ECE477 Homework
Due: 19-12-2019

Abdulkerim Güven
201711033

3 bit up counter with D flip-flops (use DFF as component)

Code for D Flip-Flop:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity d_ff is
   port(clk, d: in std_logic;
            q: out std_logic);
end d_ff;

architecture Behavioral of d_ff is
begin
   process(clk)
   begin
            if(res='1') then
                        q <= '0';
             elsif(rising_edge(clk)) then
                        q <= d;
             end if;
   end process;
end Behavioral;
```

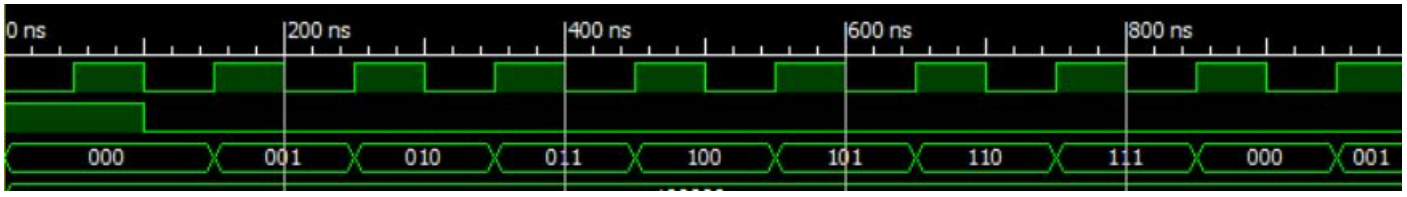
Code for the 3-bit Up Counter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity up_counter is
   port(clk, reset: in std_logic;
            count: out std_logic_vector(2 downto 0));
end up_counter;

architecture Behavioral of up_counter is

component d_ff is
   port(clk, d, res: in std_logic;
            q: out std_logic);
end component;

signal d1_out: std_logic := '1'; -- least significant bit
signal d2_out: std_logic := '0';
signal d3_out: std_logic := '0'; -- most significant bit
signal d1_in: std_logic := '0';
signal d2_in: std_logic := '0';
signal d3_in: std_logic := '0';

begin
   count(2) <= d3_out;
   count(1) <= d2_out;
   count(0) <= d1_out;
   d1: d_ff port map
   (
   clk => clk,
   d => d1_in,
   q => d1_out,
   res => reset
   );

   d2: d_ff port map
   (
   clk => clk,
   d => d2_in,
   q => d2_out,
   res => reset
   );

   d3: d_ff port map
   (
   clk => clk,
   d => d3_in,
   q => d3_out,
   res => reset
   );

   d1_in <= not d1_out;
   d2_in <= (d1_out xor d2_out);
   d3_in <= (d1_out and d2_out) xor d3_out;

end Behavioral;
```

Test Bench Code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY uc_TB IS
END uc_TB;

ARCHITECTURE behavior OF uc_TB IS

        -- Component Declaration for the Unit
Under Test (UUT)

        COMPONENT up_counter
        PORT(
        clk : IN  std_logic;
        reset : IN  std_logic;
        count : OUT  std_logic_vector(2 downto 0)
        );
        END COMPONENT;


  --Inputs
  signal clk : std_logic := '0';
  signal reset : std_logic := '1';

         --Outputs
  signal count : std_logic_vector(2 downto 0);

  -- Clock period definitions
  constant clk_period : time := 100 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
  uut: up_counter PORT MAP (
          clk => clk,
          reset => reset,
          count => count
          );

  -- Clock process definitions
  clk_process :process
  begin
          clk <= '0';
          wait for clk_period/2;
          clk <= '1';
          wait for clk_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
          -- hold reset state for 100 ns.
          wait for 100 ns;
           reset <= '0';
          wait for clk_period*10;
          -- insert stimulus here

          wait;
  end process;

END;
```

Simulation Results:



Frequency divider with D flip-flops (Divide 100 MHz to 25 MHz)

Code for the D Flip-Flop is the same as above.

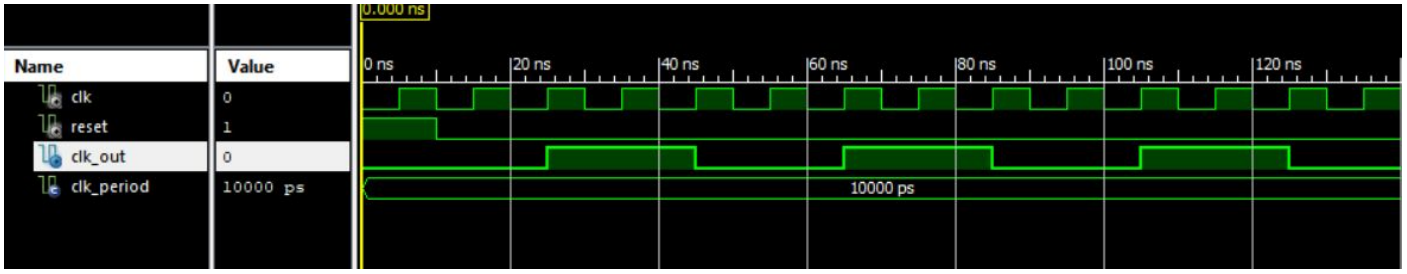
Code for the Frequency Divider:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity freq_divider is
   port(clk, reset: in std_logic;
            clk_out: out std_logic);
end freq_divider;

architecture Behavioral of freq_divider is

signal d1_in : std_logic := '1';
signal d2_in : std_logic := '0';
signal d1_out : std_logic := '0';
signal d2_out : std_logic := '0';

component d_ff is
   port(clk, d, res: in std_logic;
            q: out std_logic);
end component;

begin
    d1: d_ff port map
            (
            clk => clk,
            d => d1_in,
            q => d1_out,
            res => reset
            );
    d2: d_ff port map
            (
            clk => clk,
            d => d2_in,
            q => d2_out,
            res => reset
            );

    d1_in <= not d1_out;
    d2_in <= (d2_out xor d1_out);

    clk_out <= '1' when (d2_out='1') else '0';

end Behavioral;
```

Test Bench Code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY freq_divider_TB IS
END freq_divider_TB;

ARCHITECTURE behavior OF freq_divider_TB IS
        COMPONENT freq_divider
        PORT(
        clk : IN  std_logic;
        reset : IN  std_logic;
        clk_out : OUT  std_logic
        );
        END COMPONENT;

  --Inputs
  signal clk : std_logic := '0';
  signal reset : std_logic := '1';

         --Outputs
  signal clk_out : std_logic;

  -- Clock period definitions
  constant clk_period : time := 10 ns;

BEGIN
   -- Instantiate the Unit Under Test (UUT)
  uut: freq_divider PORT MAP (
        clk => clk,
        reset => reset,
        clk_out => clk_out
        );
  clk_process :process
  begin
         clk <= '0';
         wait for clk_period/2;
         clk <= '1';
         wait for clk_period/2;
  end process;

  stim_proc: process
  begin
        wait for 10 ns;
         reset <= '0';
        wait;
  end process;
END;
```

Simulation Results:



3 bit up counter with T flip-flop(use TFF as component)

Code for T Flip-Flop:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity t_ff is
   port(t, clk: in std_logic;
            q: out std_logic);
end t_ff;

architecture Behavioral of t_ff is
signal temp: std_logic := '0';
begin
   process (clk)
   begin
            if (rising_edge(clk)) then
                     if (t='0') then
                              temp <= temp;
                     elsif (t='1') then
                              temp <= not (temp);
                     end if;
            end if;
   end process;
q <= temp;
end Behavioral;
```

Code for Up Counter:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity three_bit_up_counter_tff is
   port(clk: in std_logic;
            count: out std_logic_vector(2 downto 0));
end three_bit_up_counter_tff;

architecture Behavioral of three_bit_up_counter_tff
is

component t_ff is
   port(clk, t: in std_logic;
            q: out std_logic);
end component;

signal t1_out: std_logic := '1'; -- least significant bit
signal t2_out: std_logic := '0';
signal t3_out: std_logic := '0'; -- most significant bit
signal t1_in: std_logic := '0';
signal t2_in: std_logic := '0';
signal t3_in: std_logic := '0';

begin
   count(2) <= t3_out;
   count(1) <= t2_out;
   count(0) <= t1_out;
   t1: t_ff port map
   (
   clk => clk,
   t => t1_in,
   q => t1_out
   );

   t2: t_ff port map
   (
   clk => clk,
   t => t2_in,
   q => t2_out
   );

   t3: t_ff port map
   (
   clk => clk,
   t => t3_in,
   q => t3_out
   );

   t1_in <= '1';
   t2_in <= t1_out;
   t3_in <= (t2_out and t1_out);

end Behavioral;
```

Test Bench Code:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY three_bit_up_tff_TB IS
END three_bit_up_tff_TB;

ARCHITECTURE behavior OF three_bit_up_tff_TB IS

        -- Component Declaration for the Unit
Under Test (UUT)

        COMPONENT three_bit_up_counter_tff
        PORT(
        clk : IN  std_logic;
        count : OUT  std_logic_vector(2 downto 0)
        );
        END COMPONENT;


  --Inputs
  signal clk : std_logic := '0';

         --Outputs
  signal count : std_logic_vector(2 downto 0);

  -- Clock period definitions
  constant clk_period : time := 100 ns;

BEGIN

  -- Instantiate the Unit Under Test (UUT)
  uut: three_bit_up_counter_tff PORT MAP (
          clk => clk,
          count => count
          );

  -- Clock process definitions
  clk_process :process
  begin
          clk <= '0';
          wait for clk_period/2;
          clk <= '1';
          wait for clk_period/2;
  end process;


  -- Stimulus process
  stim_proc: process
  begin
          -- hold reset state for 100 ns.
          wait for 100 ns;

          wait for clk_period*10;

          -- insert stimulus here

          wait;
  end process;

END;
```

Simulation Results: