HDL FSM Intro
Direct Implementation

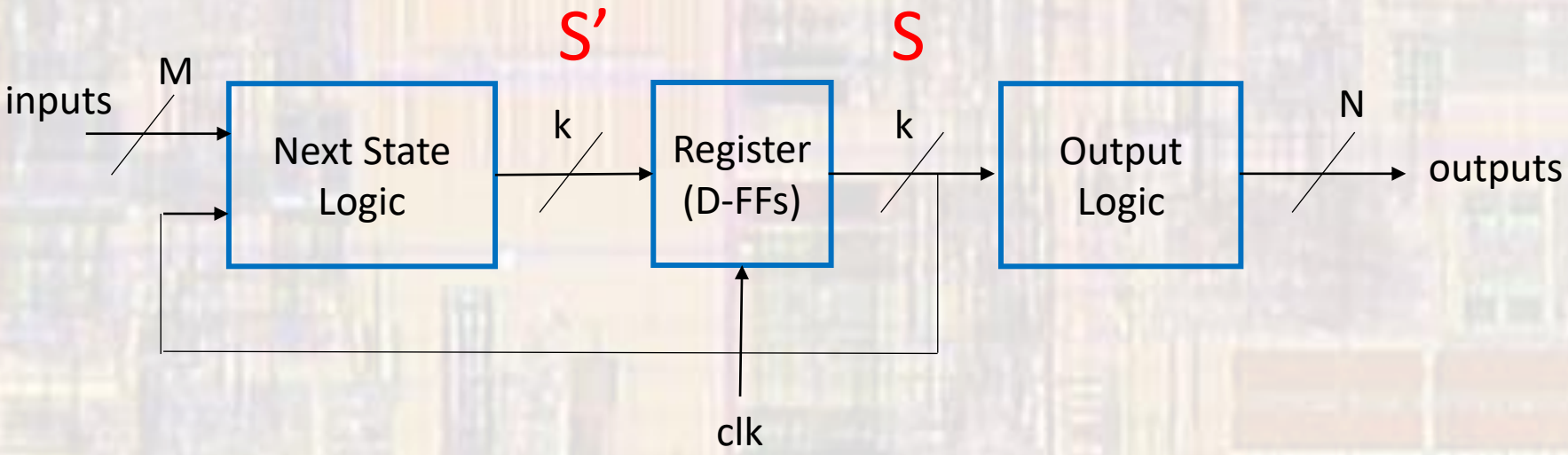
Last updated 2/18/21

HDL FSM Intro

- These slides introduce the basics of HDL based State Machines
- Upon completion: You should be able to create a direct implementation of a State Machine in VHDL

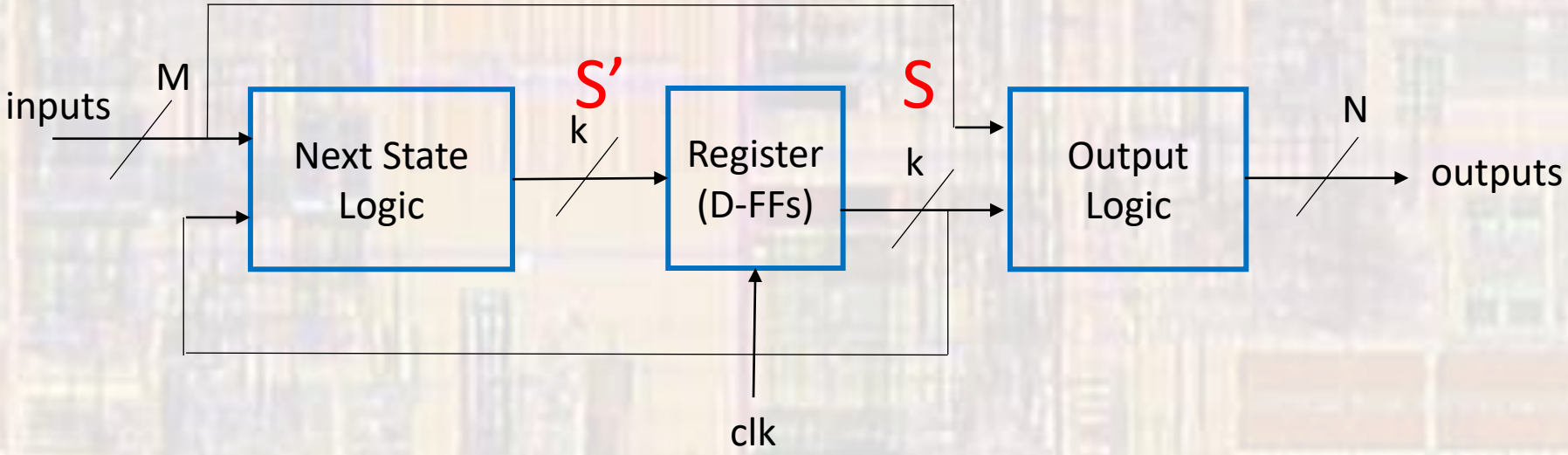
HDL FSM Intro

- Finite State Machine
  - Moore Machine
    - Outputs depend only on the current state(S)



HDL FSM Intro

- Finite State Machine
  - Mealy Machine
    - Outputs depend on the current state(S) and the inputs



HDL FSM Intro

- Mod 10 Counter

Note: this is for illustrative purpose - we would never make a counting state machine like this

HDL FSM Intro

- Mod 10 Counter
  - Next State Logic

Current State	Inputs	Next State
A		B
B		C
C		D
D		E
E		F
F		G
G		H
H		I
I		J
J		A

Binary Encoding

Current State	Inputs	Next State
0000		0001
0001		0010
0010		0011
0011		0100
0100		0101
0101		0110
0110		0111
0111		1000
1000		1001
1001		0000

HDL FSM Intro

- Mod 10 Counter
  - Output Logic

Current State	Output
A	0000
B	0001
C	0010
D	0011
E	0100
F	0101
G	0110
H	0111
I	1000
J	1001

Binary Encoding

Current State	Output
0000	0000
0001	0001
0010	0010
0011	0011
0100	0100
0101	0101
0110	0110
0111	0111
1000	1000
1001	1001

HDL FSM Intro

- FSM general structure - HDL

```
--
-- counter_mod10_fsm.vhdl
--
-- created 3/30/18
-- tj
--
-- rev 0
----------------------------------------
--
-- mod 10 up-counter - state machine
----------------------------------------
--
-- Inputs: i_rstb, i_clk
-- Outputs: o_cnt[3:0]
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_mod10_fsm is
    port (
        o_cnt :  out std_logic_vector(3 downto 0)
    );
end entity;
```

Entity declaration

```
architecture behavioral of counter_mod10_fsm is

    --
    -- in
    --
    signa                                    o 0);
    signal state_next:   unsigned(3 downto 0);

begin

    --
    -- next state logic
    --
    with state select
        state_next <=   "0001" when "0000",
                        "0010" when "0001",
                        "0011" when "0010",
                        "0100" when "0011",
                        "0101" when "0100",
                        "0110" when "0101",
                        "0111" when "0110",
                        "1000" when "0111",
                        "1001" when "1000",
                        "0000" when "1001",
                        "0000" when others;
```

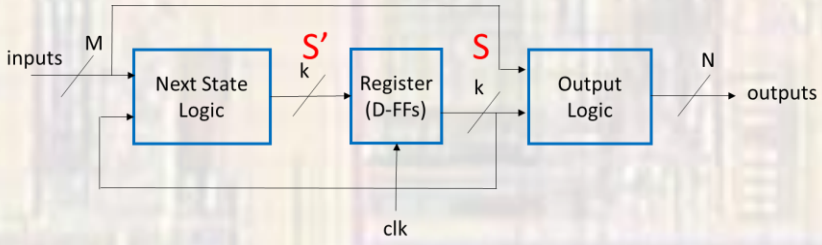
State signal declaration

Next state logic

```
    --
    -- Register logic
    --
    process(i_clk, i_rstb)
    begin
                                                    then
            state <= state_next;
        end if;
    end process;

    --
    -- Output logic
    --
    with state select
        o_cnt <= "0000" when "0000",
                 "0001" when "0001",
                 "0101" when "0101",
                 "0110" when "0110",
                 "0111" when "0111",
                 "1000" when "1000",
                 "1001" when "1001",
                 "0000" when others;

end behavioral;
```

State initialization and Update (register)

Output logic



HDL FSM Intro

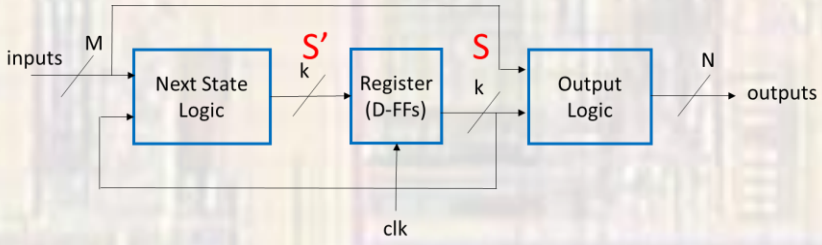
- Mod 10 Counter

Made these unsigned since they represent a count

```
--------------------------------------
--
-- counter_mod10_fsm.vhdl
--
-- created 3/30/18
-- tj
--
-- rev 0
--------------------------------------
--
-- mod 10 up-counter - state machine
--------------------------------------
--
-- Inputs: i_rstb, i_clk
-- Outputs: o_cnt[3:0]
--
--------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_mod10_fsm is
   port (
         i_clk :  in std_logic;
         i_rstb : in std_logic;

         o_cnt :  out std_logic_vector(3 downto 0)
    );
end entity;
```

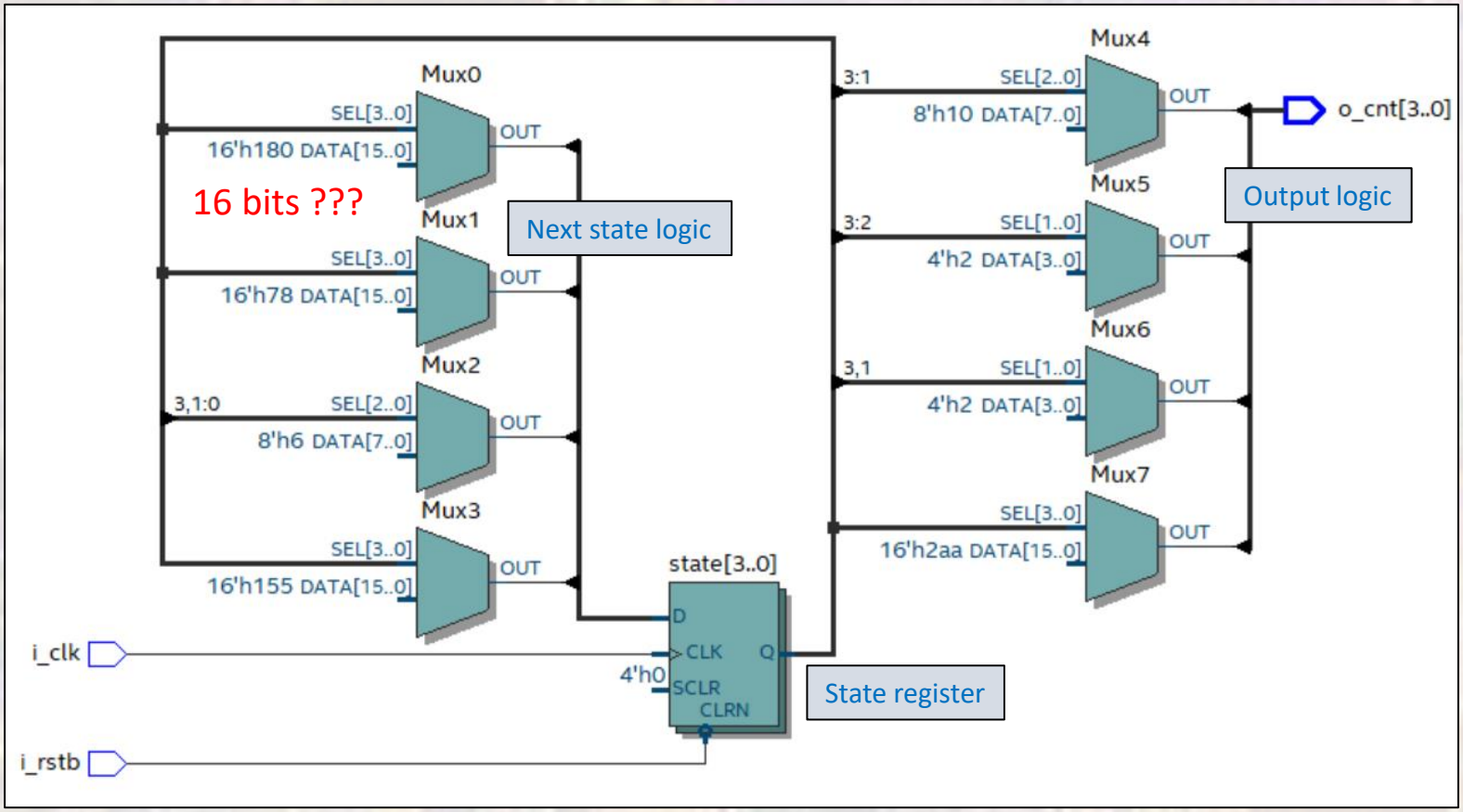
```
architecture behavioral of counter_mod10_fsm is

   --
   -- internal signals
   --
   signal state:        unsigned(3 downto 0);
   signal state_next:   unsigned(3 downto 0);

begin

   --
   -- next state logic
   --
   with state select
      state_next <=   "0001" when "0000",
                      "0010" when "0001",
                      "0011" when "0010",
                      "0100" when "0011",
                      "0101" when "0100",
                      "0110" when "0101",
                      "0111" when "0110",
                      "1000" when "0111",
                      "1001" when "1000",
                      "0000" when "1001",
                      "0000" when others;
```

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

   --
   -- Output logic
   --
   with state select
      o_cnt <= "0000" when "0000",
               "0001" when "0001",
               "0010" when "0010",
               "0011" when "0011",
               "0100" when "0100",
               "0101" when "0101",
               "0110" when "0110",
               "0111" when "0111",
               "1000" when "1000",
               "1001" when "1001",
               "0000" when others;

end behavioral;
```



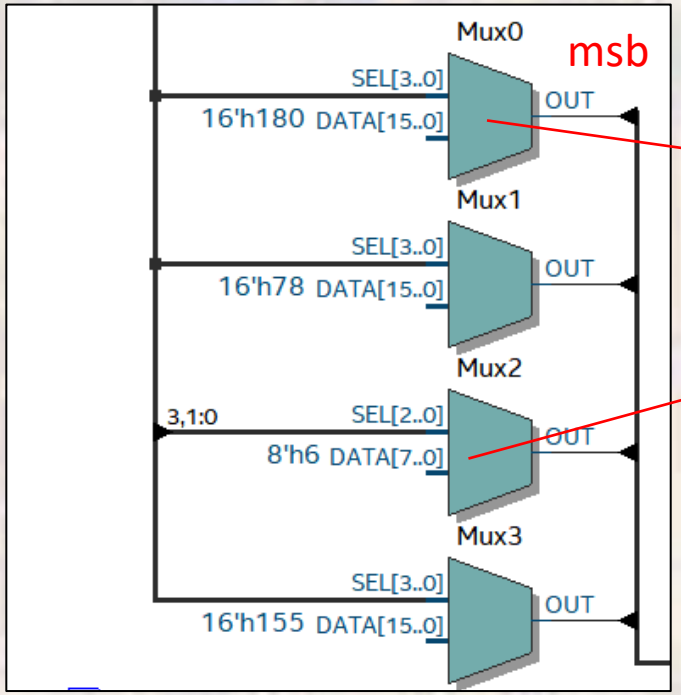
HDL FSM Intro

- Mod 10 Counter



HDL FSM Intro

- Mod 10 Counter
  - 4 16bit values instead of 16 4bit values



16'h180
0x0180
0000 0001 1000 0000

0000 0001 1000 0000
0000 0000 0111 1000
0000 0000 0110 0110
0000 0001 0101 0101

0001 0100 0000
SEL (State)

"0001" when "0000",
"0010" when "0001",

HDL FSM Intro

- Mod 10 Counter – test bench

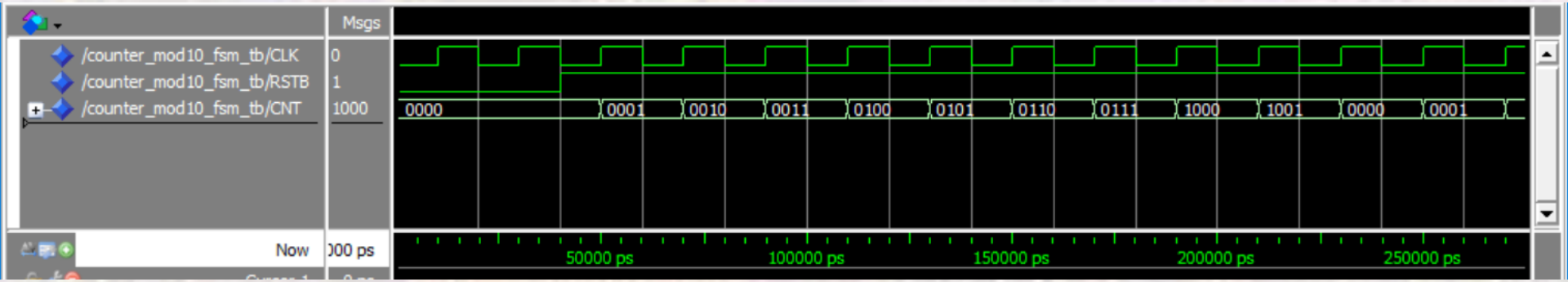
```
--------------------------------
--
-- counter_mod10_fsm_tb.vhdl
--
-- created: 3/30/18
-- by: johnsontimoj
-- rev: 0
--
-- testbench for mod 10 FSM counter
-- of counter_mod10_fsm_tb.vhdl
--
--------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity counter_mod10_fsm_tb is
   -- no port entry - testbench
end entity;

architecture testbench of counter_mod10_fsm_tb is
   signal   CLK:  std_logic;
   signal   RSTB: std_logic;

   signal   CNT:  std_logic_vector(3 downto 0);

   constant PER:  time:=   20 ns;
```

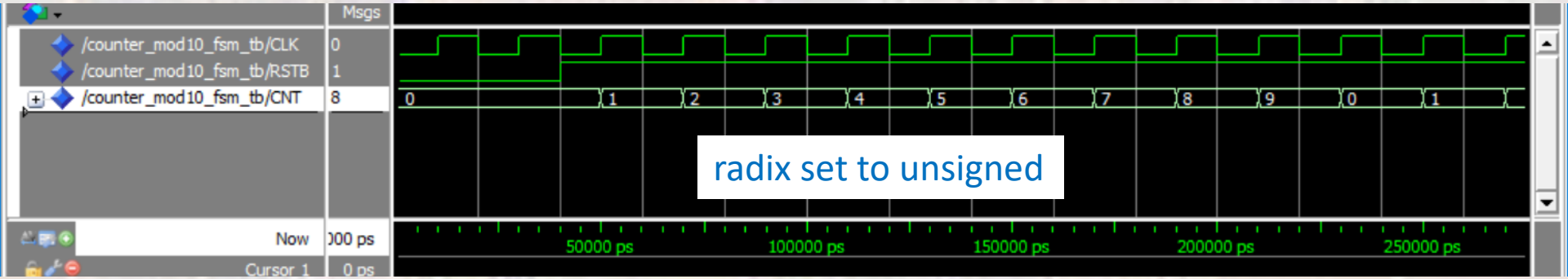
```
   ---------------------------------
   -- Component prototype
   ---------------------------------
   COMPONENT counter_mod10_fsm
      PORT
      (
         i_rstb      :    IN STD_LOGIC;
         i_clk    :    IN STD_LOGIC;
         o_cnt :    OUT STD_LOGIC_VECTOR(3 downto 0)
      );
   END COMPONENT;
   ------------------------------------

   begin

   ---------------------------------------
   -- Device under test (DUT)
   ---------------------------------------
   DUT: counter_mod10_fsm
      port map(
               i_clk => CLK,
               i_rstb => RSTB,
               o_cnt => CNT
      );
```

```
   ---------------------------------------
   -- Test processes
   ---------------------------------------

   -- Clock process
   clock: process    -- note - no sensitivity list
   begin
      CLK <= '0';
      wait for PER/2;
      infinite: loop
         CLK <= not CLK; wait for PER/2;
      end loop;
   end process;

   -- Reset process
   reset: process    -- note - no sensitivity list
   begin
      RSTB <= '0'; wait for 2*PER;
      RSTB <= '1'; wait;
   end process reset;

   -- Run Process
   -- no run process needed - run sim to 250 ns

   ------------------------------------------
   -- End test processes
   ------------------------------------------

end architecture;
```

HDL FSM Intro

- Mod 10 Counter





HDL FSM Intro

Mod 10 Counter using a case statement

HDL FSM Intro

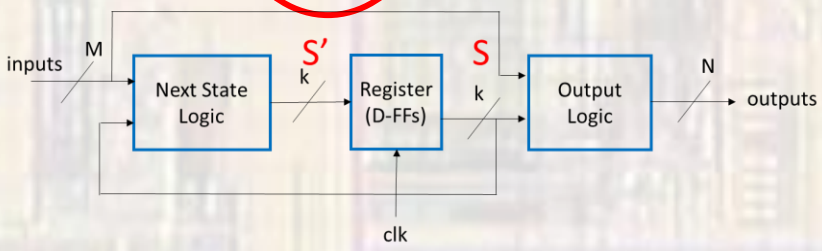
- Mod 10 Counter – case statement

Made these unsigned since they represent a count

```
--------------------------------------
--
-- counter_mod10_fsmA.vhdl
--
-- created 3/30/18
-- tj
--
-- rev 0
--------------------------------------
--
-- mod 10 up-counter - state machine with case statement
--------------------------------------
--
-- Inputs: i_rstb, i_clk
-- Outputs: o_cnt[3:0]
--
--------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_mod10_fsmA is
   port (
         i_clk :     in std_logic;
         i_rstb : in std_logic;

         o_cnt :  out std_logic_vector(3 downto 0)
    );
end entity;
```

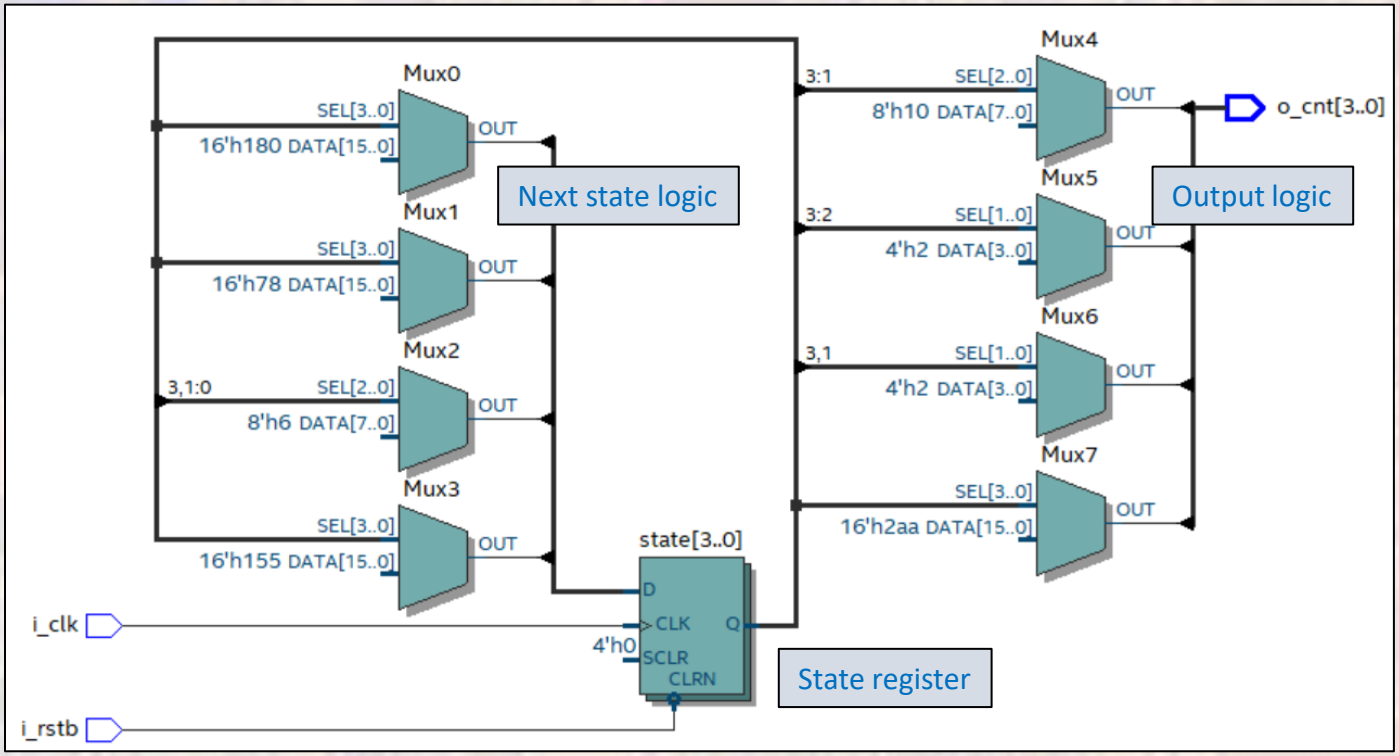
```
architecture behavioral of counter_mod10_fsmA is

   --
   -- internal signals
   --
   signal state:        unsigned(3 downto 0);
   signal state_next:   unsigned(3 downto 0);

begin

   --
   -- next state logic
   --
   process(all)
   begin
      case state is
         when "0000" => state_next <= "0001";
         when "0001" => state_next <= "0010";
         when "0010" => state_next <= "0011";
         when "0011" => state_next <= "0100";
         when "0100" => state_next <= "0101";
         when "0101" => state_next <= "0110";
         when "0110" => state_next <= "0111";
         when "0111" => state_next <= "1000";
         when "1000" => state_next <= "1001";
         when "1001" => state_next <= "0000";
         when others => state_next <= "0000";
      end case;
   end process;
```

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

   --
   -- Output logic
   --
   with state select
      o_cnt <=    "0000" when "0000",
                  "0001" when "0001",
                  "0010" when "0010",
                  "0011" when "0011",
                  "0100" when "0100",
                  "0101" when "0101",
                  "0110" when "0110",
                  "0111" when "0111",
                  "1000" when "1000",
                  "1001" when "1001",
                  "0000" when others;

end behavioral;
```



HDL FSM Intro

- Mod 10 Counter –with case statement



HDL FSM Intro

Mod 10 Counter using a case statement X 2

HDL FSM Intro

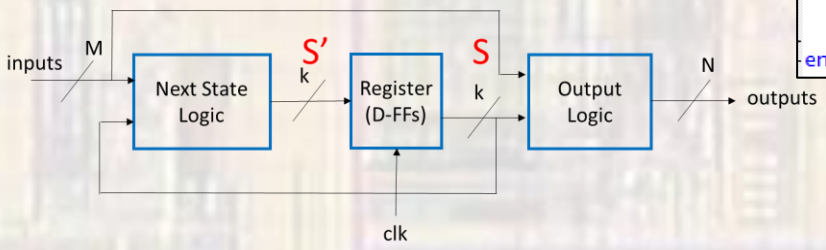
- Mod 10 Counter – case statement X 2

Made these unsigned since they represent a count

```
--------------------------------------
--
-- counter_mod10_fsmB.vhdl
--
-- created 3/30/18
-- tj
--
-- rev 0
----------------------------------------
--
-- mod 10 up-counter - state machine with case statement
--                     case statement for output logic
----------------------------------------
--
-- Inputs: i_rstb, i_clk
-- Outputs: o_cnt[3:0]
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_mod10_fsmB is
   port (
         i_clk :     in std_logic;
         i_rstb : in std_logic;

         o_cnt :  out std_logic_vector(3 downto 0)
    );
end entity;
```

```
architecture behavioral of counter_mod10_fsmB is

   --
   -- internal signals
   --
   signal state:        unsigned(3 downto 0);
   signal state_next:   unsigned(3 downto 0);

begin

   --
   -- next state logic
   --
   process(all)
   begin
      case state is
         when "0000" => state_next <= "0001";
         when "0001" => state_next <= "0010";
         when "0010" => state_next <= "0011";
         when "0011" => state_next <= "0100";
         when "0100" => state_next <= "0101";
         when "0101" => state_next <= "0110";
         when "0110" => state_next <= "0111";
         when "0111" => state_next <= "1000";
         when "1000" => state_next <= "1001";
         when "1001" => state_next <= "0000";
         when others => state_next <= "0000";
      end case;
   end process;
```

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

   --
   -- Output logic
   --
   process(all)
   begin
      case state is
         when "0000" => o_cnt <= "0000";
         when "0001" => o_cnt <= "0001";
         when "0010" => o_cnt <= "0010";
         when "0011" => o_cnt <= "0011";
         when "0100" => o_cnt <= "0100";
         when "0101" => o_cnt <= "0101";
         when "0110" => o_cnt <= "0110";
         when "0111" => o_cnt <= "0111";
         when "1000" => o_cnt <= "1000";
         when "1001" => o_cnt <= "1001";
         when others => o_cnt <= "0000";
      end case;
   end process;


end behavioral;
```



HDL FSM Intro

- Mod 10 Counter –with case statement X 2