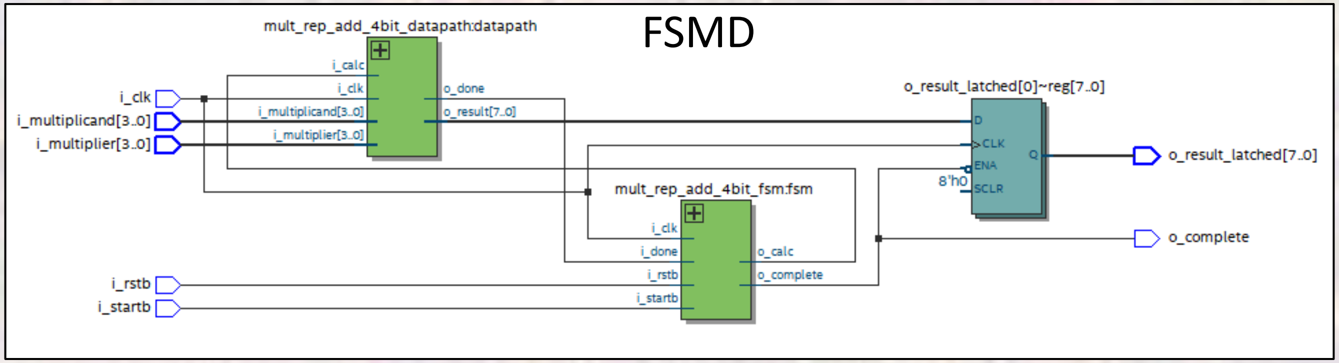


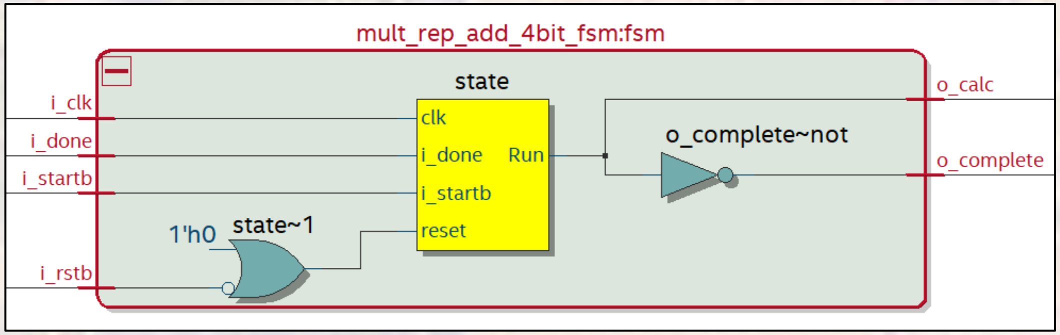
```
   ------------------------------------
   -- Device instantiations
   ------------------------------------
   fsm: mult_rep_add_4bit_fsm
      port map(
         i_clk        => i_clk,
         i_rstb       => i_rstb,
         i_startb     => i_startb,
         i_done       => done,
         o_calc       => calc,
         o_complete   => complete_sig
    );

   datapath: mult_rep_add_4bit_datapath
      port map(
         i_clk              => i_clk,
         i_calc             => calc,
         i_multiplicand     => i_multiplicand,
         i_multiplier       => i_multiplier,
         o_done             => done,
         o_result           => result
    );

   --
   -- latch the result so it doesnt reset
   --
   process(i_clk)
   begin
      if(rising_edge(i_clk)) then
         if (complete_sig = '0') then
            o_result_latched <= result;
         end if;
      end if;
   end process;

   --
   -- output logic
   --
   o_complete <= complete_sig;

end behavioral;
```

FSMD Multiplier

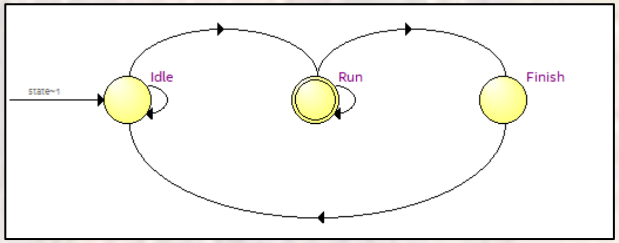
- RTL

FSMD

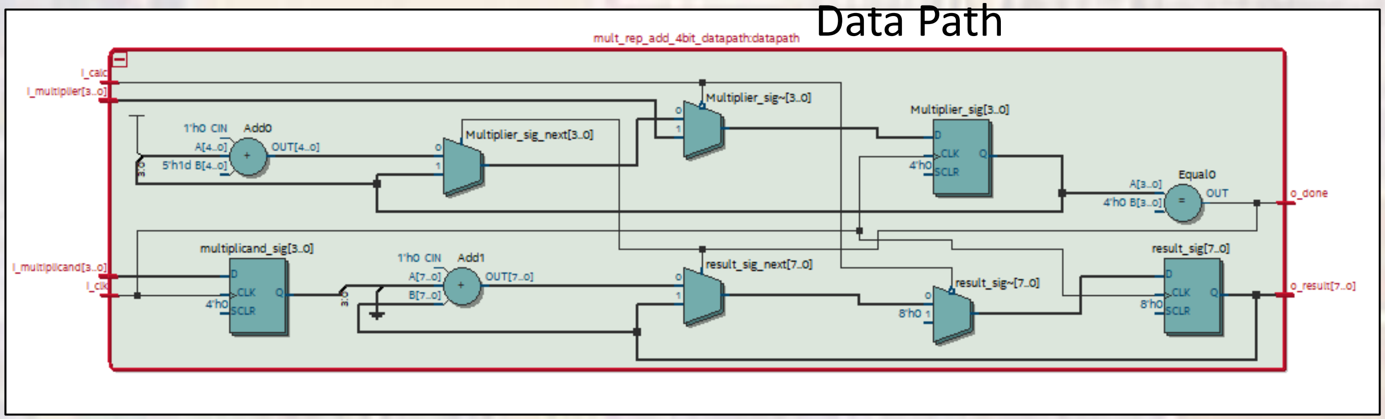




FSM



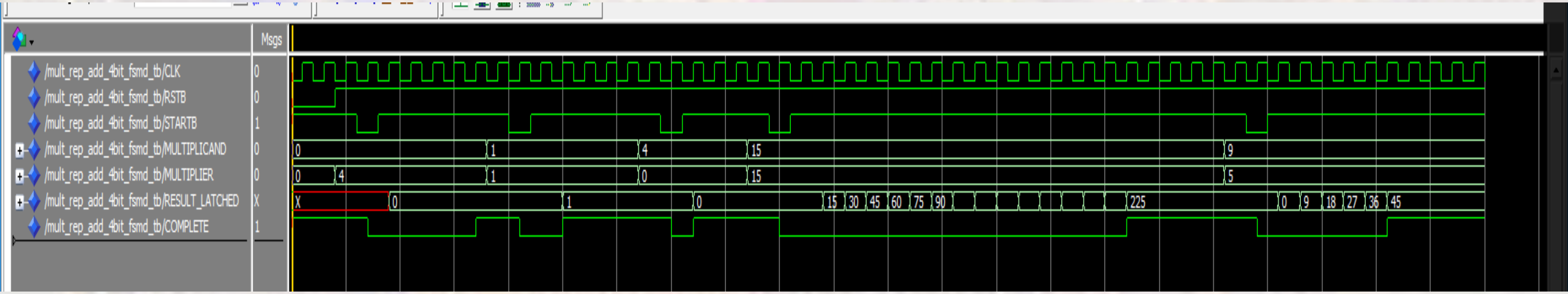
Data Path



FSMD Multiplier

- Verification

Results and intermediate values


Msgs
/mult_rep_add_4bit_fsmd_tb/CLK 0
/mult_rep_add_4bit_fsmd_tb/RSTB 0
/mult_rep_add_4bit_fsmd_tb/STARTB 1
/mult_rep_add_4bit_fsmd_tb/MULTIPLICAND 0
/mult_rep_add_4bit_fsmd_tb/MULTIPLIER 0
/mult_rep_add_4bit_fsmd_tb/RESULT_LATCHED X
/mult_rep_add_4bit_fsmd_tb/COMPLETE 1
0
1
4
15
9
0
4
1
0
15
5
X
0
1
0
15
30
45
60
75
90
225
0
9
18
27
36
45

Note: Takes “multiplier” number of clock cycles

Takes a variable number of clock cycles