

EE2000 2018/19 Quiz 3 - 90-mins – Total marks: 60 marks

Question 1: (ROM + VHDL) – 15 marks

- a) What is the size of the smallest ROM that is needed to implement “A 16-bit adder”? Assume no enable signal is needed, specify the answer as number of entries X data width. (Hint: add two 16-bit numbers to get a 17-bit sum)

Suggested Answer: $2^{32} \times 17$

- b) Write the VHDL module “Q1” for the above 16-bit adder, input A, B, and output S. Assume no extra carry-in, carry-out signals.

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

entity Q1 is
    port(
        A, B: in std_logic_vector(15 downto 0);
        S: out std_logic_vector(16 downto 0)
    );
end Q1;

architecture Behavioral of Q1 is
begin
    S <= std_logic_vector(unsigned('0'&A) + unsigned('0'&B));
end Behavioral;
```

- c) Write the VHDL simulation module “Q1_tb” for the above 16-bit adder.

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

entity Q1_tb is
end Q1_tb;

architecture Behavioral of Q1_tb is
    component Q1 is
        port(
            A, B: in std_logic_vector(15 downto 0);
            S: out std_logic_vector(16 downto 0)
        );
    end component;
    signal A, B: std_logic_vector(15 downto 0);
    signal S: std_logic_vector(16 downto 0);
begin
    uut: Q1 port map (A=>A, B=>B, S=>S);

    stimuli: process
    begin
        for i in 0 to 65535 loop
            for j in 0 to 65535 loop
                A <= std_logic_vector(to_unsigned(j,16));
                B <= std_logic_vector(to_unsigned(i,16));
                wait for 10ns;
            end loop;
        end loop;
        wait;
    end process;
end Behavioral;
```

Question 2: (ROM + VHDL) 10 marks

- a) Given $F = A'B' + BC'$ and $G = AC + B'$, write a complete VHDL module "Q2" that realizes the two functions F and G, using an 8-word x 2-bit ROM (8 rows, each row has two bits). Please use array type declaration and constant declaration that defines the contents of the ROM.

```
library IEEE;
use IEEE.numeric_bit.all;

entity FG is
    port(A, B, C: in bit; F, G: out bit);
end FG;

architecture ROM of FG is
    type ROM8_2 is array (0 to 7) of unsigned(0 to 1);
    constant ROM1: ROM8_2 := ("11", "11", "10", "00", "01", "01", "10", "01");
    signal romout: unsigned(0 to 1);
    signal index: unsigned(0 to 2);
begin
    index <= A & B & C;
    Romout <= ROM1(to_integer(index));
    F <= Romout(0);
    G <= Romout(1);
end ROM;
```

- b) Please provide the testbench for the above VHDL module "Q2_tb".

```
library IEEE;
use IEEE.NUMERIC_BIT.ALL;

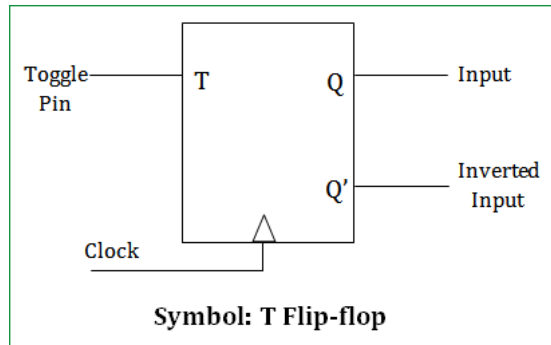
entity Q2_tb is
end Q2_tb;

architecture Behavioral of Q2_tb is
    component FG is
        port (
            A, B, C: in bit;
            F, G: out bit
        );
    end component;
    signal A, B, C, F, G: bit;
    signal tmp: bit_vector(2 downto 0);
begin
    uut: FG port map (A=>A, B=>B, C=>C, F=>F, G=>G);

    stimuli: process
    begin
        for i in 0 to 7 loop
            tmp <= bit_vector((to_unsigned(i,3)));
            A <= tmp(0);
            B <= tmp(1);
            C <= tmp(2);
            wait for 10ns;
        end loop;
        wait;
    end process;
end Behavioral;
```

Question 3: (Sequential Logic + VHDL) 15 marks

a) In this question, please provide the truth table of a T-typed Flip-flop.



Q(n)	T	Q(n+1)	Q'(n+1)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

b) Please provide the VHDL module for the above T-FF "Q3".

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity t_flipflop is
    port (T: in std_logic;
          CLK : in std_logic;
          Q: out std_logic);
end t_flipflop;

architecture beh_t_trigger of t_flipflop is
    signal temp: std_logic;
begin
    process (Reset,CLK)
    begin
        if (rising_edge(CLK)) then
            temp <= T xor temp;
        end if;
    end process;

    Q <= temp;
end beh_t_trigger;
    
```

c) Please provide the VHDL simulation module for the above T-FF “Q3_tb”.

```
library ieee;
use ieee.std_logic_1164.all;

entity tb_t_flipflop is
end tb_t_flipflop;

architecture tb of tb_t_flipflop is

    component t_flipflop
        port (T: in std_logic;
              CLK : in std_logic;
              Q  : out std_logic);
    end component;

    signal T: std_logic;
    signal CLK : std_logic;
    signal Q  : std_logic;

    constant TbPeriod : time := 10 ns; -- EDIT Put right period here
    signal TbClock : std_logic := '0';
    signal TbSimEnded : std_logic := '0';

begin

    dut : t_flipflop port map (T=>T, CLK => CLK, Q => Q);

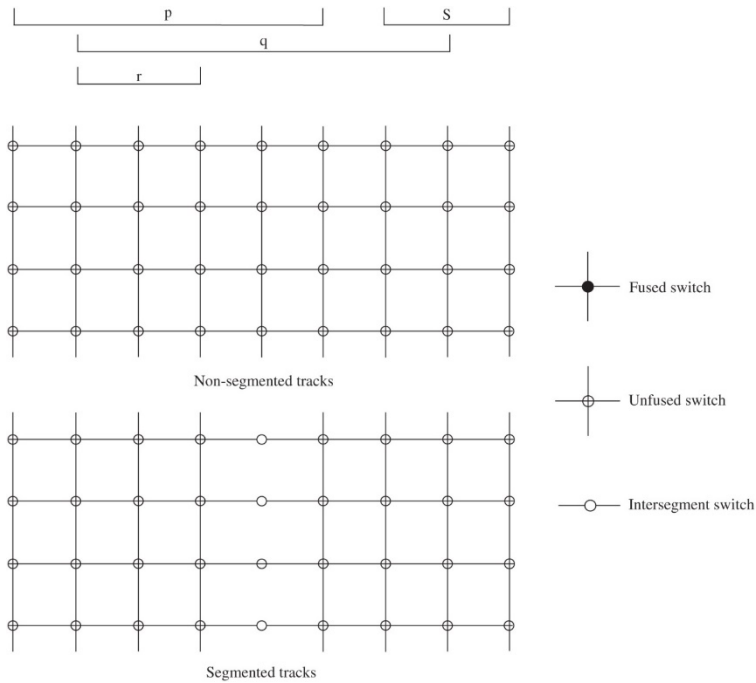
    -- Clock generation (Here can be generated easier)
    TbClock <= not TbClock after TbPeriod/2 when TbSimEnded /= '1' else '0';
    CLK <= TbClock;

    -- Students can only add the codes to define the value of T
    stimuli : process
    begin
        T <= '0';
        Wait for 15ns;
        T <= '1';
        Wait for 10ns;
        T <= '0';
        Wait for 20ns;
        TbSimEnded <= '1';
        wait;
    end process;

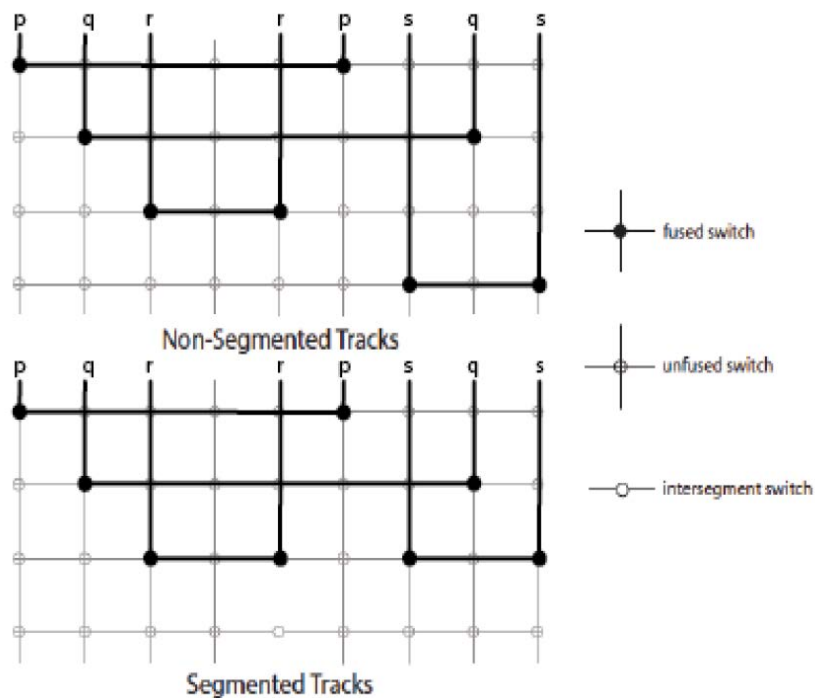
end tb;
```

Question 4: (FPGA) 10 marks

- Route the p, q, r, s nets on the non-segmented tracks shown below. Use the minimum number of tracks possible.
- Route the p, q, r, s nets on the segmented tracks shown below. Use the minimum number of tracks possible.



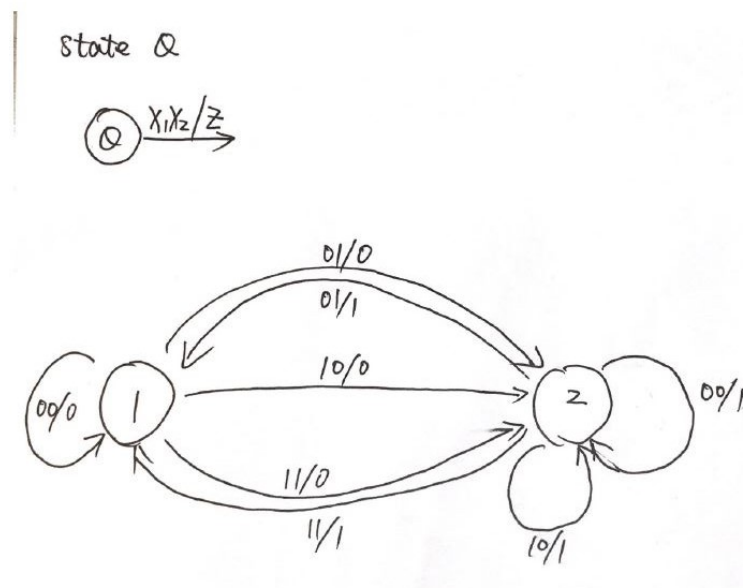
Suggested Answer:



Question 5: (State Machine) 10 marks

- a) A Moore sequential machine with two inputs (X_1 and X_2) and one output (Z) has the following state table. Please provide the VHDL module that describes the machine at the behavioral level. Assume that the state changes occur 10 ns after the falling edge of the clock, and output changes at the time when state changes. Draw the state diagram.

present state	next state				output (Z)
	$X_1X_2 = 00$	01	10	11	
1	1	2	2	2	0
2	2	1	2	1	1



- b) Write the VHDL code for the above State machine "Q5".

```

entity moore_mach is
    port(
        X1, X2: in bit;
        Clk: in bit;
        Z: out bit);
end moore_mach;

architecture moore_mach_bhv of moore_mach is
    signal state: natural := 1;
begin
    process(Clk)
    begin
        if Clk = '0' and Clk'event then
            case state is
                when 1 =>
                    if (X1 or X2) = '1' then
                        state <= 2 after 10 ns;
                        Z <= '0' after 20 ns; end if;
            end case;
        end if;
    end process;
end moore_mach_bhv;
  
```

```
        when 2 =>
            if X2 = '1' then
                state <= 1 after 10 ns;
                Z <= '1' after 20 ns; end if;
            when others => null;
        end case;
    end if;
end process;
end moore_mach_bhv;
```

You can use the back side of the paper, if needed.

~ End ~