(Samarth Aggarwal)
2016CS10395

COL215 Digital Logic and System Design
Minor Test 2

06.10.2017 **Max Marks: 10**

1. 2's complement of an n-bit binary number $B = b_{n-1}\, b_{n-2} \ldots b_1\, b_0$ is given by

(a) $b'_{n-1}\, b'_{n-2} \ldots b'_{k+1}\, b_k\, b_{k-1} \ldots b_1\, b_0$, where b_k is the rightmost 0 in B. ×
(b) $b'_{n-1}\, b'_{n-2} \ldots b'_{k+1}\, b_k\, b_{k-1} \ldots b_1\, b_0$, where b_k is the rightmost 1 in B. ✓
(c) $b_{n-1}\, b_{n-2} \ldots b_{k+1}\, b'_k\, b'_{k-1} \ldots b'_1\, b'_0$, where b_k is the rightmost 0 in B.
(d) $b_{n-1}\, b_{n-2} \ldots b_{k+1}\, b'_k\, b'_{k-1} \ldots b'_1\, b'_0$, where b_k is the rightmost 1 in B.

Which one of the above is correct? Give a proof. Here b_0 is the least significant bit.

[2]

2. Consider the algorithm given below for dividing 16 bit unsigned *Dividend* by 16 bit unsigned *Divisor*. The *Quotient* and *Remainder* are also 16 bit. R and D are 32 bit integers, R is signed and D is unsigned. Give a VHDL implementation of this algorithm such that it takes 18 clock cycles – one cycle for each iteration of the loop and one cycle each for the initialization step and the final step. Do not use variables, LOOP, WAIT and AFTER.

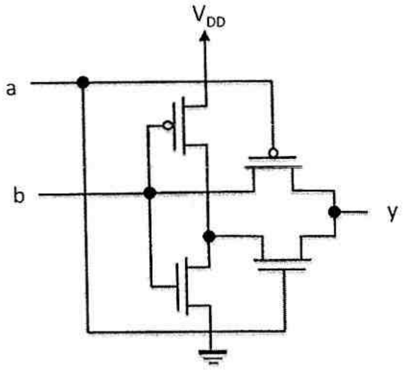
n=16

```
Initialization step:    R = Dividend; Q = 0; D = Divisor x $2^{n-1}$
Loop:  for i = 0 to 15 do {
            if (R < 0) R = R + D else R = R - D
            Q = 2 x Q + s        where s is the sign bit of adder/subtractor output
            D = D / 2
       }
Final step:    if (R < 0) Remainder = R + ~~B~~ Divisor else Remainder = R
               Quotient = Q
```

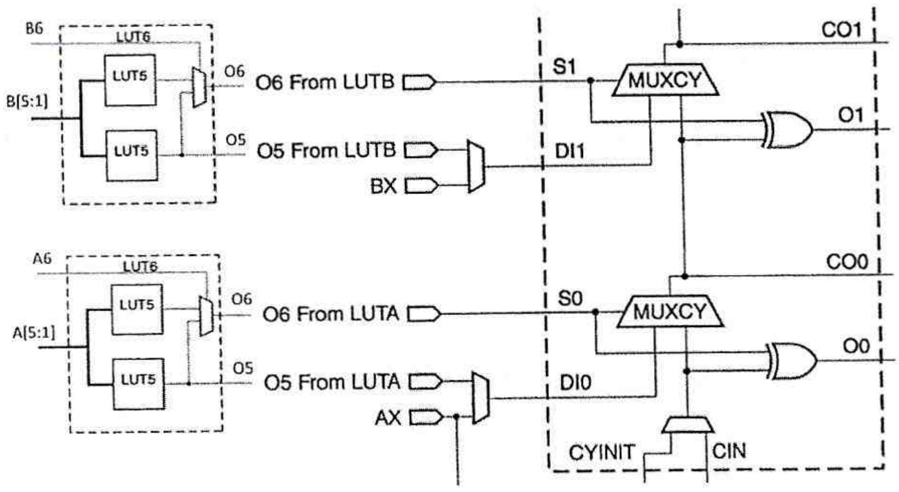
[3]

3. What logic gate is realized by the circuit shown? Does the circuit suffer from any major drawbacks? Give reasons for your answer.

[3]



4. The figure shows a part of a slice of Spartan 6 series FPGAs. On the right side, a portion of the carry chain circuit (2 bits out of 4) is shown and on the left side, two of the 6-input LUTs are shown. Note that each 6-input LUT consists of two 5-input LUTs. What logic functions should be implemented by the LUTs shown to make a 2-bit portion of a fast carry propagate adder?



[2]

A1) Let C = (2's complement of B)

$\Rightarrow C = \bar{B} + 1$

$\bar{B} = b'_{n-1} b'_{n-2} \cdots b'_{k} b'_{k-1} \cdots b'_1 b'_0$

Adding 1 to a bit, inverts the bit. Also, a carry is generated only if the original bit was 1. Hence, on adding 1 to a bit that is 1 will continue the chain while adding 1 ~~to~~ a bit that is 0 will terminate the chain. So this chain continues only till the first 0 is encountered from right. That bit would have been 1 ~~before~~ in B as we are adding 1 to $\bar{B}$.

Let b_k be the rightmost 1 in B.

So all bits ~~to~~ right of b_k are 0.

$$\Rightarrow B = b_{n-1} b_{n-2} \cdots b_{k+1} \underset{b_k}{1} \underbrace{000\cdots0}_{k \text{ bits}}$$

$$\bar{B} = b'_{n-1} b'_{n-2} \cdots b'_{k+1} \underset{b'_k}{0} \underbrace{111\cdots1}_{k\text{-bits}}$$

~~$C = \bar{B} + 1$~~

~~$\bar{B} = b'_{n-1} b'_{n-2} \cdots b'_{k+1}$~~

$$\bar{B} = \sum_{i=k+1}^{n-1} b'_i 2^i + \sum_{i=0}^{k-1} 2^i = \sum_{i=k+1}^{n-1} b'_i 2^i + 2^k - 1$$

$$\bar{B} + 1 = \sum_{i=k+1}^{n-1} b'_i 2^i + 2^k$$

$$\bar{B} + 1 = b'_{n-1} b'_{n-2} \cdots b'_{k+1} \underset{b_k}{1} \underbrace{00\cdots0}_{k\text{-bits}}$$

$$\bar{B} + 1 = b'_{n-1} b'_{n-2} \cdots b'_{k+1} b_k b_{k-1} \cdots b_1 b_0$$

Hence option B is correct.

Sheet 1 (Samarth Aggarwal 2016CS10395)

A2)
```
use IEEE
entity divider is
  ~~port (RD: in integer;)~~
  port ( dividend, divisor : in integer; clk : in bit;
               quotient, remainder, : out bit_vectors(16 downto
                                                             0)
               ~~clk : in~~
         );
end divider
architecture arc of divider is
   signal r  : ~~integer bit_vector~~ integer ;
   signal d  : integer;
   signal r1 : bit vector (31 downto 0)
   signal d1 : bit_vector(31 downto 0)
   signal q  : bit_vector (15 downto 0)




begin
 process (clk, dividend, divisor)
 begin
  if clk'event and clk='1' then
     if dividend'event or divisor'event then
       r <= dividend;
       d <= divisor ;
       q <= "0000000000000000";
         r1 <= q & q ;
       d1 <= q & q ;
       for i in 0 to 15 generate
          r1 <= r1 + (r mod 2);
          r <= r/2 ;
          d1 <= d1 + (d mod 2);
          d <= d/2 ;
       end generate
     d <= d (15downto 0) & "0000000000000000";
   ~~end if~~ end if
```

else then

Sheet 2 (Samarth Aggarwal 2016CS10395)

A3) ~~the~~ $y = \bar{a}b + a\bar{b}$

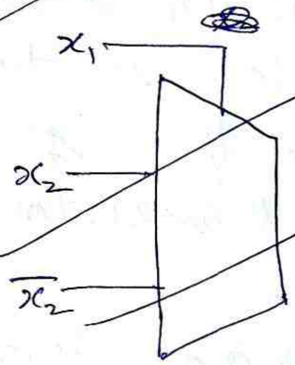
The XOR logic gate is realised because when a=1, the NMOS transistor is turned on and it connects ~~y~~ to output ~~of~~ ~~the~~ CMOS inverter. When a=0, PMOS transistor is on and it connects y to b.

This circuit suffers a major drawback in 2 cases:-

I) When a=1, b=0 ⇒ In this case, the NMOS transistor is used to pass 1 to the other side which the NMOS does ~~not do~~ with a drop. So here y will be $(V_{dd} - V_T)$ where V_T is the threshold voltage of NMOS.

II) When a=0, b=0, ⇒ Here PMOS transistor is used to pass a 0 to the other side. Since PMOS ~~passes~~ ~~a~~ does not pass 0 efficiently hence this is drawback. y will attain the value V_T here as the PMOS turns off if y goes below that value.

~~A4) $f = x_1\overline{x_2} + \overline{x_1}\,x_2$~~



~~LUTs should implement~~

A4) $f = x_1 x_2 x_3 + x_1\overline{x_2}\,\overline{x_3} + \overline{x_1}\,x_2\overline{x_3} + \overline{x_1}\,\overline{x_2}\,x_3$

LUT 1 should implement $(x_1 \oplus x_2 \oplus x_3)$

& LUT 2 should implement $(x_1 x_2 + x_2 x_3 + x_3 x_1)$.