***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 141 (Fall 2014)**

**Major Exam II**

**Saturday November 29, 2014**

**Time: 150 minutes, Total Pages: 11**

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

**Notes:**

* Do not open the exam book until instructed
* **Calculators are not 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** | **17** |  |
| **2** | **14** |  |
| **3** | **10** |  |
| **4** | **12** |  |
| **5** | **12** |  |
| **Total** | **65** |  |

**Question 1 (17 Points)**

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

For the given K-map representing the Boolean function F, answer the following questions:

1. Which one of the following is an Implicant of F:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Term | A’C’ | A’BD | AC | A’B’C’ | BCD’ |  |
| Implicant (Y/N) | N | Y | N | N | Y |  |

1. Which one of the following is a Prime Implicant (PI) of F:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Term | AC’ | A’BC | BC’D | C’D | AD’ |  |
| PI  (Y/N) | Y | Y | N | Y | Y |  |

1. Which one of the following is an Essential Prime Implicant (EPI) of F:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | C’D | A’BC | A’C’ | BC’D | AD’ |  |
| EPI  (Y/N) | Y | N | N | N | Y |  |

1. Obtain a simplified sum-of-product (SOP) expression for F.

F= AD’ + C’D + A’BC

1. The following Boolean expression F= AD + A’C’D’ is a simplified version of the expression F= A’B’C’D’ + ABCD + AB’C’D. Are there any don`t care conditions? If so, what are they?

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

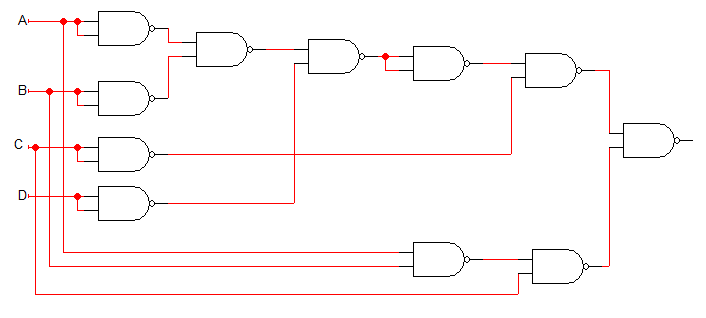
The don’t care conditions are:

A’BC’D’. ABC’D, AB’CD

1. Implement the circuit given below **using only 2-input NAND gates**. Redraw the circuit to obtain a multi-level NAND circuit implementation. Assume that only the true form of each input variable is available.



The implementation using only 2-input NAND gates is:



**Question 2. (14 Points)**

1. Fill in all blank cells in the two tables below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Binary | Equivalent decimal value with the binary interpreted as: | | | |
| Unsigned number | Signed-magnitude number | Signed-1’s complement number | Signed-2’s complement number |
| 10110110 | 182 | -54 | -73 | -74 |

|  |  |  |  |
| --- | --- | --- | --- |
| Decimal | Binary representation in 8 bits: | | |
| Signed-magnitude representation | Signed-1’s complement representation | Signed-2’s complement representation |
| + 100 | 01100100 | 01100100 | 01100100 |
| - 100 | 11100100 | 10011011 | 10011100 |

1. Show how the following arithmetic operations are performed using 5-bit signed 2’s-complement system. Check for overflow and mark clearly any overflow occurrences.

|  |  |
| --- | --- |
| (i)  01001 (+9)  - 11110 (-2)  01001 (+9)  + 00010 (+2)  -------------  01011 (+11)  Overflow: Yes/**No** | (ii)  10100 (-12)  + 11100 (-4)    10000 (-16)  Overflow: Yes/**No** |
| (iii)  11111 (-1)  + 11111 (-1)  11110 (-2)  Overflow: Yes/**No** | (iv)  01101 (+13)  - 11101 (-3)  01101 (+13)  + 00011 (+3)  -------------  10000 (-16)  Overflow: **Yes**/No |

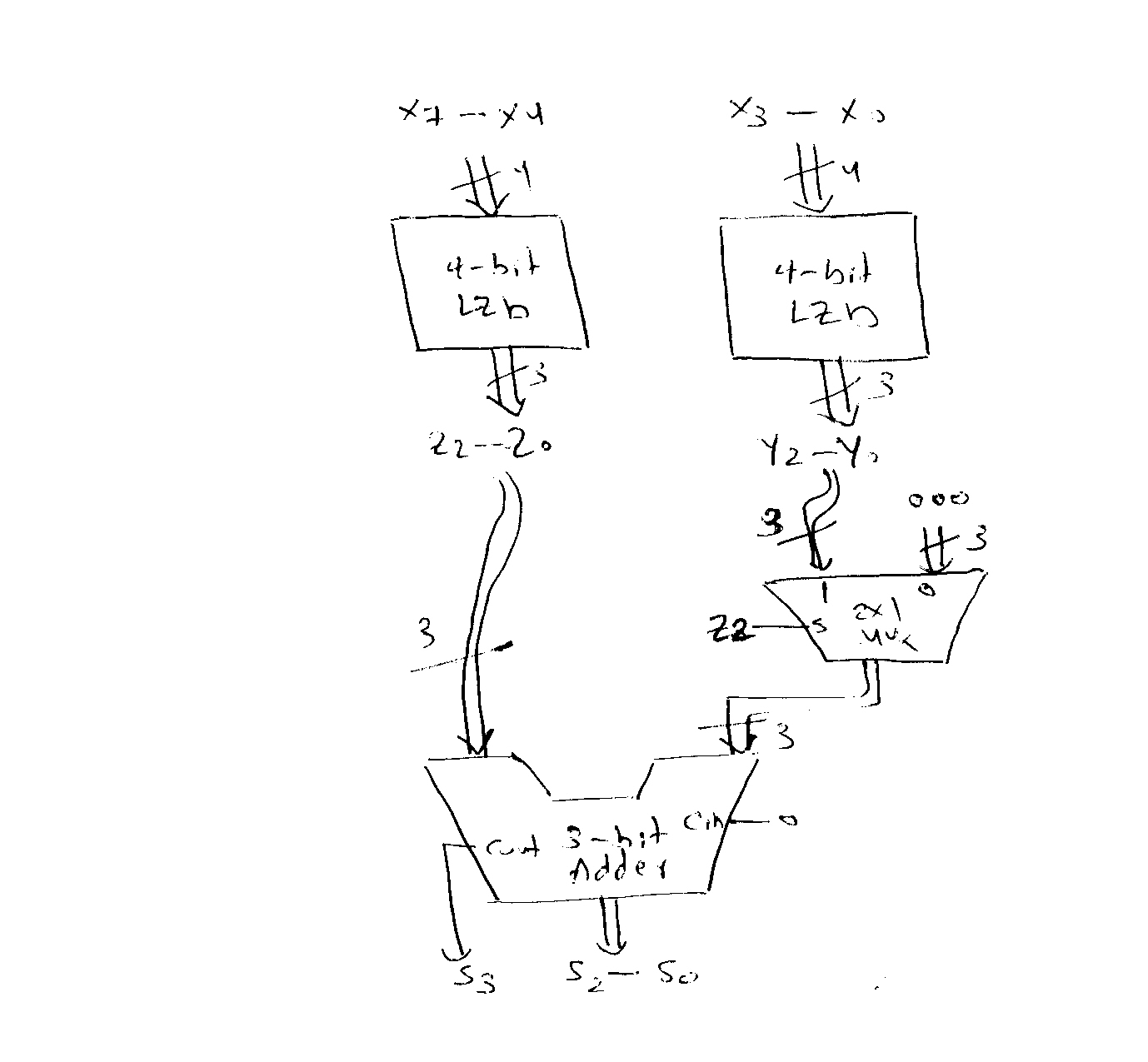
**Question 3. (10 points)**

1. It is required to design a combinational circuit that receives a 4-bit input number, X3X2X1X0, and computes the number of leading zero’s in the input. For example, if the input X3X2X1X0=0111 or X3X2X1X0=0100, the output should produce a result indicating that there is a single leading zero. Construct the truth table of the circuit. You do not need to derive the Boolean expressions of the outputs. (5 points)
2. Using a block diagram of the design of the 4-bit leading-zero detector circuit in (i) and any other needed MSI blocks (e.g. Adder, Comparator, Multiplexer, Decoder, etc.), design a combinational circuit that receives an 8-bit input number, X7X6X5X4X3X2X1X0, and computes the number of leading zero’s in the input. (5 points)

(i)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X3 | X2 | X1 | X0 | Z2 | Z1 | Z0 |
| 1 | x | x | x | 0 | 0 | 0 |
| 0 | 1 | x | x | 0 | 0 | 1 |
| 0 | 0 | 1 | x | 0 | 1 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |

(ii)



**Question 4. (12 Points)**

Using *only* the following modules:

* One 2-to-4 Decoder with enable,
* One 4-to-1 MUX,
* A maximum of five 1-to-2 DeMUXs /Decoders, and
* The minimum number of 2-input NAND gates (*If needed*)

Implement the following assuming that *signals are available only in the “True” but not the complement* form:

1. A 3-to-8 Decoder (*you may use this decoder as a black-box in solving* (ii) and/or (iii) below)
2. F1(a,b)= ab+a’b’
3. F2(a,b,c)= m0+ m1+m2+m4+m7

**Label all your signals (inside and outside MSI components).**

**(i)**

Knowing how to construct 2-to-4 decoder using 1-to-2 DeMuxs (3 points MAX)::

* Correct general structure (1 point)
* Correct general structure with accurate port labeling (3 points)

Knowing how to construct 3-to-8 decoder using 2-to-4 decoders (3 points MAX):

* Correct general structure (1 point)
* Correct general structure with accurate port labeling (2 points)
* Using NAND in place of inverter ( 1 point)



(ii)

Knowing how to implement a 2 variable function using 4-to-1 Mux (2 points MAX):

* Correct general structure (1 point)
* Correct general structure with accurate port labeling (2 points)



(iii)

Knowing how to implement a 2 variable function using 3-to-8 Decoder (4 points MAX):

1. Showing that it is more efficient to use 3-input NOR instead of 5-input OR (1 point)
2. Showing the implementation od 3-input NOR using 2-input NANDs (1 point)
3. Implementation of F2 (2 points MAX):

* Showing the correct implementation of F2 (1 point)
* Showing the correct implementation of F2 with accurate port labeling (2 point)

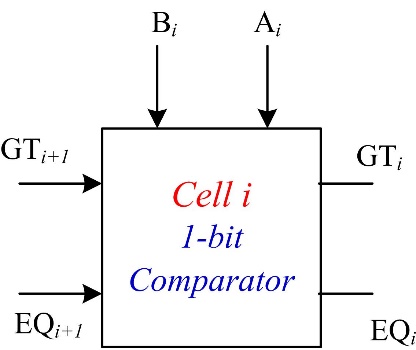


**Question 5. (12 Points)**

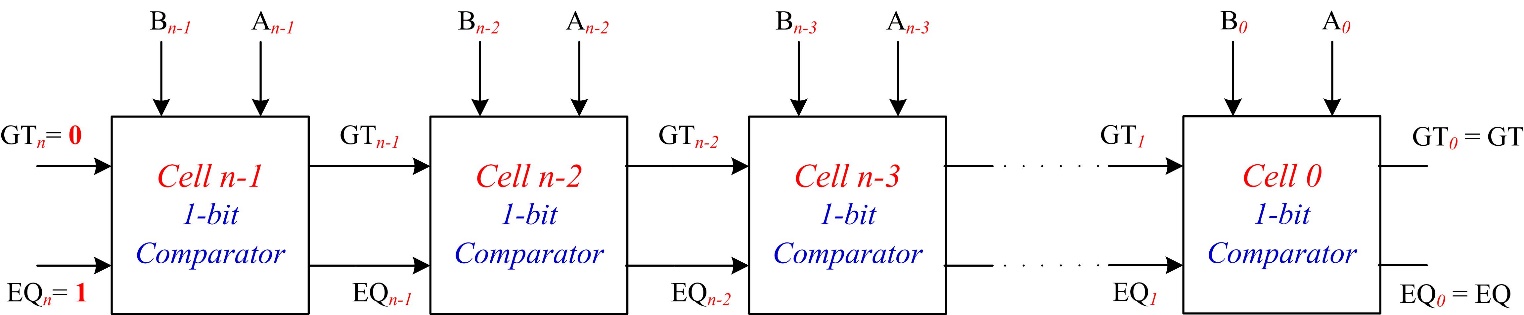
# It is required to design an *n*-bit magnitude comparator. The circuit receives two *n*-bit unsigned numbers ***A*** *and* ***B*** and produces two outputs **GT** and **EQ** *as* given in the table to the right.

|  |  |
| --- | --- |
|  | **GT EQ** |
| IF A > B | 1 0 |
| IF A = B | 0 1 |
| IF A < B | 0 0 |

# The input operands are processed in a bitwise manner *starting with the most significant bit (MSB)*. The comparator circuit is constructed using *n identical copies* of the basic 1-bit *cell* shown to the right. C*ell* ***i*** processes the ***i***th input bits (A*i* and B*i*) together with information passed to it from its predecessor cell (GT*i+1* and EQ*i+1*). It produces two output bits (GT*i* and EQ*i*). The *cell* output GT*i* =1 *iff* (**A*n-1* A*n-2*… A*i+1* A*i* >** **B*n-1* B*n-2*… B*i+1* B*i***) and EQ*i* =1 *iff* (**A*n-1* A*n-2*… A*i+1* A*i* =** **B*n-1* B*n-2*… B*i+1* B*i***).

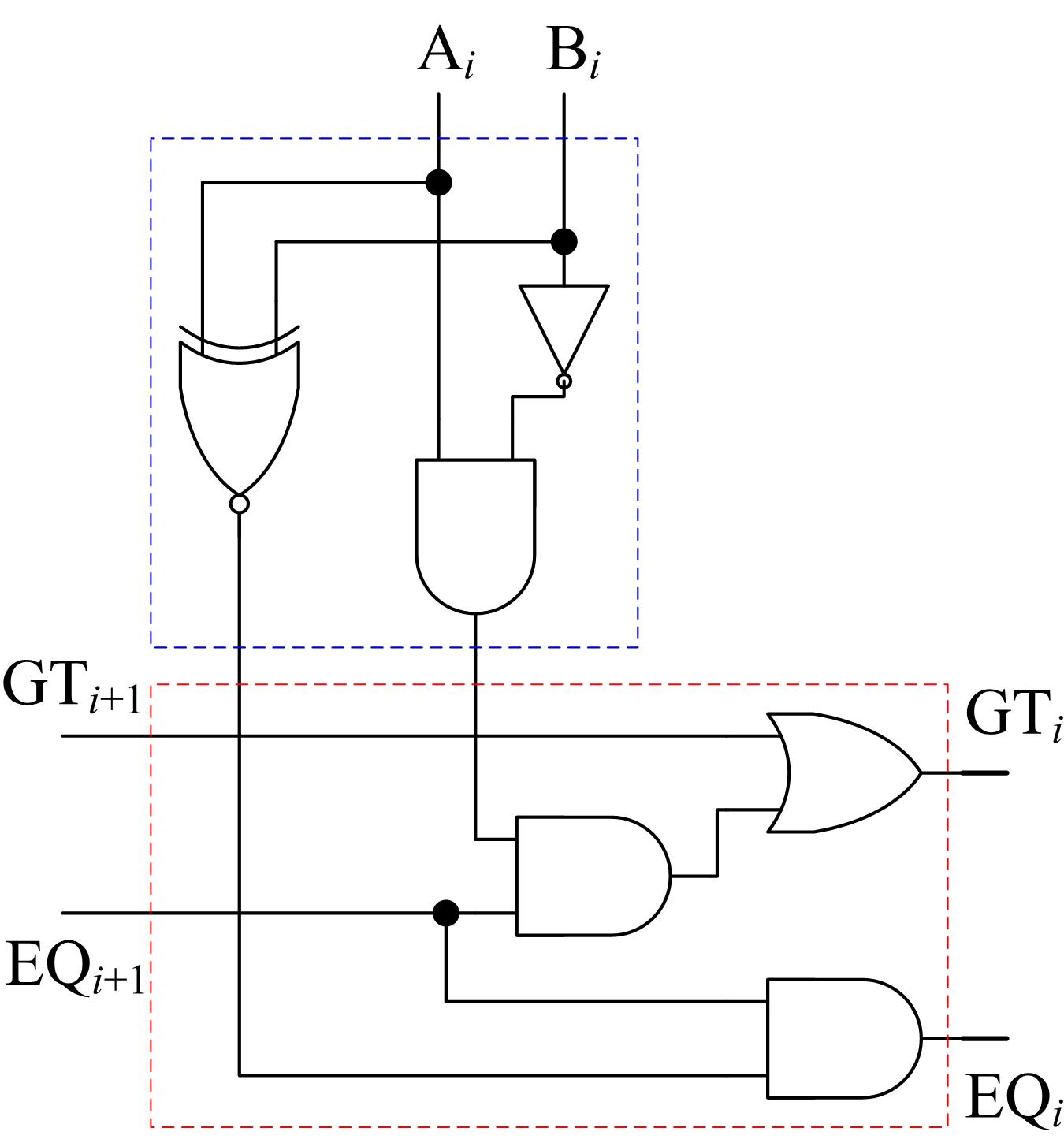


# The Figure below shows the *n*-bit comparator circuit implemented using *n* copies of the basic 1-bit cell. *The output of the n-bit comparator is that of the nth cell copy* (*cell 0; the least-significant*)*. Note that the inputs* ***GTn*** *and* ***EQn*** *to the first cell* (*cell n-1; the most significant*) *are set to* ***0 and 1*** *respectively as there are no more significant bits*.



# Boolean expressions of the outputs of ***cell i*** and its gate-level implementation are given below:

***,***and

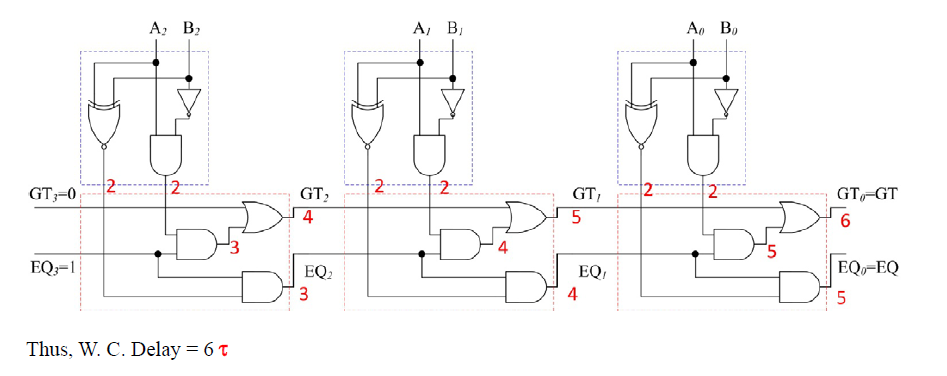


Cell *i* Gate Level Implementation

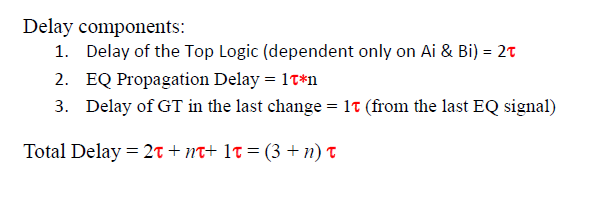
***EQi =* (*Ai* 🞊 *Bi*). *EQi+1***

Assuming that the **XOR** and **XNOR** gates have a delay of**2** while** *all OTHER gates* (*including inverters*) *have a delay of* **1**** *,* calculate:

1. The worst case delay of the 3-bit comparator (as a function of ****) shown below (4 Points)

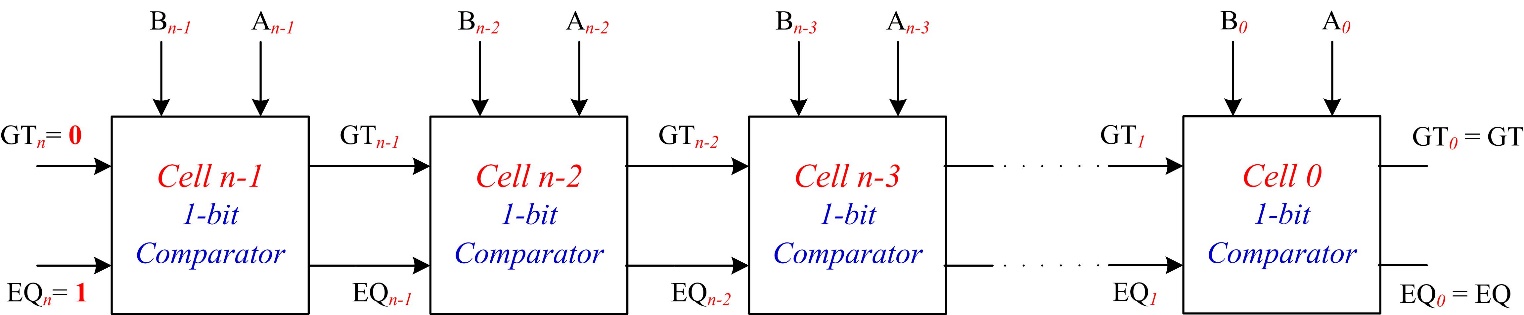


1. The worst case delay of an n-bit comparator (as a function of *n* and ****) (3 Points)



1. Suggest a design for a cascadeable 3-bit comparator with lookahead capability. What is the worst case delay of this unit (using the same delay model of **2** for XOR/XNOR gates and **1** for all other gates **(**irrespective of their fanin)? (5 Points)

*For convenience, the comparator circuit and Boolean expressions of the cell are repeated here.*



# Boolean expressions of the outputs of ***cell i*** and its gate-level implementation are given below:

***,***and

***EQi =* (*Ai* 🞊 *Bi*). *EQi+1***

