8.3

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  entity jkff is  port(  clk: in std\_logic;  q: out std\_logic;  jk: in std\_logic\_vector(1 downto 0)  );  end jkff;  architecture jk\_arch of jkff is  signal q\_reg: std\_logic;  signal q\_next: std\_logic;  begin  process(clk)  begin  if(clk'event and clk='1') then  q\_reg <= q\_next;  end if;  end process;    -- next state logic  q\_next <= q\_reg when jk="00" else  '0' when jk="01" else  '1' when jk="10" else  not q\_reg;    -- output logic  q <= q\_reg;    end jk\_arch; |

Negative T-flipflop

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  entity neg\_tff is  port(  clk: in std\_logic;  q: out std\_logic;  t: in std\_logic;  reset: in std\_logic  );  end neg\_tff;  architecture negt\_arch of neg\_tff is  signal q\_reg: std\_logic;  signal q\_next: std\_logic;  begin  process(clk)  begin  if(clk'event and clk='0') then  if(reset = '1') then  q\_reg <= '0';  else  q\_reg <= q\_next;  end if;  end if;  end process;    -- next state logic  q\_next <= q\_reg when t='0' else  not(q\_reg);    -- output logic  q <= q\_reg;    end negt\_arch; |

8.5

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  entity univ\_shift is  port(  clk, reset: in std\_logic;  ctrl: in std\_logic\_vector(2 downto 0);  q: out std\_logic\_vector(3 downto 0);  d: in std\_logic\_vector(3 downto 0)  );  end univ\_shift;  architecture univ\_arch of univ\_shift is  signal r\_reg: std\_logic\_vector(3 downto 0);  signal r\_next: std\_logic\_vector(3 downto 0);  begin  process(clk, reset)  begin  if(reset = '1') then  r\_reg <= (others => '0');  elsif(clk'event and clk='1') then  r\_reg <= r\_next;  end if;  end process;    -- next state logic  with ctrl select  r\_next <= r\_reg when "000", -- pause  r\_reg(2 downto 0) & d(0) when "001", -- shift left  d(3) & r\_reg(3 downto 1) when "010", -- shift right  r\_reg(2 downto 0) & r\_reg(3) when "011", -- rotate left  r\_reg(0) & r\_reg(3 downto 1) when "100", -- rotate right  d when others;  q <= r\_reg;  end univ\_arch; |

8.7

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.numeric\_std.all;  entity threetwelve is  port(  clk: in std\_logic;  q: out std\_logic\_vector(3 downto 0)  );  end threetwelve;  architecture tt\_arch of threetwelve is  signal r\_reg: unsigned(3 downto 0);  signal r\_next: unsigned(3 downto 0);  begin  process(clk)  begin  if(clk'event and clk='1') then  r\_reg <= r\_next;  end if;  end process;    -- next state logic  r\_next <= r\_reg + 1 when(r\_reg >= "0011" and r\_reg < "1100") else  "0011";    -- output logic  q <= std\_logic\_vector(r\_reg);    end tt\_arch; |

8.8

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.numeric\_std.all;  entity arbitrary\_mod is  port(  clk, reset: in std\_logic;  q: out std\_logic\_vector(2 downto 0)  );  end arbitrary\_mod;  architecture mod\_arch of arbitrary\_mod is  signal r\_reg: unsigned(2 downto 0);  signal counter, counter\_next: unsigned(2 downto 0);  begin  process(clk,reset)  begin  if(reset='1') then  counter <= (others=>'0');  elsif(clk'event and clk='1') then  counter <= counter\_next;  end if;  end process;    -- next state logic  counter\_next <= (others=>'0') when counter=4 else  counter+1;    r\_reg <= "000" when counter\_next=4 else  "011" when counter\_next=0 else  "110" when counter\_next=1 else  "101" when counter\_next=2 else  "111";    -- output logic  q <= std\_logic\_vector(r\_reg);    end mod\_arch; |

8.10

|  |
| --- |
| library ieee;  use ieee.std\_logic\_1164.all;  use ieee.numeric\_std.all;  entity mil\_count is  port(  clk: in std\_logic;  q: out std\_logic  );  end mil\_count;  architecture mil\_arch of mil\_count is  signal r\_reg: unsigned(18 downto 0) := to\_unsigned(0,19);  signal r\_next, r\_inc: unsigned(18 downto 0) := to\_unsigned(0,19);  signal zero, zero\_next: std\_logic := '0';  begin  process(clk)  begin  if(clk'event and clk='1') then  r\_reg <= r\_next;  end if;  end process;    r\_inc <= r\_reg + 1;  r\_next <= (others=>'0') when r\_inc = 500000 else -- this is mod 500000 counter !!  r\_inc;    process(clk)  begin  if(clk'event and clk='1') then  zero <= zero\_next;  end if;  end process;    zero\_next <= not zero when r\_inc = 500000 else  zero;  -- output logic  q <= zero;    end mil\_arch; |