Advanced Digital Design

Homework-2

Author: Neeha Hammad

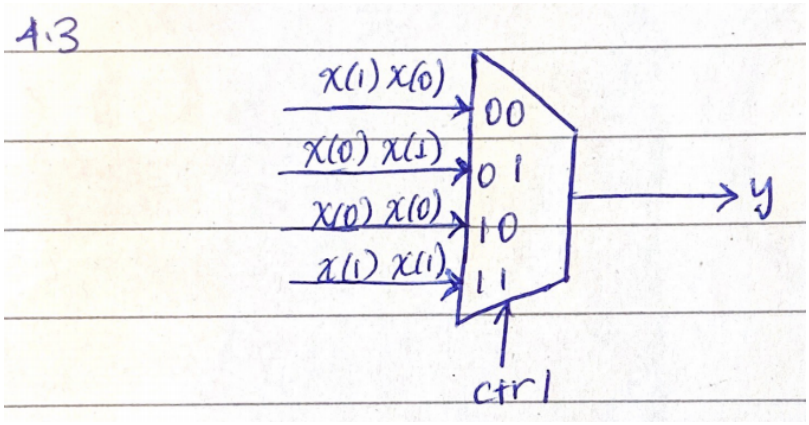
4.3 Consider a 2-by-2 switch. It has two input data ports, x(0) and x(l), and a 2-bit control signal, ctrl. The input data are routed to output ports y (0) and y (1) according to the ctrl signal. The function table is specified below.

Input ctrl	Output y1	y0	Function
00	x1	x0	pass
01	x0	x1	cross
10	x0	x0	broadcast x0
11	x1	x1	broadcast x1

(a) Use concurrent signal assignment statements to derive the circuit.

```
entity entity43 is
port (ctrl: in std_logic_vector (1 downto 0);
        x: in std_logic_vector (1 downto 0);
        y: out std_logic_vector (1 downto 0)
        ); end entity43
        architecture arch43 of entity43
        begin
                with ctrl select
                  y <= x(1)x(0) when "00",
                       x(0)x(1) when "01",
                       x(0)x(0) when "10",
                       x(1)x(1) when others;
end arch43
```

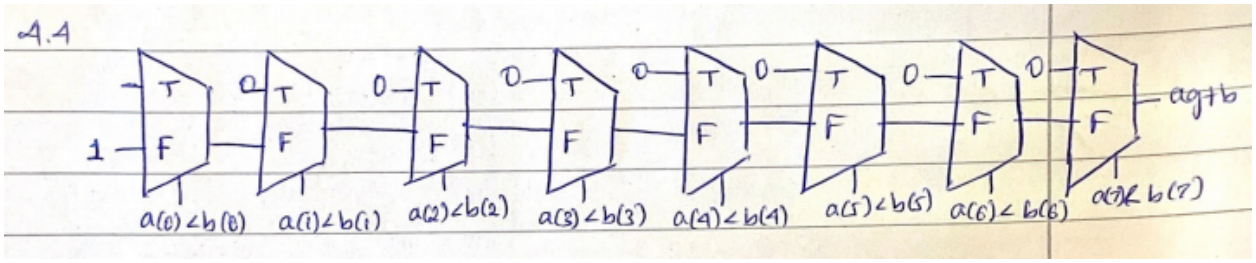
(b) Draw the conceptual diagram.



(c) Expand it into gate-level circuit and derive the simplified logic expression in sum-of-products format.

4.4 Consider a comparator with two 8-bit inputs, a and b. The a and b are with the std-logic-vector data type and are interpreted as unsigned integers. The comparator has an output, agtb, which is asserted when a is greater than b. Assume that only a single bit comparator is supported by synthesis software. Derive the circuit with concurrent signal assignment statement(s).

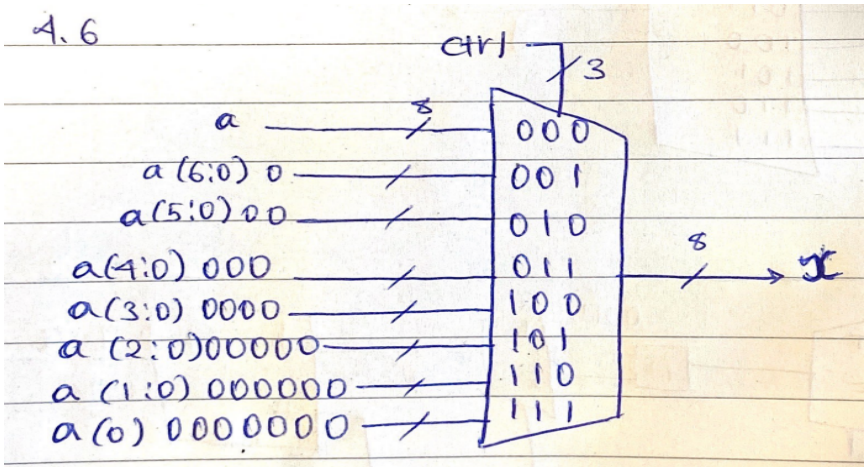
```
entity entity44 is
port (ctrl: in std_logic_vector (7 downto 0);
      a: in std_logic_vector (7 downto 0);
      b: out std_logic_vector (7 downto 0) );
end entity44
       architecture arch44 of entity44
       begin
         with ctrl select
         agtb <= '1' when a(7) > b(7) else if
                 '0' when a(7) <= b(7) else if
                 '1' when a(6) > b(6) else if
                 '0' when a(6) <= b(6) else if
                  '1' when a(5) > b(5) else if
                  '0' when a(5) <= b(5) else if
                  '1' when a(4) > b(4) else if
                 '0' when a(4) <= b(4) else if
                 '1' when a(3) > b(3) else if
                 '0' when a(3) <= b(3) else if
                 '1' when a(2) > b(2) else if
                 '0' when a(2) <= b(2) else if
                  '1' when a(1) > b(1) else
                  '0' when a(1) <= b(1)
end arch44;
```



4.6 We wish to design a shift-left circuit manually. The inputs include a, which is an 8-bit signal to be shifted, and ctrl, which is a 3-bit signal specifying the amount to be shifted. Both are with the std-logic-vector data type. The output y is an 8-bit signal with the std-logic-vector data type. Use concurrent signal assignment statements to derive the circuit and draw the conceptual diagram.

```
entity entity46 is
port (ctrl: in std_logic_vector (9 downto 0);
        a: in std_logic_vector (7 downto 0);
        x: out std_logic_vector (7 downto 0)
); end entity46

        architecture arch46 of entity46
        begin
        with ctrl select x <= a when "000";
                 a(6:0) when "001";
                a(5:0) when "010";
                a(4:0) when "011";
                a(3:0) when "100";
                a(2:0) when "101";
                 a(1:0) when "110";
                 a(0) when others;
end arch46;
```



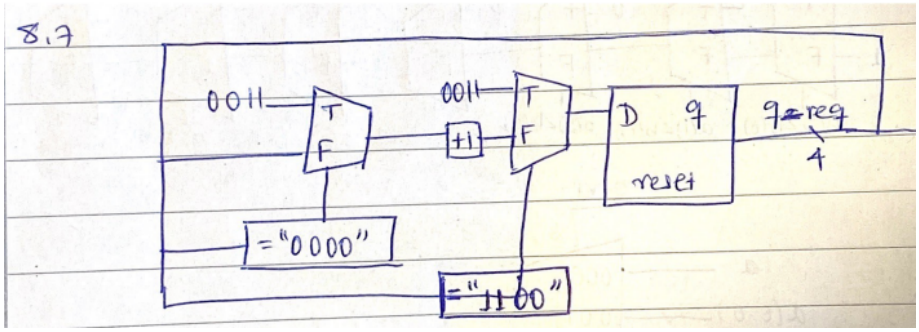
8.7 Consider a 4-bit counter that counts from 3 ("001 1") to 12 ("1 100") and then wraps around. If the counter enters an unused state (such as "0000") because of noise, it will restart from "001 1" at the next rising edge of the clock. Derive the VHDL code for this circuit and draw the conceptual top-level diagram.

```
entity entity87 is
port( clk: in std_logic;
        reset: in std_logic;
        q: out std_logic_vector (3 downto 0);
 ); end entity87

architecture arch87 of entity87
```

```
   signal q_reg: std_logic_vector (3 downto 0);
   signal q_next: std_logic_vector (3 downto 0);

begin
process (clk, reset)
   begin
                  if (reset='l') then
                  q-reg = "0011";
                  else if (clk= 'event' and clk= '1') then
                        q-reg <= q_next;
                  end if;
  end process;
  q_next <= "0011" when q_reg= "0000" else if
            "0011" when q_reg = "1100" else
            q_req+1;
   q<= q_next;
end arch87
```



8.8 Redesign the arbitrary counter of Section 8.5.3 using a mod-5 counter and special output decoding logic. Derive the VHDL code for this design.

Input pattern	Next pattern
000	011
011	110
110	101
101	111
111	000

```
entity arbi-seq-counter4 is
port ( clk, reset: in std-logic;
        q: out std-logic-vector (2 downto 0)
 ); end arbi-seq-counter4;
```

```
architecture two-seg-arch of arbi-seq-counter4 is
 signal r-reg:std-logic-vector (2 downto 0) ;
 signal r-next :std-logic-vector (2 downto 0) ;

begin
 -- register
process (clk, reset )
begin
        if (reset='l') then r-reg <= (others=>'0');
        elseif (clk ='event' and clk='l') then
        r_reg <= r_next;
        end if ;
end process;

--- next-state logic
r-next <= "001" when r-reg="000" else
          "010" when r-reg="001" else
          "011" when r-reg="010" else
          "100" when r-reg="011" else
          "000"; when r-reg="111"
--- output logic
q<= r_reg;
end two_seg_arch;
```

8.11 Consider the block diagram of the decade counter in Figure 8.13. Let T,, and TsetzLp of the D FF be 1 and 0.5 ns, and the propagation delays of the incrementor, comparator and multiplexer be 5, 3 and 0.75 ns respectively. Assume that no further optimization will be performed during synthesis. Determine the maximal clock rate.

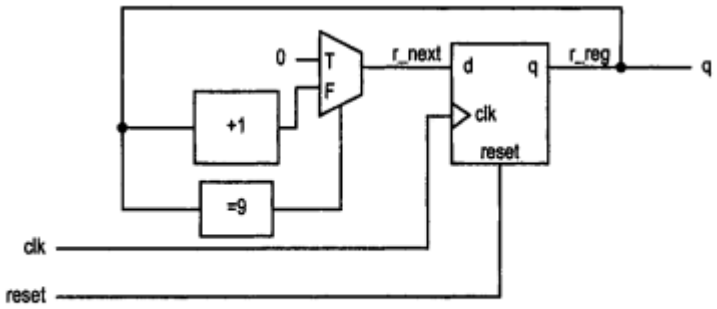


Figure 8.13 Conceptual diagram of a decade counter.

8.11, Maximal clock rate f_{max} is the inverse of minimum period of the clock $T_c(min)$, i.e $f_{max} = \frac{1}{T_c(min)}$

$$T_{cq} + T_{setup} + T_{next} \leq T_c$$

$$T_c(min) = T_{cq} + T_{setup} + T_{next}(max)$$

$$T_{next}(max) = max(5,3) + 0.75 = 5.75ns$$

$$T_c(min) = 1 + 5.75 + 0.5$$

$$= 7.25ns$$

$$\therefore f_{max} = \frac{1}{7.25ns} = 137.9MHz$$