***King Fahd University of Petroleum and Minerals***

***College of Computer Science and Engineering***

***Computer Engineering Department***

**COE 202: Digital Logic Design (3-0-3)**

**Term 151 (Fall 2015-2016)**

**Major Exam 2**

**Saturday Nov. 21, 2015**

**Time: 120 minutes, Total Pages: 12**

**Name:\_\_KEY\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ID:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Section: \_\_\_\_\_\_\_**

**Notes:**

* Do not open the exam book until instructed
* **No Calculators are allowed** (*basic, advanced, cell phones, etc*.)
* Answer all questions
* All steps must be shown
* Any assumptions made must be clearly stated

|  |  |  |
| --- | --- | --- |
| **Question** | **Maximum Points** | **Your Points** |
| **1** | **7** |  |
| **2** | **12** |  |
| **3** | **7** |  |
| **4** | **12** |  |
| **5** | **16** |  |
| **6** | **8** |  |
| **7** | **11** |  |
| **Total** | **73** |  |

**Question 1. Choose the correct answer (one answer only) (7 Points)**

1. Which of the following represents a 4-input XNOR function?

a) **b)**

c) d)

1. NOR-OR (NOR first level and OR second level) function implementation is equivalent to:
   1. NAND-OR
   2. AND-NOR
   3. NOR-AND
   4. **OR-NAND**

|  |  |  |  |
| --- | --- | --- | --- |
| X |  |  | 1 |
|
| 1 |  |  | 1 |
|

1. Minimizing the shown k-map results in:
   1. 2 terms, 2-variable each
   2. 2 terms, 1-variable each
   3. 1 term with 2 variables
   4. **1 term with 1 variable**
2. Considering , which of the following represents a single prime implicant having the largest area in a k-map (i.e., the largest group of 1's):

**Question 2. (12 Points)**

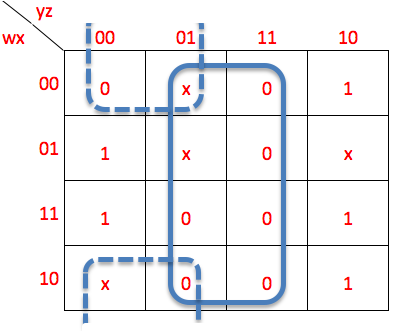
1. Represent in the k-map shown below

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | *y z* | | | |
|  |  | 00 | 01 | 11 | 10 |
| *x* | 0 | 1 | 0 | 1 | 0 |
|
| 1 | 0 | 1 | 1 | 0 |
|

1. Given shown in the k-map

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | CD |  |  |  |  |  | | | |
| AB |  | 00 | | 01 | | 11 | | 10 | |
|  | 00 |  |  |  |  |  |  |  |  |
|  |  | |  | |  | |  | |
|  | 01 |  |  |  |  |  |  |  |  |
|  | 1 | | 1 | |  | |  | |
|  | 11 |  |  |  |  |  |  |  |  |
| 1 | |  | | 1 | | 1 | |
| 10 |  |  |  |  |  |  |  |  |
|  | | 1 | | 1 | | 1 | |

* 1. List all essential prime implicants
  2. Obtain minimized SOP expression of F

1. Given function with don’t care conditions
   1. Use k-maps to provide minimized POS expression for F

* 1. Implement F using minimum number of 2-input NOR gates



**Question 3. (7 Points)**

Given an n-bit signed 2's complement number, **X**,it is required to design an iterative combinational circuit to compute the 2's complement of **X**.

## Sow the inputs and outputs of the 1-bit 2's complement iterative cell to be used for designing the n-bit 2's complement circuit. (2 Points)

## Show the truth table of the 1-bit 2's complement cell. (2 Points)

## Obtain simplified equations for the outputs of the 1-bit 2's complement cell using only the following gate types: NOT, AND, OR, XOR. (2 Points)

## Using the 1-bit 2's complement cell, draw a block diagram for a circuit to compute the 2's complement of a 3-bit number X. (1 Point)

We will us a signal (One) that propagates between cells to indicate whether we have got one or not.

1-bit

2's comp.

Onei-1

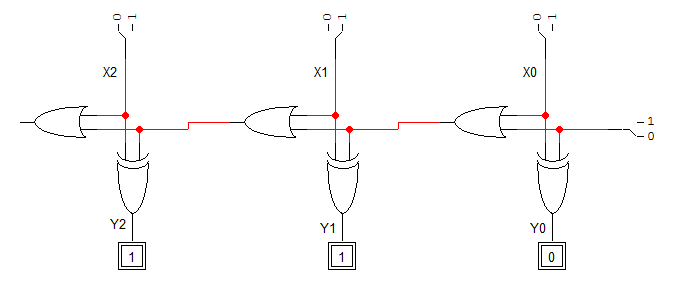
Onei

Xi

Yi

|  |  |  |  |
| --- | --- | --- | --- |
| **Onei-1** | **Xi** | **Onei** | **Yi** |
| **0** | **0** | **0** | **0** |
| **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **0** |

**Yi = Xi ⊕ Onei-1 Onei = Onei-1 + Xi**

****

**Alternative Solution:**

We will us a signal (Cin) that propagates between cells to indicate whether we have carry or not. The 2's complement will be computed as the 1's complement + 1.

1-bit

2's comp.

Ci-1

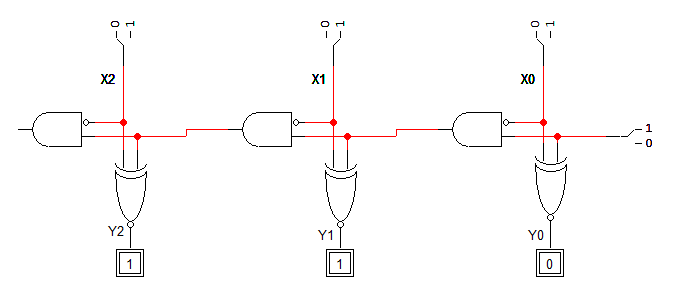
Xi

Ci

Yi

|  |  |  |  |
| --- | --- | --- | --- |
| **Ci-1** | **Xi** | **Ci** | **Yi** |
| **0** | **0** | **0** | **1** |
| **0** | **1** | **0** | **0** |
| **1** | **0** | **1** | **0** |
| **1** | **1** | **0** | **1** |

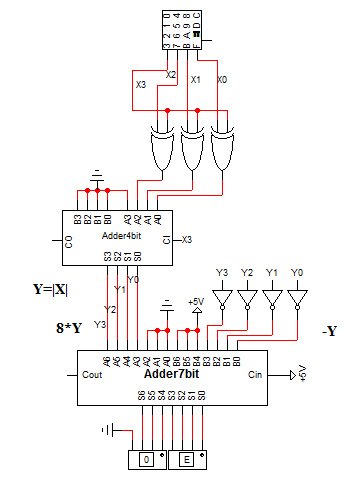
**Yi = (Xi ⊕ Ci-1)' Ci = Ci-1 Xi'**

****

**Question 4. (12 Points)**

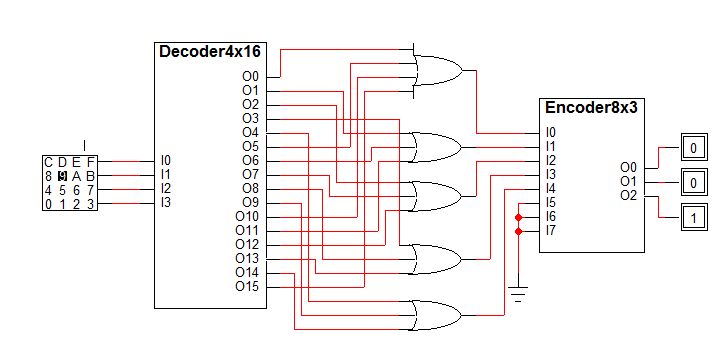
## It is required to design a circuit to compute the equation **Y=|7\*X|**, i.e., Y is equal to the absolute value of 7\*X, where X is a **4-bit signed number in 2's complement representation**. Your circuit should be designed using the minimum number and sizes of the following MSI components (Adders, Multiplexers) and additional logic gates if needed. Show clearly the size of all used components.

## (7 Points)



## It is required to design a circuit to compute the equation **Y=X mod 5**, i.e. Y is the remainder of dividing X by 5, where is **X is a 4-bit unsigned number**. For example, 9 mod 5=4 and 10 mod 5=0. Your circuit should be designed using the minimum number and sizes of the following MSI components (Decoder, Encoder) and additional logic gates if needed. Show clearly the size of all used components.

## (5 Points)



**Question 5. (16 Points)**

1. Fill in all blank cells in the table below. [4 points]

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Binary (6-bits) | Equivalent decimal value with the binary interpreted as: | | | |
| Unsigned integer | Signed-magnitude number | Signed-1’s complement number | Signed-2’s complement number |
| 110110 | **54** | **-22** | **-9** | **-10** |

1. Fill in all blank cells in the table below. [6 points]

|  |  |  |  |
| --- | --- | --- | --- |
| Decimal | Binary representation in **6 bits:** | | |
| Signed-magnitude representation | Signed-1’s complement representation | Signed-2’s complement representation |
| + 29 | **011101** | **011101** | **011101** |
| - 29 | **111101** | **100010** | **100011** |
|  | Binary representation in **8 bits:** | | |
| Signed-magnitude representation | Signed-1’s complement representation | Signed-2’s complement representation |
| - 29 | **10011101** | **11100010** | **11100011** |

1. Show how the following arithmetic operations are performed using 6-bit signed 2’s-complement system. Check for overflow and mark clearly any overflow occurrences. [6 points]

|  |  |
| --- | --- |
| (1)  110100 **110100**  **-** 111110 **+ 000010**  **110110**    Overflow: Yes/**No** | (2)  110111  +111000  **101111**    Overflow: Yes/**No** |
| (3)  111110  + 111111  **111101**  Overflow: Yes/**No** | (4)  001101 **001101**  - 111101 **+ 000011**  **010000**    Overflow: Yes/**No** |

**Question 6. (8 Points)**

1. Show the implementation of the function F(x,y,z) = ∑m(0,2,5,6,7) [6 points]
2. Using a single MUX of minimum size
3. Using a minimum size decoder and a single gate with minimum number of inputs
   * + - 1. **F has 3 I/Ps 🡪 4-to-1 MUX:**

**4x1**

**MUX**

S0

**0**

**1**

**2**

**3**

Z’

S1

Z

Z'

1

X

F

Y

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **X** | **Y** | **Z** | **F** | **F** |
| **0** | **0** | **0** | **1** | **Z’** |
| **0** | **0** | **1** | **0** |
| **0** | **1** | **0** | **1** | **Z’** |
| **0** | **1** | **1** | **0** |
| **1** | **0** | **0** | **0** | **Z** |
| **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **1** | **1** |

* + - * 1. **Since # of Maxterms is smaller than the # of Minterms, we implement F as POM. Note that Mi = m’i , so we need to invert-AND F’s Maxterms, i.e. NOR them:**

**I0 m0**

**m1**

**I1 m2**

**m3**

**I2 m4**

**m5**

**m6**

**m7**

x

y

z

F

1. Implement a 4-to-1 MUX using a minimum number of 2-to-1 MUXs. Clearly mark and label all inputs. [2 points]

S0

**0**

**1**

D0

S1

D2

D1

D3

Y

**0**

**1**

**0**

**1**

**Question 7. (11 Points)**

## A piece of hardware is described as a Verilog module. One of the given below Verilog codes is the correct description of this piece:

1. Indicate which of these codes is valid and which is invalid fully justifying your answer? (3 Points)
2. Give the logic diagram of this piece of hardware? (3 Points)

|  |  |
| --- | --- |
| **VALID**  **q70001.jpg** | **module** V\_Q7A (**output** **reg** z, u, **input** x, w, t, q);  **wire** v, y;  **assign** y = (q==1'b1) ? v : w; //if q=1 y=v else y=w  **always** **@(**x,y,t)  **begin**  z = (x^y)^t;  u = (x&y) | (x&t) | (y&t);  **end**  **not**  (v , w); //inverter gate instance  **endmodule** |
| **INVALID**  (Continuous Assignment cannot be included within an always block) | **module** V\_Q7B (**output** **reg** z, u, **input** x, w, t, q);  **wire** v;  **reg** y;  **always** **@(**x,q,t)  **begin**  **assign** y=(q==1'b1)? v:w; //if q=1 y=v else y=w  z = (x^y)^t;  u = (x&y) | (x&t) | (y&t);  **end**  **not**  (v , w); //inverter gate instance  **endmodule** |
| **INVALID**  (INSTANTATION statement cannot be included within an always block) | **module** V\_Q7C (**output** **reg** z, u, **input** x, w, t, q);  **reg** v, y;  **always** **@(**x,q,t, v)  **begin**  **if** (q ==1'b1) y = v ; else y = w;  z = (x^y)^t;  u = (x&y) | (x&t) | (y&t);  **not**  (v , w); //inverter gate instance  **end**  **endmodule** |

## You are to write a test bench for the 4-bit adder module which has the following declaration: (5 Points)

**module** adder4 **(output reg [3** : 0**]** sum , **output reg** cout , **input [3** : 0**]** A , B**) ;**

|  |  |  |
| --- | --- | --- |
| **Time Unit** | **A** | **B** |
| 0 | 5 | 6 |
| 10 | 15 | 9 |
| 20 | 9 | 3 |
| 30 | 13 | 14 |

Use the shown test patterns

**module** tb\_add4 **() ;**

**wire** [3:0] sum **;**

**wire** cout **;**

**reg** [3:0] A, B **;**

**adder4** UUT1 (sum, cout, A, B) **;**

**initial**

**begin**

A=5 ; B = 6;

#10 A=15 ; B = 9;

#10 A=9 ; B = 3;

#10 A=13 ; B = 14;

**end**

**endmodule**

