**FSM for detecting 1010 overlapping sequence**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity seq1010 is

 Port ( x,clk : in STD\_LOGIC;

 z : out STD\_LOGIC);

end seq1010;

architecture Behavioral of seq1010 is

Type state is (A, B, C, D);

signal y: state;

begin

 Process(clk)

 begin

 If(clk'event and clk = '1') then

 case y is

 when A =>

 If(x = '0') then y <= A; z <= '0';

 Else y <= B; z <= '0';

 End If;

 when B =>

 If(x = '0') then y <= C; z <= '0';

 Else y <= B; z <= '0';

 End If;

 when C =>

 If(x = '0') then y <= A; z <= '0';

 Else y <= D; z <= '0';

 End If;

 when D =>

 If(x = '0') then y <= C; z <= '1';

 Else y <= B; z <= '0';

 End If;

 End Case;

 End If;

 End Process;

end Behavioral;

**FSM for detecting 1010 overlapping sequence**



**JK FLIP-FLOP WITH ASYNCHRONOUS CLEAR & SYNCHRONOUS PRESET**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

entity jkff is

 Port ( j : in STD\_LOGIC;

 k : in STD\_LOGIC;

 clk : in STD\_LOGIC;

 q : out STD\_LOGIC;

 qbar : out STD\_LOGIC;

 pr : in STD\_LOGIC;

 clr : in STD\_LOGIC);

end jkff;

architecture Behavioral of jkff is

signal temp: std\_logic;

begin

Process (clr, clk, pr)

Begin

If(clr = '0') then temp<='0';

ElsIf(clk'event AND clk = '0') then

 If(pr = '0') then temp<= '1';

 Else

 If(j = '0' AND k = '1') then temp <= '0';

 ElsIf(j = '1' AND k = '0') then temp <= '1';

 ElsIf(j = '1' AND k = '1') then temp <= not temp;

 End If;

 End If;

End If;

End Process;

 q <= temp;

 qbar <= not temp;

end Behavioral;

**JK FLIP-FLOP WITH ASYNCHRONOUS CLEAR & SYNCHRONOUS PRESET**



**3:8 DECODER USING WHEN-ELSE**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity decoder is

 Port ( data : in STD\_LOGIC\_VECTOR (02 downto 0);

 output : out STD\_LOGIC\_VECTOR (07 downto 0));

end decoder;

architecture Dataflow of decoder is

begin

output <=

 "11111110" when data = "000" else

 "11111101" when data = "001" else

 "11111011" when data = "010" else

 "11110111" when data = "011" else

 "11101111" when data = "100" else

 "11011111" when data = "101" else

 "10111111" when data = "110" else

 "01111111" when data = "111";

end Dataflow;

**3:8 DECODER USING WHEN-ELSE**



**16 Bit Barrel Shifter**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

use IEEE.STD\_LOGIC\_ARITH.ALL;

use IEEE.STD\_LOGIC\_UNSIGNED.ALL;

entity barrel\_shift is

port( DIN : in std\_logic\_vector(15 downto 0);

 SHIFT\_AMT : in std\_logic\_vector(3 downto 0);

 CLK, ENB : in std\_logic;

 DOUT : out std\_logic\_vector (15 downto 0));

end barrel\_shift;

architecture Behavioral of barrel\_shift is

signal DIN\_BIT, DOUT\_BIT : bit\_vector (15 downto 0);

signal S\_INT : integer;

begin

 DIN\_BIT <= to\_bitvector(DIN);

 S\_INT <= CONV\_INTEGER(SHIFT\_AMT);

 process(CLK)

 begin

 if(CLK' event and CLK='1') then

 if(ENB = '0') then

 DOUT <= DIN;

 else

 DOUT\_BIT <= DIN\_BIT ror S\_INT;

 DOUT <= to\_stdlogicvector(DOUT\_BIT);

 end if;

 end if;

 end process;

end Behavioral;

