Enumerate States

Last updated 1/22/21

Enumerated States

- These slides introduce Enumerated States
- Upon completion: You should be able to create a state machine using enumerated states

Enumerated States

- Enumerated States
  - Similar to enumerated types in C
  - Define a new TYPE with a fixed set of recognized values

```
type STATE_TYPE is (A, B, C, D, E, F, G, H, I, J);
```

  - Each state can be identified by its position (integer)
    - A – 0
    - B – 1

      …
    - J – 9

    - STATE_TYPE’pos(D) → 3

Enumerated States

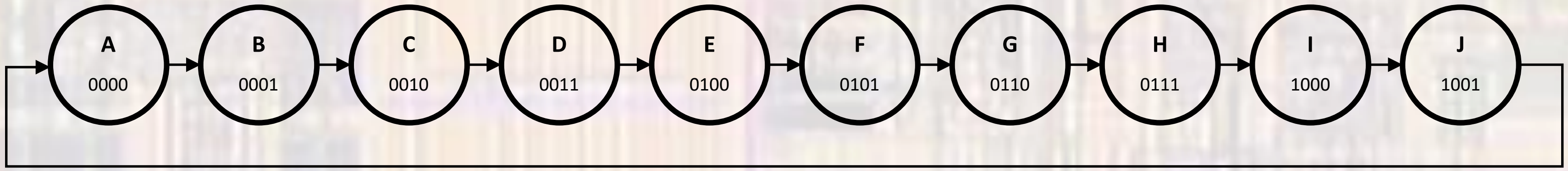
- Enumerated States
  - Create signals of the enumerated type

```
signal state:        STATE_TYPE;
signal state_next:   STATE_TYPE;
```

  - Synthesis is done using the enumerated states with the corresponding encoding
    - Binary
    - One Hot
    - Gray
    - Johnson
    - Auto

Enumerated States

Mod 10 Counter using Enumerated States

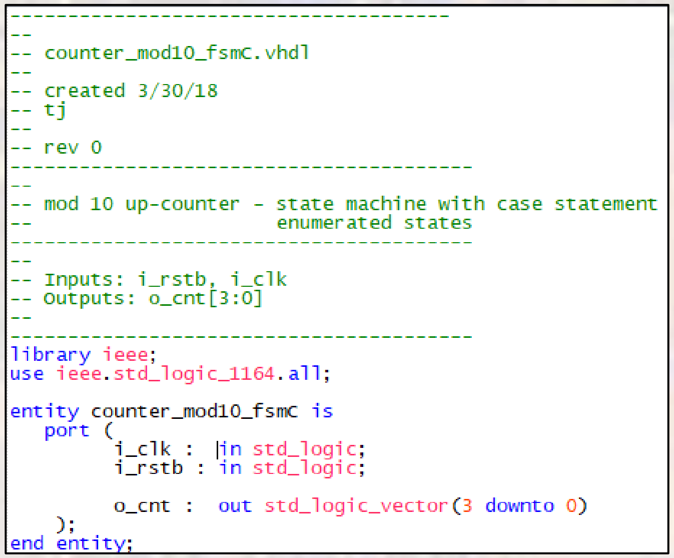


Enumerated States

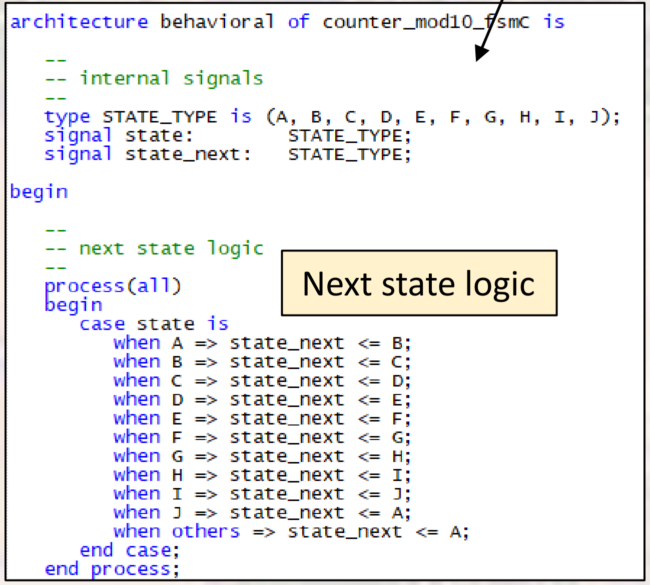
- Mod 10 Counter – Enumerated States

Enumerated state declaration

State initialization and update (register logic)

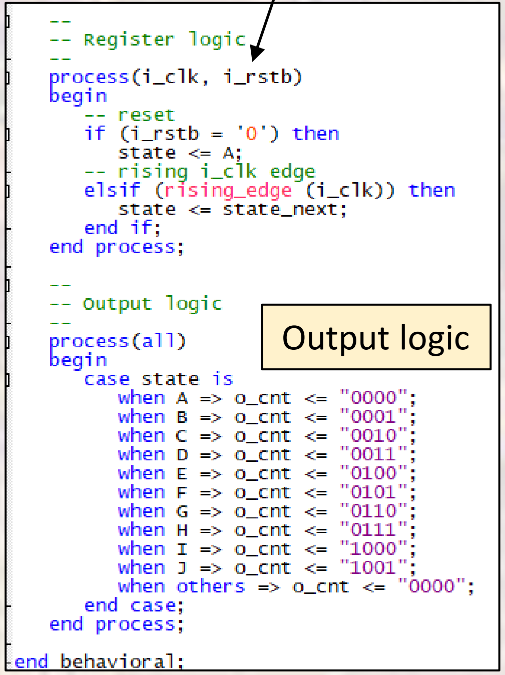


```
----------------------------------------
--
-- counter_mod10_fsmC.vhdl
--
-- created 3/30/18
-- tj
--
-- rev 0
----------------------------------------
--
-- mod 10 up-counter - state machine with case statement
--                     enumerated states
----------------------------------------
--
-- Inputs: i_rstb, i_clk
-- Outputs: o_cnt[3:0]
--
----------------------------------------
library ieee;
use ieee.std_logic_1164.all;

entity counter_mod10_fsmC 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_fsmC is

   --
   -- internal signals
   --
   type STATE_TYPE is (A, B, C, D, E, F, G, H, I, J);
   signal state:        STATE_TYPE;
   signal state_next:   STATE_TYPE;

begin

   --
   -- next state logic
   --
   process(all)
   begin
      case state is
         when A => state_next <= B;
         when B => state_next <= C;
         when C => state_next <= D;
         when D => state_next <= E;
         when E => state_next <= F;
         when F => state_next <= G;
         when G => state_next <= H;
         when H => state_next <= I;
         when I => state_next <= J;
         when J => state_next <= A;
         when others => state_next <= A;
      end case;
   end process;
```

Next state logic

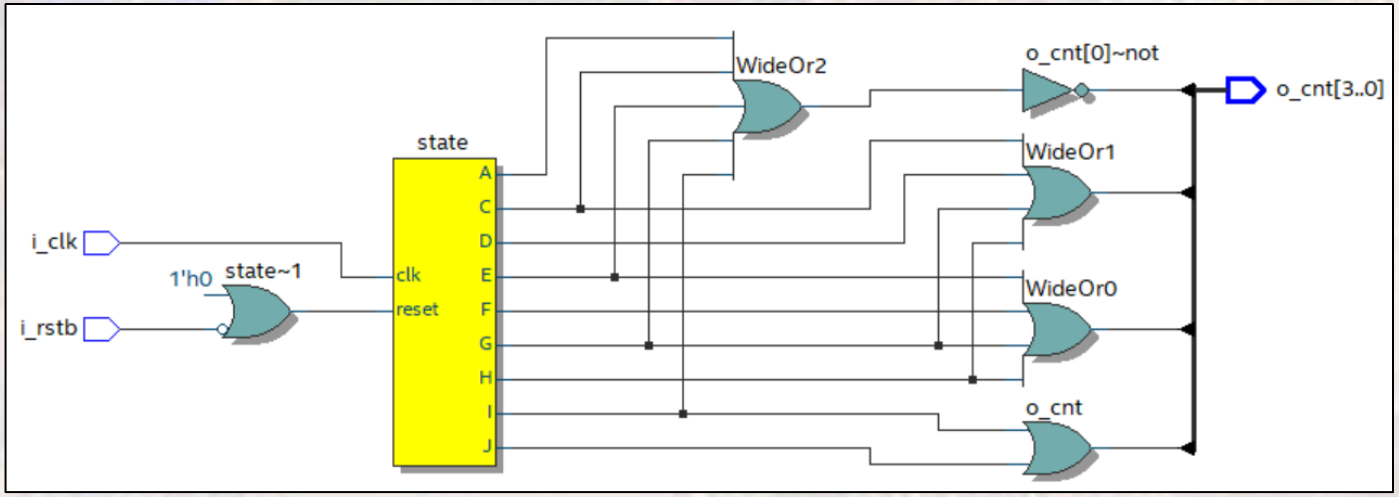


```
   --
   -- Register logic
   --
   process(i_clk, i_rstb)
   begin
      -- reset
      if (i_rstb = '0') then
         state <= A;
      -- rising i_clk edge
      elsif (rising_edge (i_clk)) then
         state <= state_next;
      end if;
   end process;

   --
   -- Output logic
   --
   process(all)
   begin
      case state is
         when A => o_cnt <= "0000";
         when B => o_cnt <= "0001";
         when C => o_cnt <= "0010";
         when D => o_cnt <= "0011";
         when E => o_cnt <= "0100";
         when F => o_cnt <= "0101";
         when G => o_cnt <= "0110";
         when H => o_cnt <= "0111";
         when I => o_cnt <= "1000";
         when J => o_cnt <= "1001";
         when others => o_cnt <= "0000";
      end case;
   end process;

end behavioral;
```

Output logic

Enumerated States

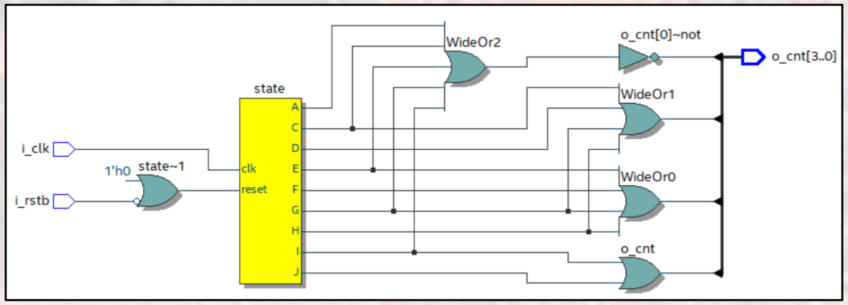
- Mod 10 Counter –Enumerated States

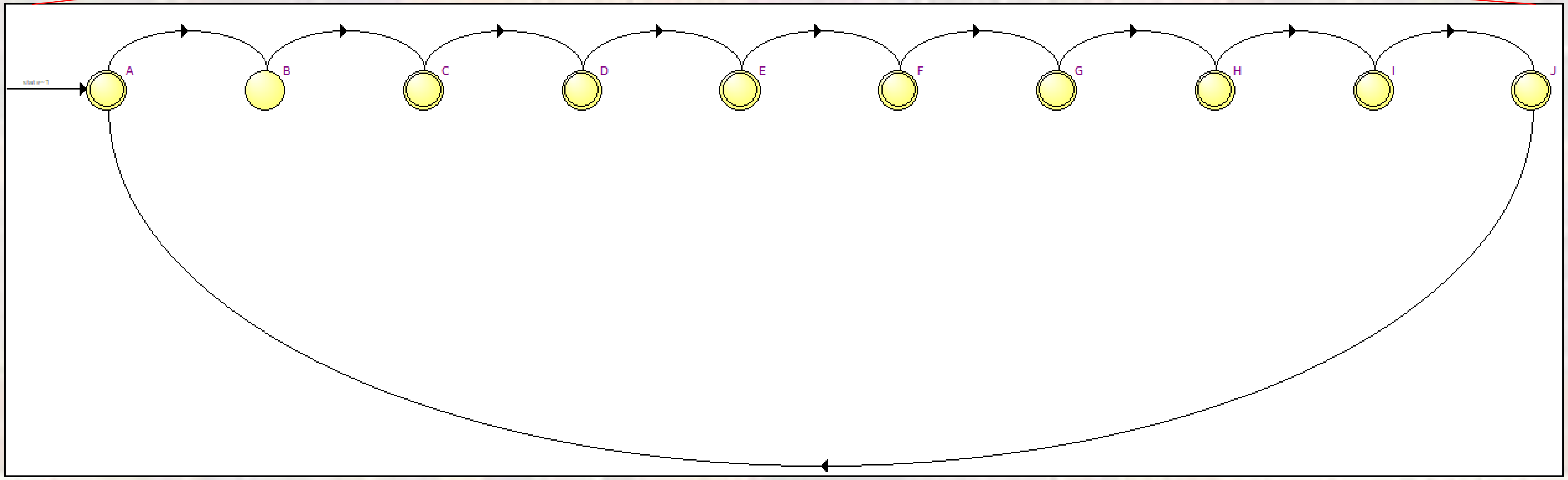


Quartus recognized part of the code as a state machine

Enumerated States

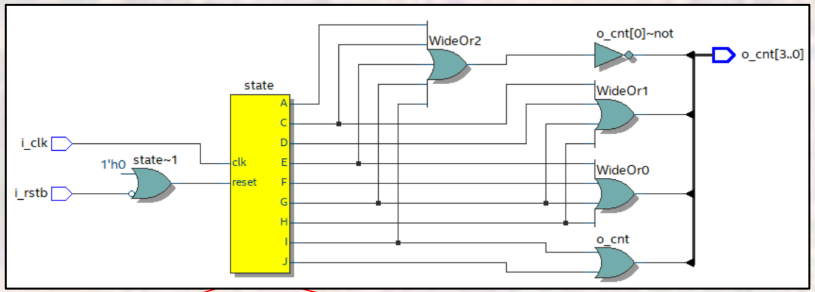
- Mod 10 Counter –Enumerated States

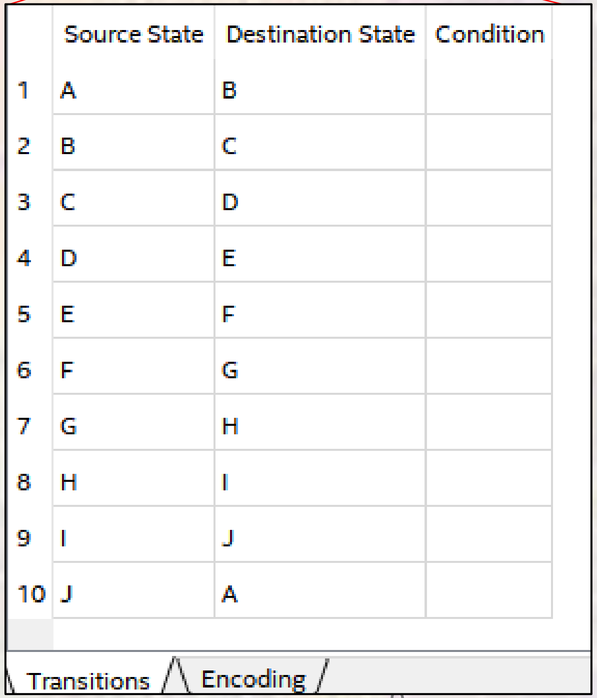




Enumerated States

- Mod 10 Counter –Enumerated States



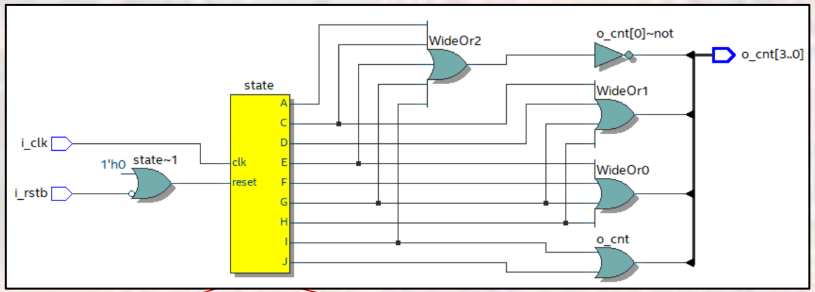


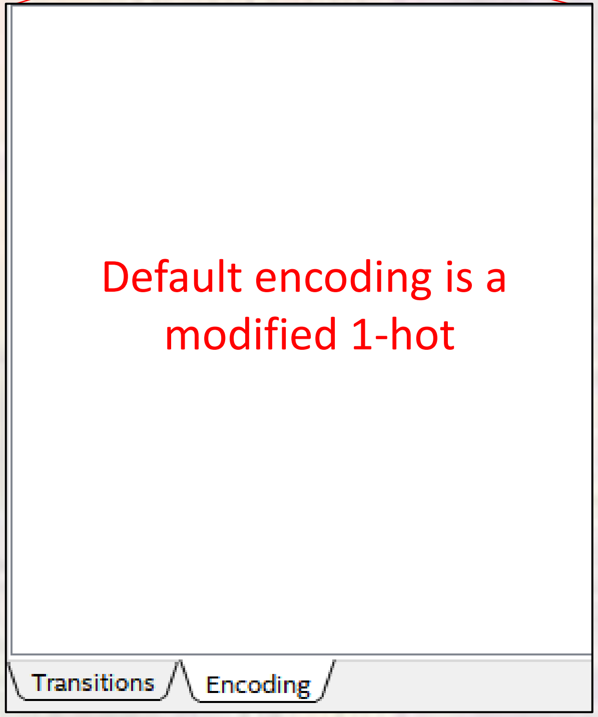
	Source State	Destination State	Condition
1	A	B	
2	B	C	
3	C	D	
4	D	E	
5	E	F	
6	F	G	
7	G	H	
8	H	I	
9	I	J	
10	J	A	

Transitions
Encoding

Enumerated States

- Mod 10 Counter –Enumerated States

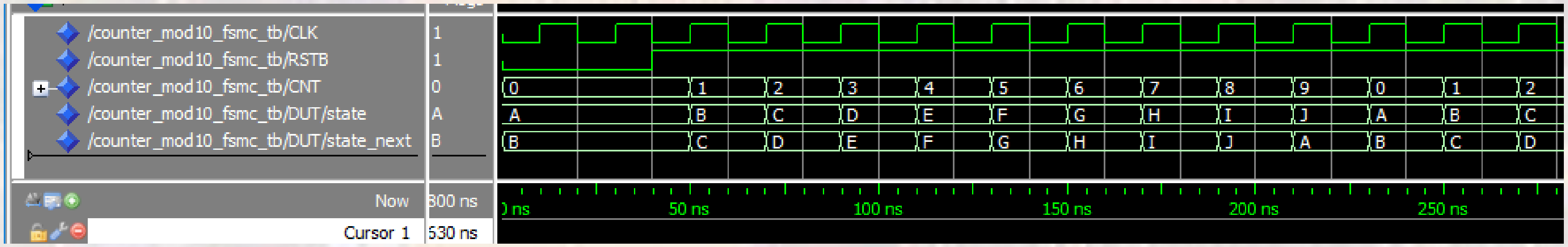




Default encoding is a modified 1-hot

Enumerated States

- Mod 10 Counter –Enumerated States
  - Add the states and re-run the simulation


/counter_mod10_fsmc_tb/CLK
/counter_mod10_fsmc_tb/RSTB
/counter_mod10_fsmc_tb/CNT
/counter_mod10_fsmc_tb/DUT/state
/counter_mod10_fsmc_tb/DUT/state_next
Now
Cursor 1
300 ns
630 ns
0 ns
50 ns
100 ns
150 ns
200 ns
250 ns

```
/counter_mod10_fsmc_tb/CLK             1
/counter_mod10_fsmc_tb/RSTB            1
/counter_mod10_fsmc_tb/CNT             0   0 1 2 3 4 5 6 7 8 9 0 1 2
/counter_mod10_fsmc_tb/DUT/state       A   A B C D E F G H I J A B C
/counter_mod10_fsmc_tb/DUT/state_next  B   B C D E F G H I J A B C D
```