EE457_Discussion_Signed_Unsigned_Adder_subtracter_ques_r1.pdf

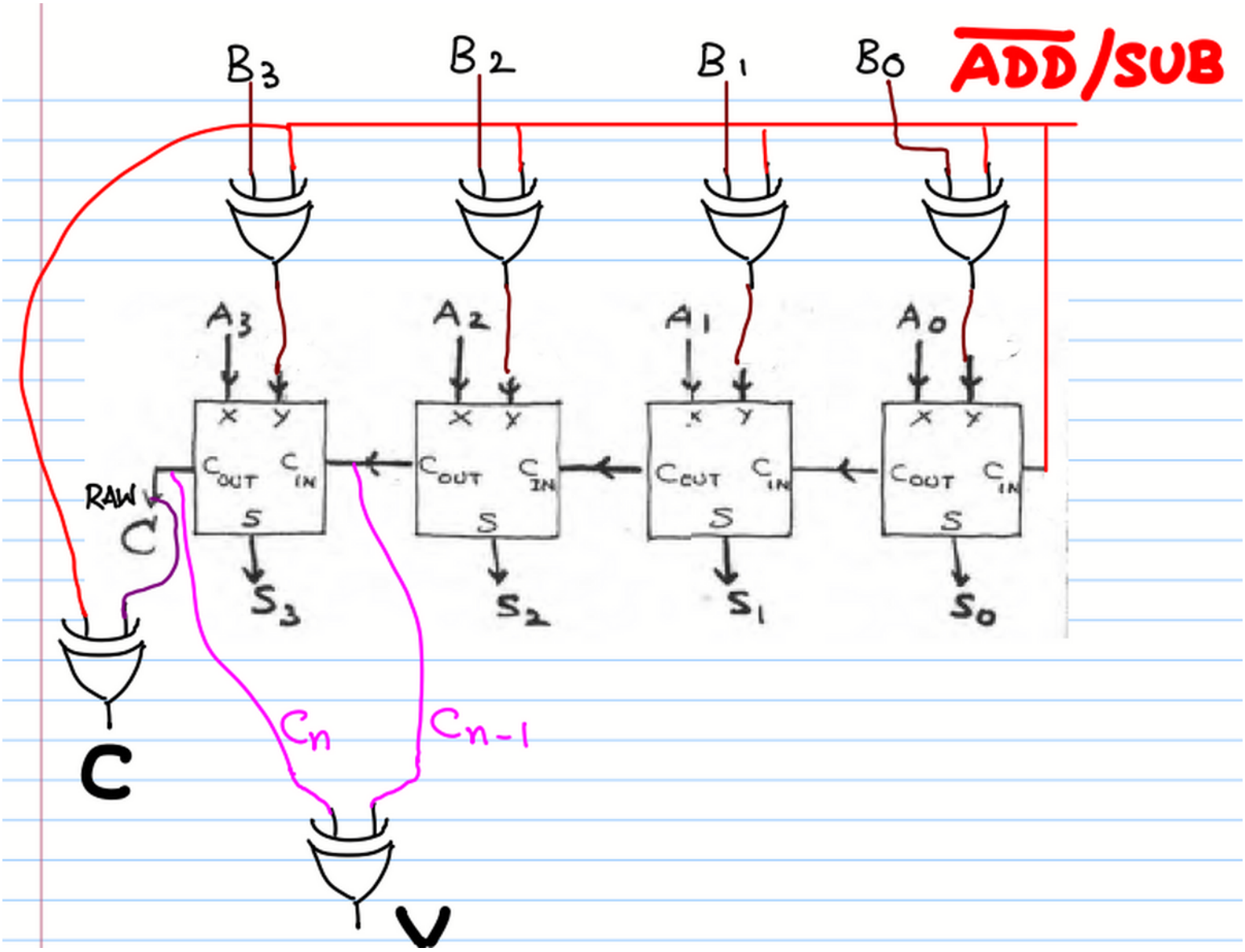
1. Adder-Subtracter design and overflow detection

2. Incrementer / Decrementer design using a regular adder and using half-adders

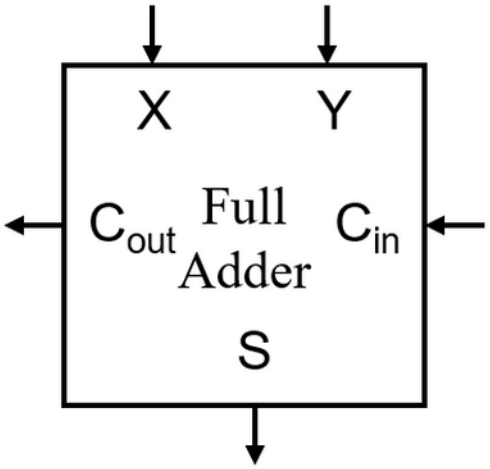
2.1 XOR and XNOR operations. Making a 6-bit adder using two 4-bit adders

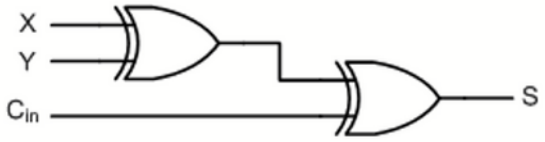
3. Positional weights -- sign extension -- multiply by 2 or 4 or any power of 2,
   Division by 2 or 4 or any power of 2 and the integer quotient

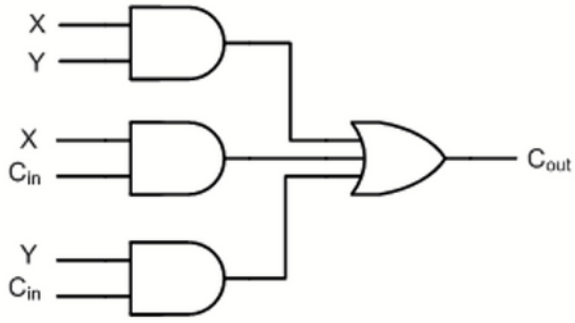
4. ALU Lab -- SLT -- SLTU -- Comparison unit design using a subtractor

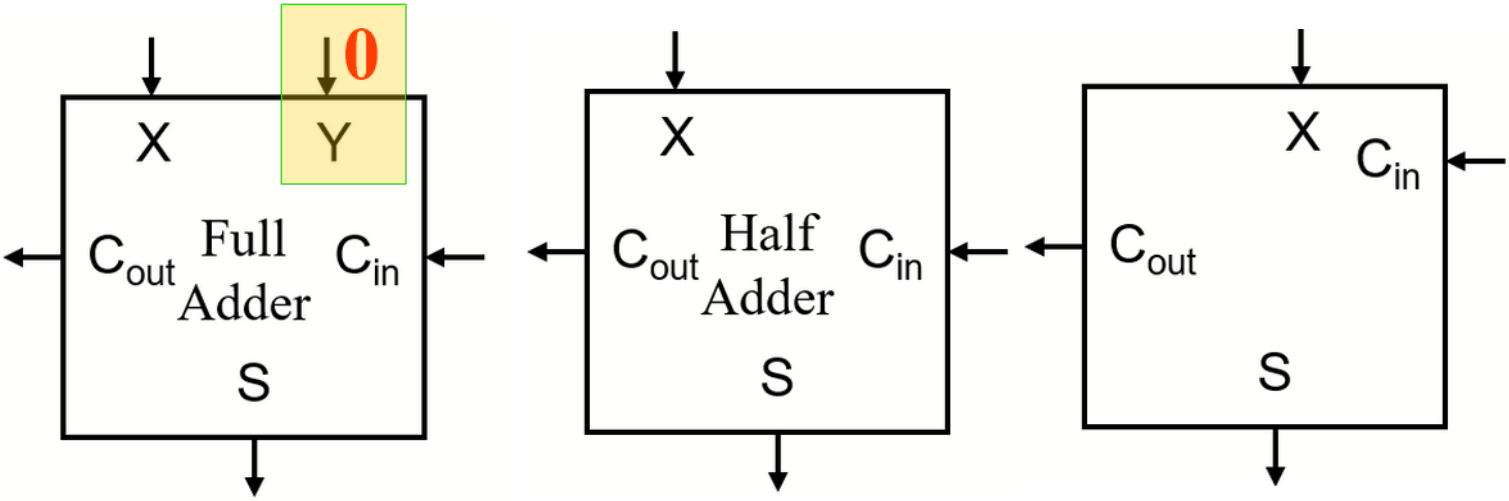


2. Incrementer / Decrementer design using a regular adder and using half-adders

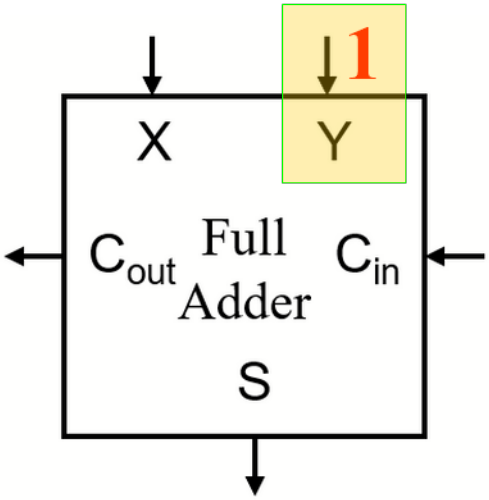


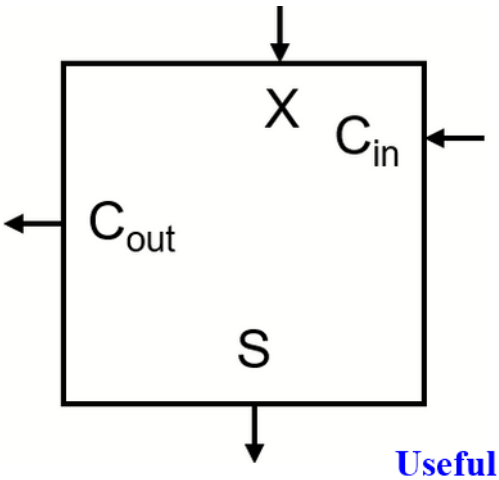






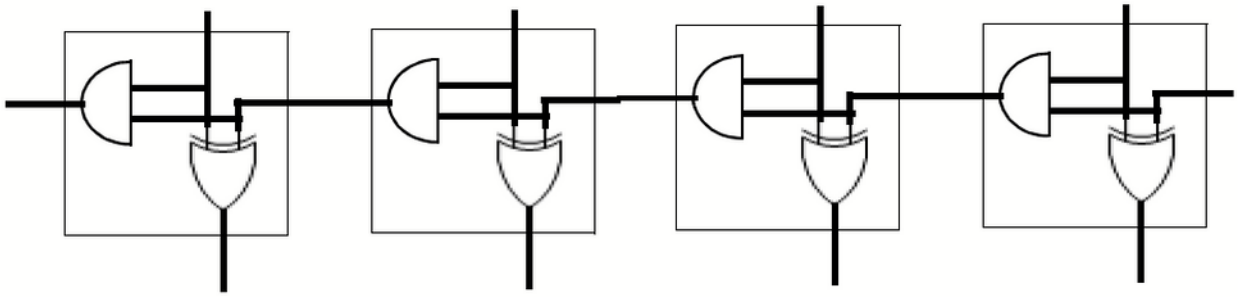
Useful in an incrementer design





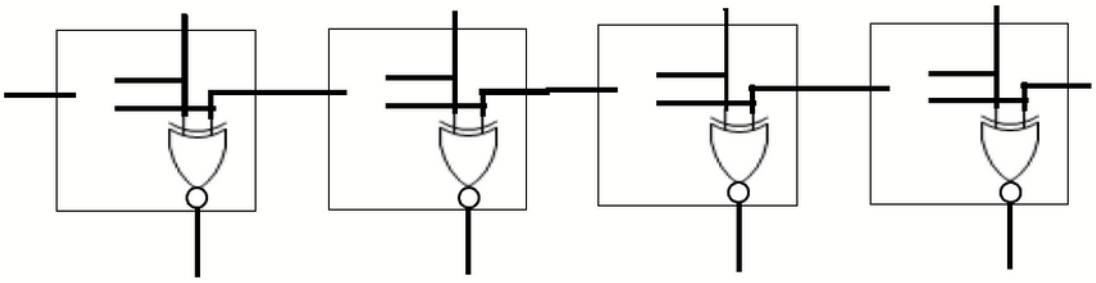
Useful in a decrementer design

Design a 4-bit incrementer Y[3:0] = X[3:0] + 1
Simplify the first half-adder and also derive the unsigned overflow C (C= 1 => overflow) and the signed overflow V (V = 1 => overflow).



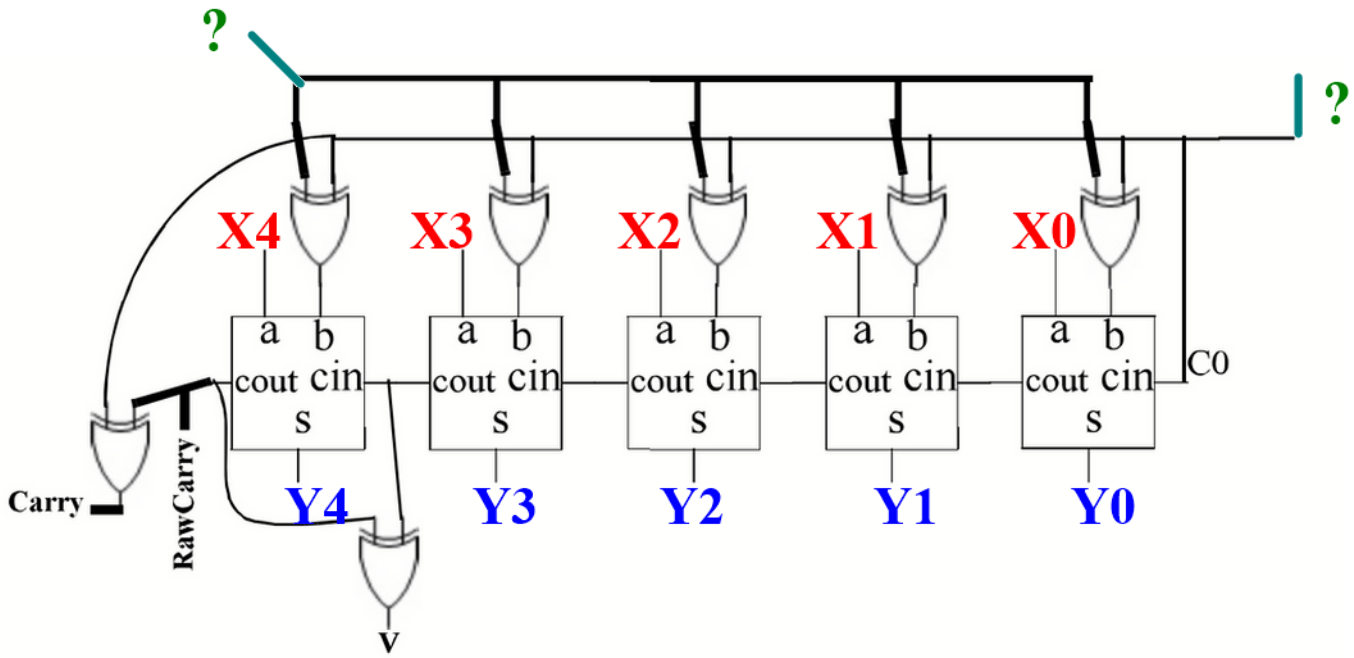
Incrementer

Design a 4-bit decrementer Y[3:0] = X[3:0] - 1
Simplify the first building block and also derive the unsigned overflow C (C= 1 => overflow) and the signed overflow V (V = 1 => overflow).



Decrementer

1.2 Here, we use a 5-bit adder/subtracter built using 5 full-adders to design an incrementer/decrementer. Given below is an incomplete design. complete it and label the control pin as **INC/DEC** or **INC/DEC**. This incrementer/decrementer is good for _____ (signed only / unsigned only / both unsigned and signed numbers) and the overflow indicators are C = ____ (1/0) for _____ and V = _____ (0/1) for _____ .



Fall 2013 **EE457** **Instructor: Gandhi Puvvada**

Quiz (~ 10%) **Date: 9/27/2013, Friday THH301/201/101**

Name: Calculators are allowed; Closed-book, Closed-notes, No cheat-sheets **Time: 2:00 - 4:40 PM**

Last name	First name

Total points: 213

Student ID: Do NOT write any student ID or SSN **Perfect score: 200 / 213**

1 (54 points) 35 min.

Unsigned numbers and Signed numbers represented in 2's complement system

(8 pts) **1.1** If you perform **A - B** by performing **A + B' + 1** and find the resulting difference to be all 32 zeros, can you conclude that A is equal to B or do you need know if A and B are signed numbers or unsigned numbers or do you need to know the Raw Carry (C32) or the overflow (V)?

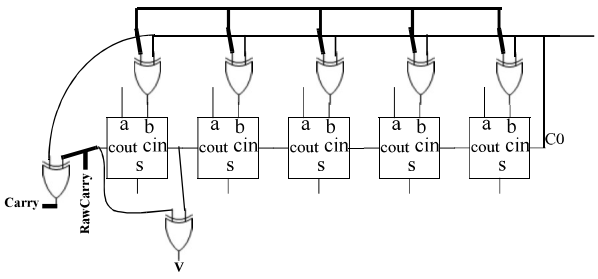
_____

_____

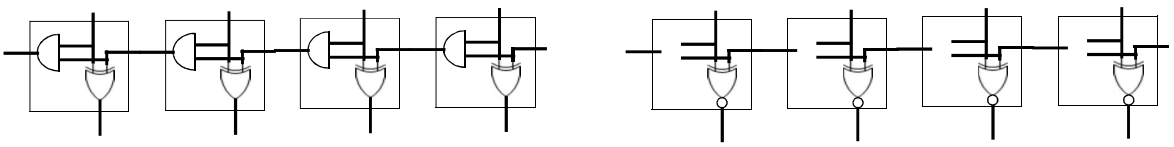
Your lab partner says that if A is equal to B, the raw carry C32 is always a 1 as we will be able to return the money borrowed from bank in our example. You agree or disagree? _____
Your lab partner says that if A is equal to B, the overflow V is always a 0 as A + B' then will necessarily produce all 32 1's and if you add a 1 to it, both C31 and C32 turnout to be a 1 and hence V comes out as zero. You agree or disagree? _____

1.2 Here, we use a 5-bit adder/subtracter built using 5 full-adders to design an incrementer/decrementer. Given below is an incomplete design. complete it and label the control pin as **$\overline{\texttt{INC}}$/DEC** or **INC/$\overline{\texttt{DEC}}$**. This incrementer/decrementer is good for _____ (signed only / unsigned only / both unsigned and signed numbers) and the overflow indicators are C = ____ (1/0) for _____ and V = _____ (0/1) for _____ .

(6 pts)



(12 pts) Given below on the left is a student's incomplete design of a 4-bit incrementer (just incrementer only) using 4 half-adders. Simplify the first half-adder and also derive the unsigned overflow C (C = 1 => overflow) and the signed overflow V (V = 1 => overflow). Similarly complete the design of a decrementer on the right, simplifying if possible and derive overflows.



Incrementer Decrementer

SOLUTION

Fall 2013 | **EE457** | **Instructor: Gandhi Puvvada**

Quiz (~ 10%) | **Date: 9/27/2013, Friday THH301/201/101**

Name: Calculators are allowed; Closed-book, Closed-notes, No cheat-sheets | **Time: 2:00 - 4:40 PM**

Last name	First name

Total points: 213

Student ID: Do NOT write any student ID or SSN | **Perfect score: 200 / 213**

1 (54 points) 35 min.

Unsigned numbers and Signed numbers represented in 2's complement system

(8 pts) **1.1** If you perform **A - B** by performing **A + B' + 1** and find the resulting difference to be all 32 zeros, can you conclude that A is equal to B or do you need know if A and B are signed numbers or unsigned numbers or do you need to know the Raw Carry (C32) or the overflow (V)?

We can conclude that A is equal to B. Equality is same for signed or unsigned. There can not be any overflow when you do X-X.

Your lab partner says that if A is equal to B, the raw carry C32 is always a 1 as we will be able to return the money borrowed from bank in our example. You agree or disagree? Agree

Your lab partner says that if A is equal to B, the overflow V is always a 0 as A + B' then will necessarily produce all 32 1's and if you add a 1 to it, both C31 and C32 turnout to be a 1 and hence V comes out as zero. You agree or disagree? agree

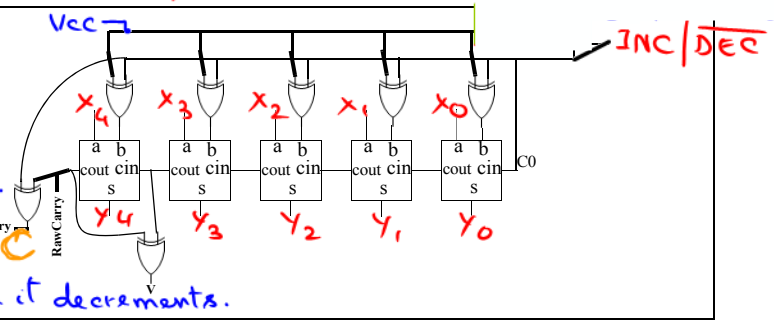
1.2 Here, we use a 5-bit adder/subtracter built using 5 full-adders to design an incrementer/decrementer. Given below is an incomplete design. Label the control pin as **INC/DEC** or **INC/DEC**.

Note: We are flipping when we are incrementing

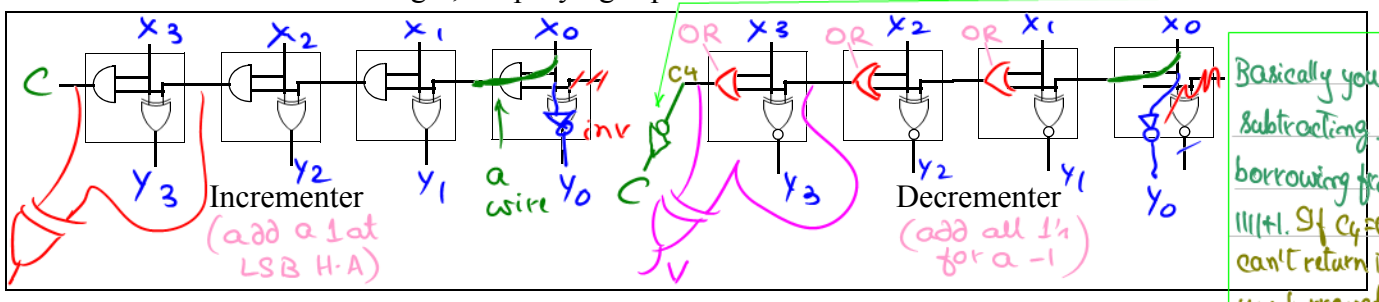
This incrementer/decrementer is good for both signed and unsigned numbers (signed only / unsigned only / both unsigned and signed numbers) and the overflow indicators are C = 0 (1/0) for unsigned and V = 1 (0/1) for signed.

(6 pts)

The B-inputs are tied to Vcc. So when the control INC/DEC = 1 all B' inputs are Zeros and $C_0 = 1$. So it increments. When INC/DEC = 0 the $C_0 = 0$ and the B' inputs are all 1's. All 1's is a -1 and it decrements.



(12 pts) Given below on the left is a student's incomplete design of a 4-bit incrementer (just incrementer only) using 4 half-adders. Simplify the first half-adder and also derive the unsigned overflow C (C = 1 => overflow) and the signed overflow V (V = 1 => overflow). Similarly complete the design of a decrementer on the right, simplifying if possible and derive overflows.

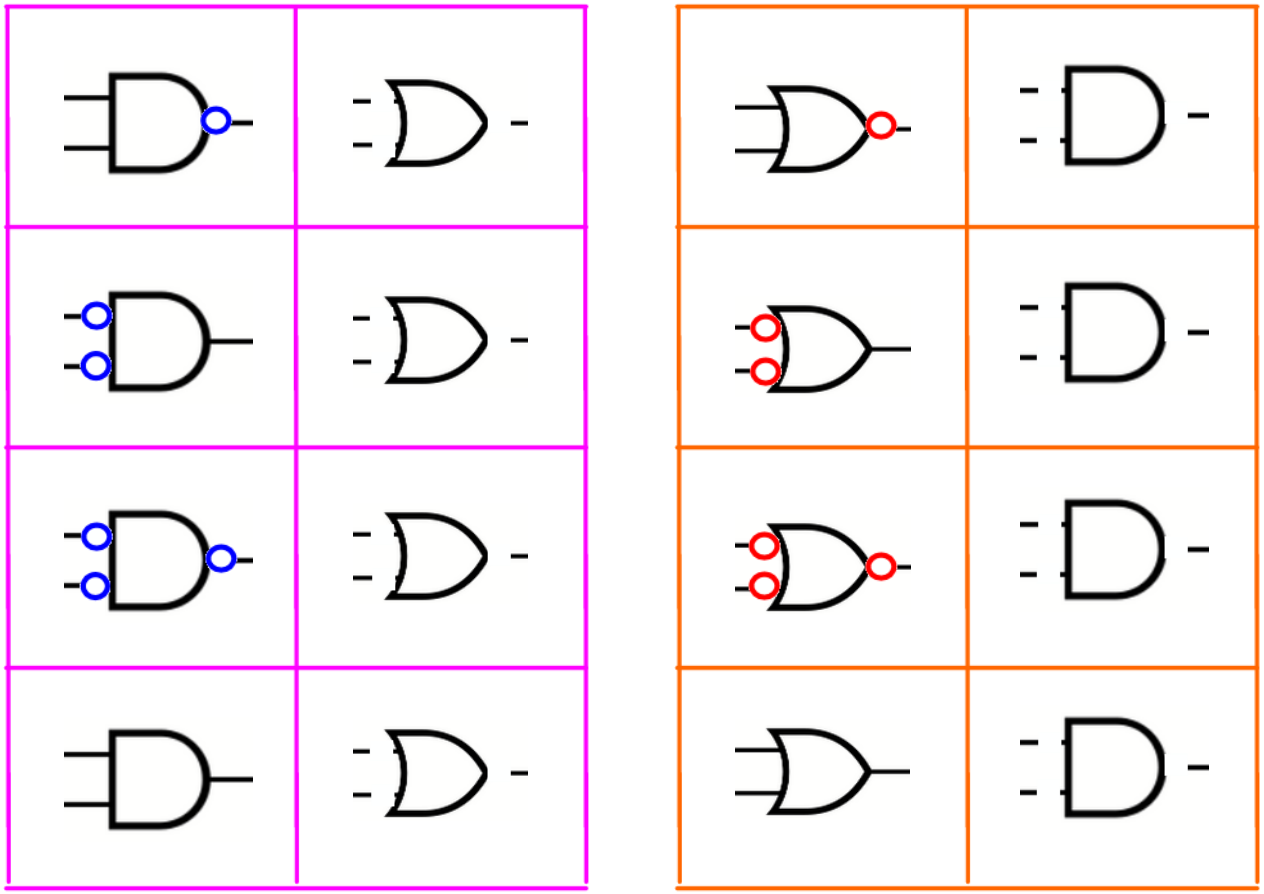


Basically you are subtracting 1 by borrowing from bank 1111+1. If $C_4 = 0$, you can't return the money you borrowed => hence Overflow in unsigned decrementer

2.1 XOR and XNOR operations. Making a 6-bit adder using two 4-bit adders

Exercise

By De Morgan's theorem,
we know the following
equivalences. Complete them!

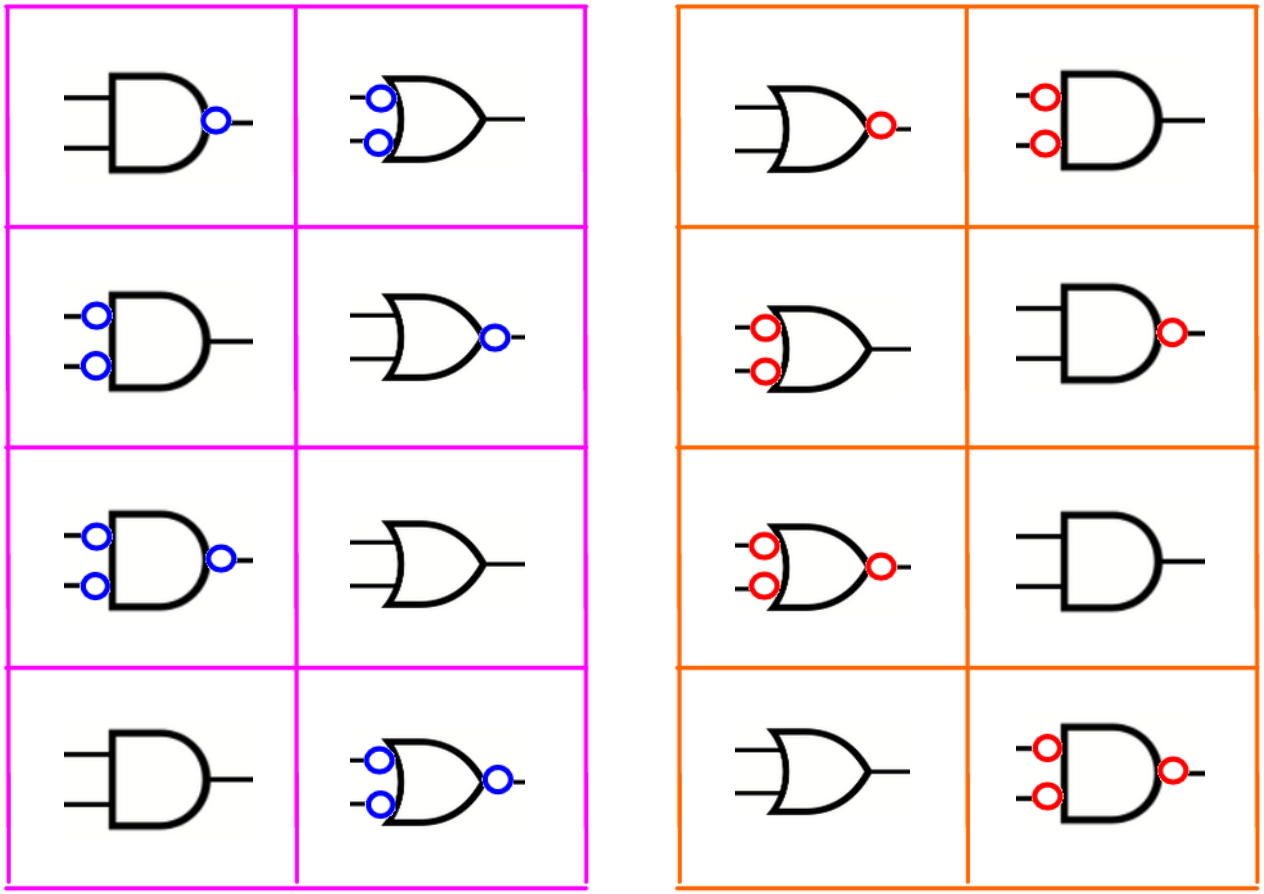


$$X \cdot Y = \overline{\overline{X} \cdot \overline{Y}} = X \quad Y$$

$$A + B = \overline{\overline{A} + \overline{B}} = A \quad B$$

Solution

By De Morgan's theorem, we know the following equivalences. Complete them!



$$X \bullet Y = \overline{\overline{X \bullet Y}} = \overline{X} + \overline{Y}$$

$$A + B = \overline{\overline{A + B}} = \overline{A} \bullet \overline{B}$$

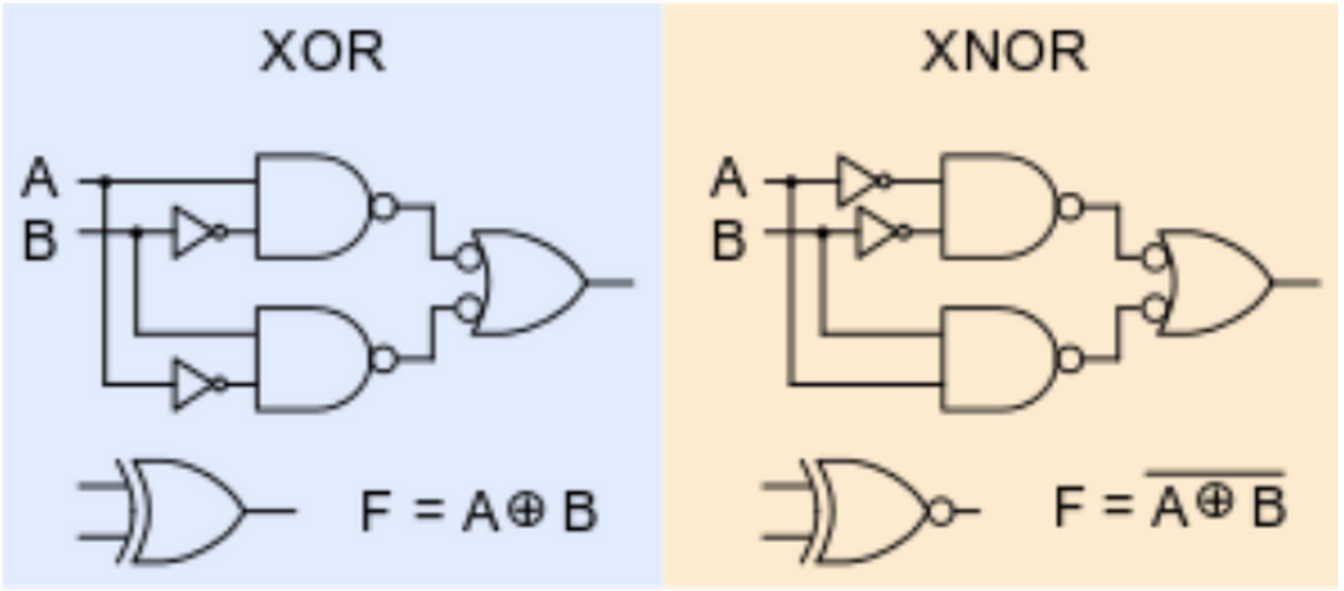
Information

You know XOR and XNOR gates and their Truth Tables.

https://digilent.com/reference/learn/fundamentals/digital-logic/xor-and-xnor/start

$$F_{SOP} = A \cdot \overline{B} + \overline{A} \cdot B$$

$$F_{SOP} = \overline{A} \cdot \overline{B} + A \cdot B$$



A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

A	B	F
0	0	1
0	1	0
1	0	0
1	1	1

Exercise

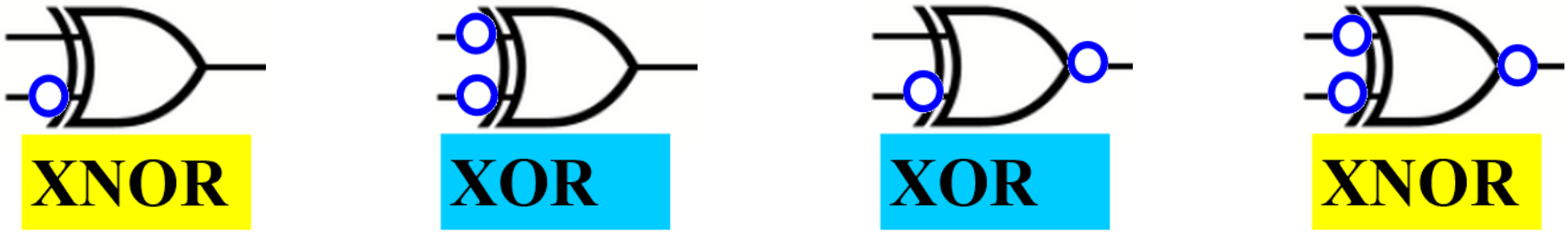
Identify all XOR gates and all XNOR gates below.
If any of them is neither, cross it off.

In the case of AND/NAND/OR/NOR, we know how to mask (block) or pass a variable X input using a Control C input.

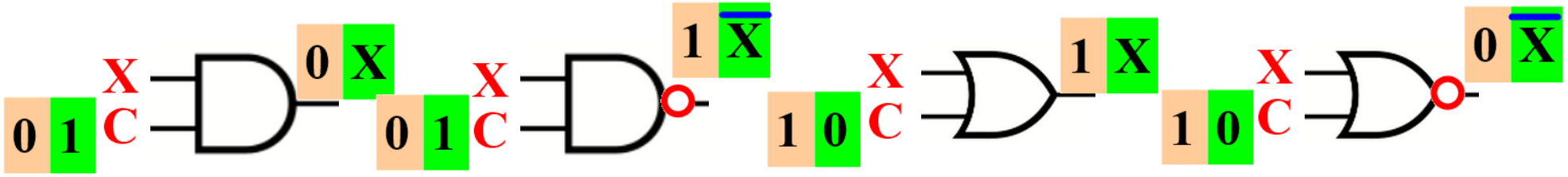
Similarly, in the case of XOR/XNOR, ...

Solution

Identify all XOR gates and all XNOR gates below. If any of them is neither, cross it off.



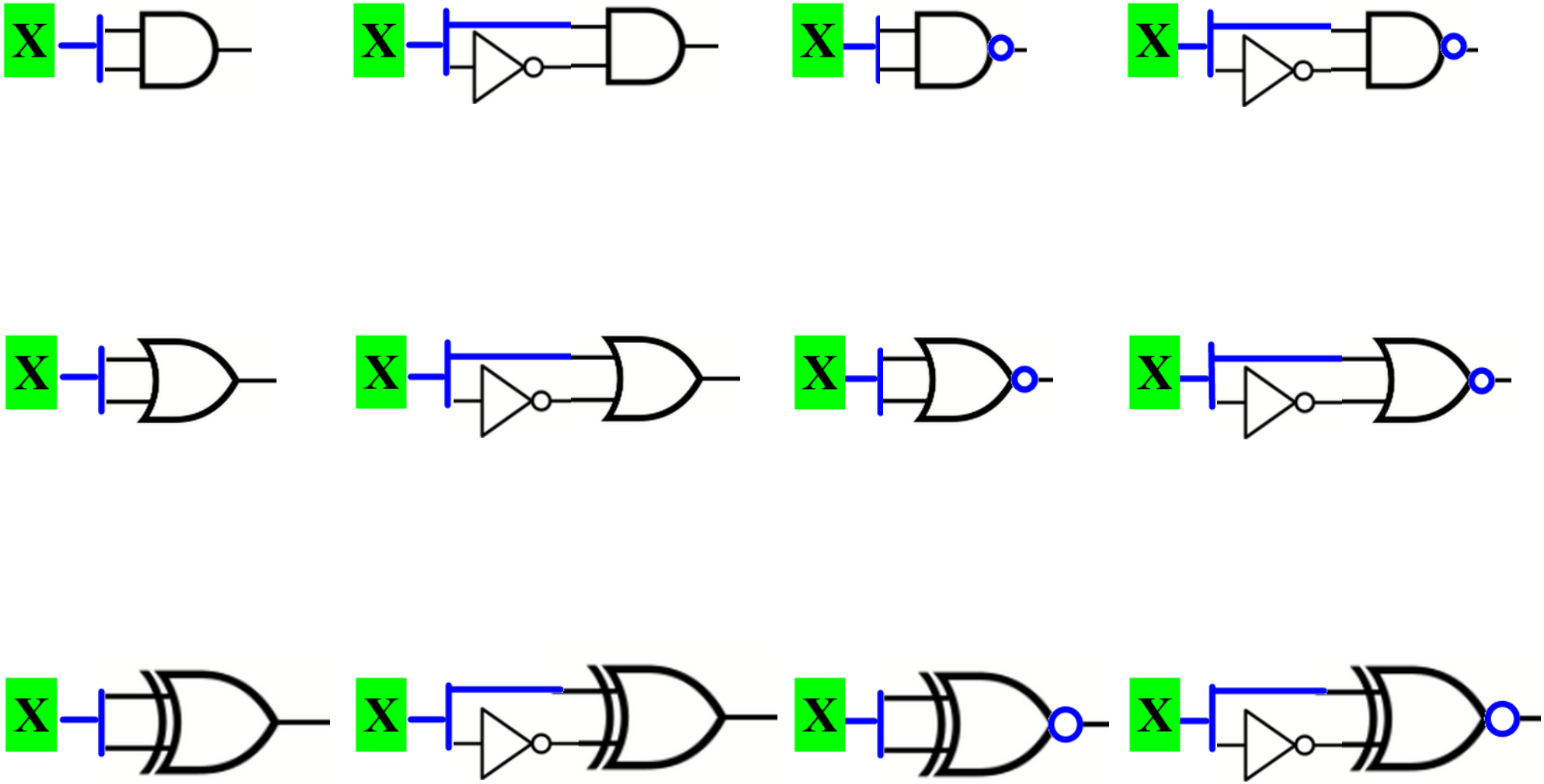
In the case of AND/NAND/OR/NOR, we know how to mask (block) or pass a variable X input using a Control C input.



Similarly, in the case of XOR/XNOR, ...

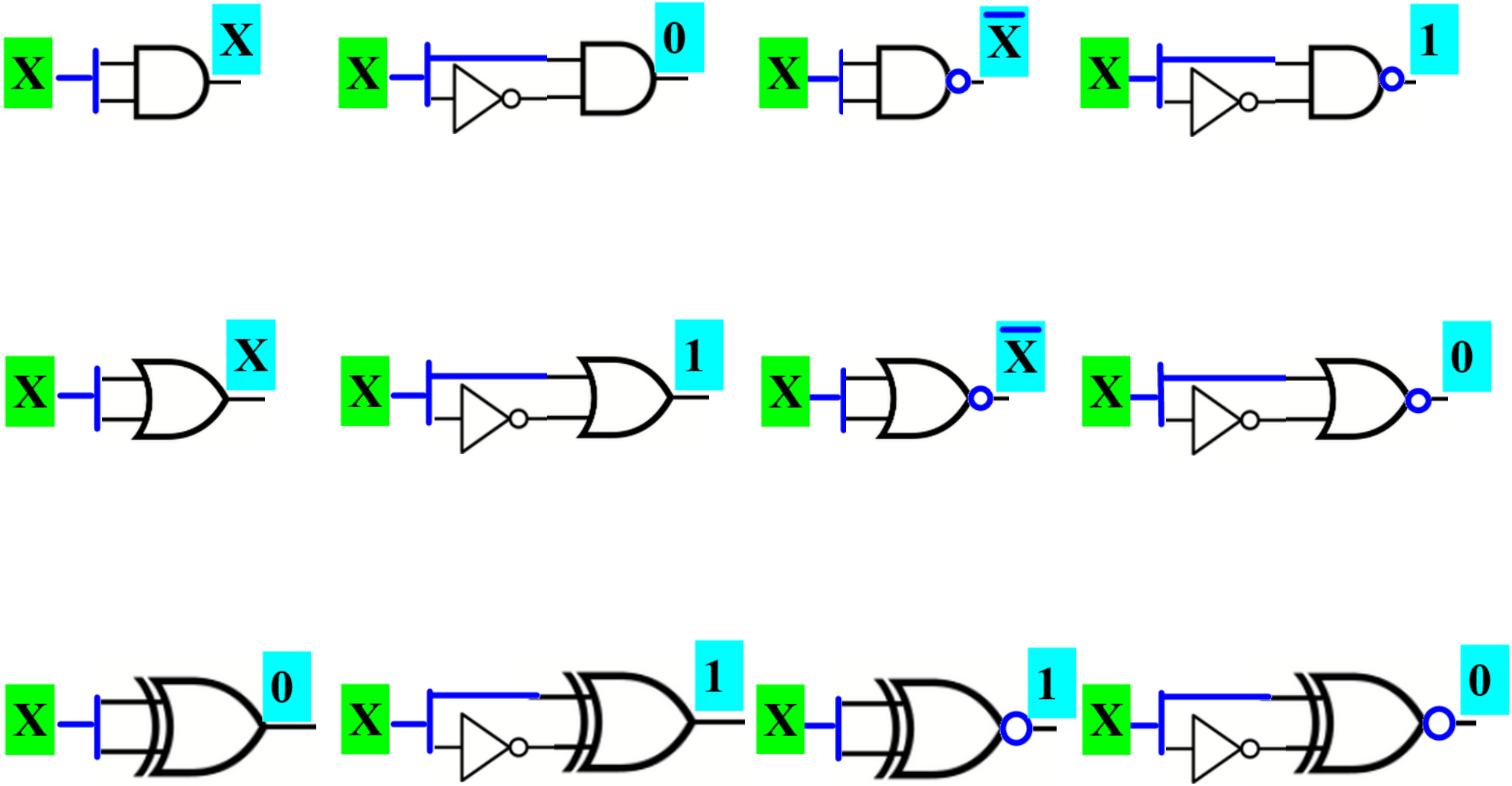
Exercise

What happens if we connect the variable X to both inputs as shown? Is the output a constant or a function of X



Solution

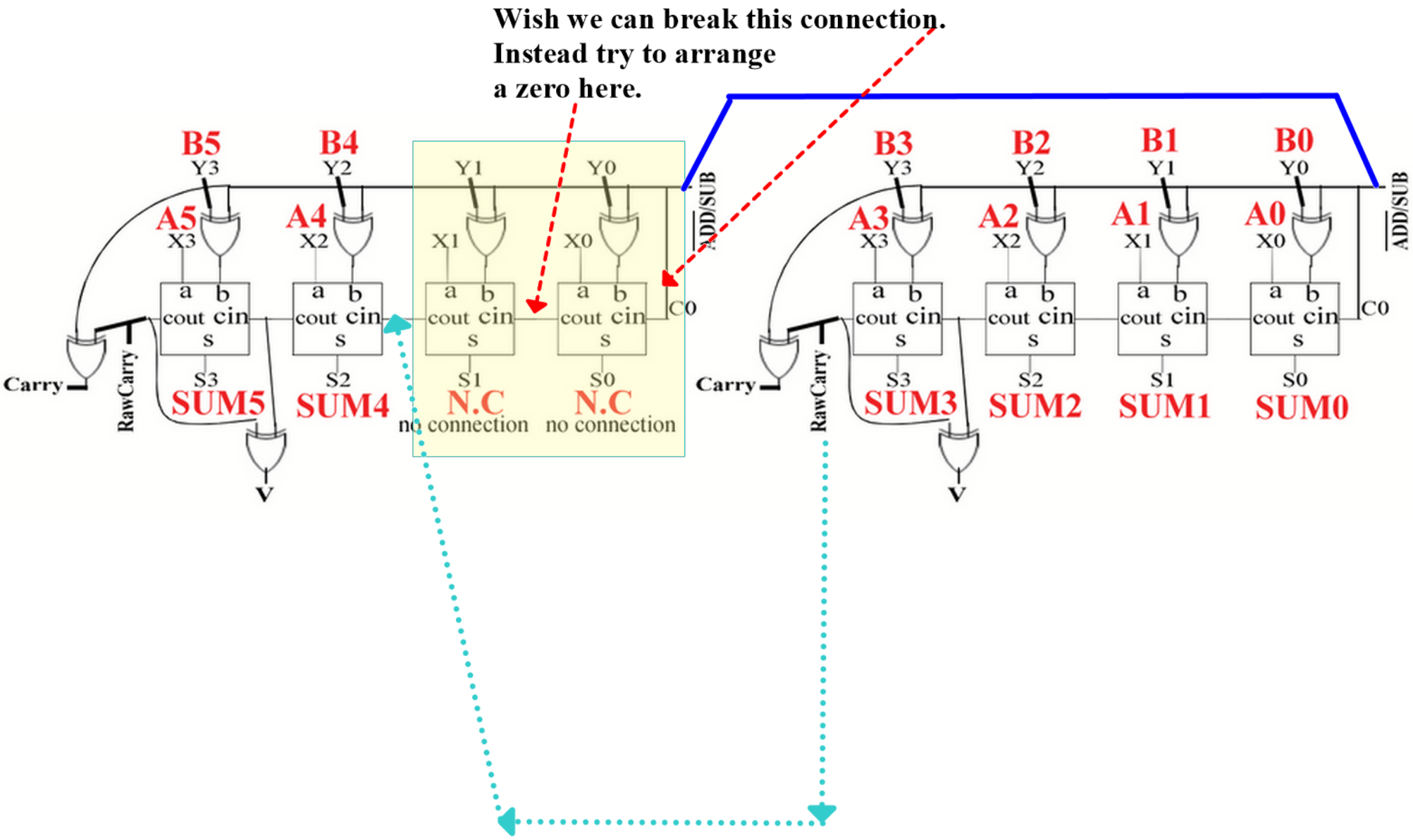
What happens if we connect the variable X to both inputs as shown? Is the output a constant or a function of X



Exercise

Spring 2011 Quiz Q#1.3

Build a 6-bit Adder/Subtractor, using two 4-bit adder/ subtractors. You do not have access to internal nodes of the given 4-bit items. Complete the design below.

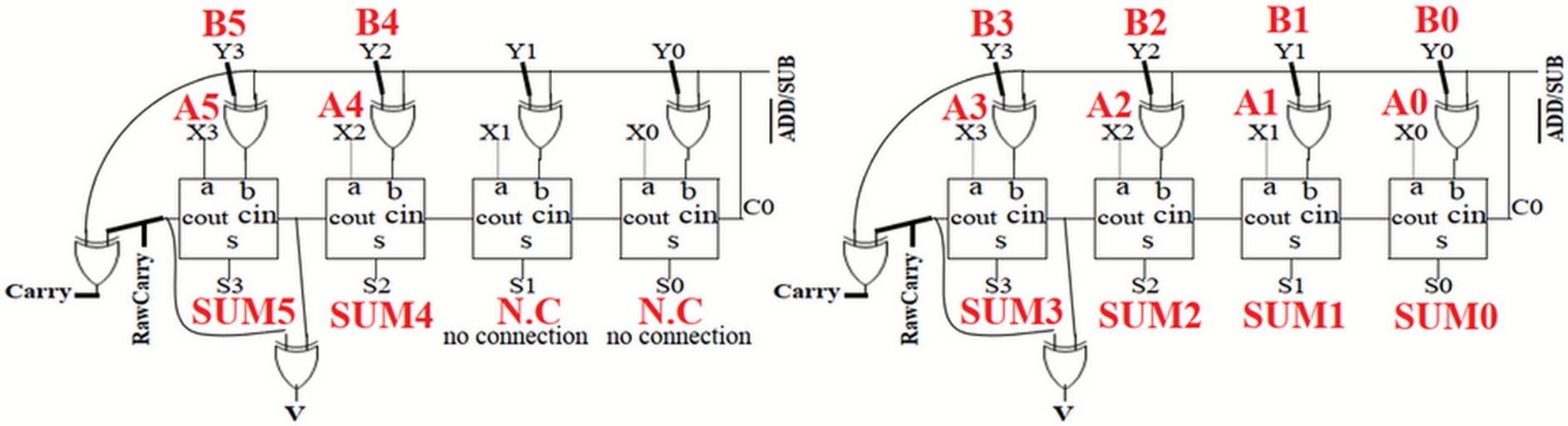


Exercise

Spring 2011 Quiz Q#1.3

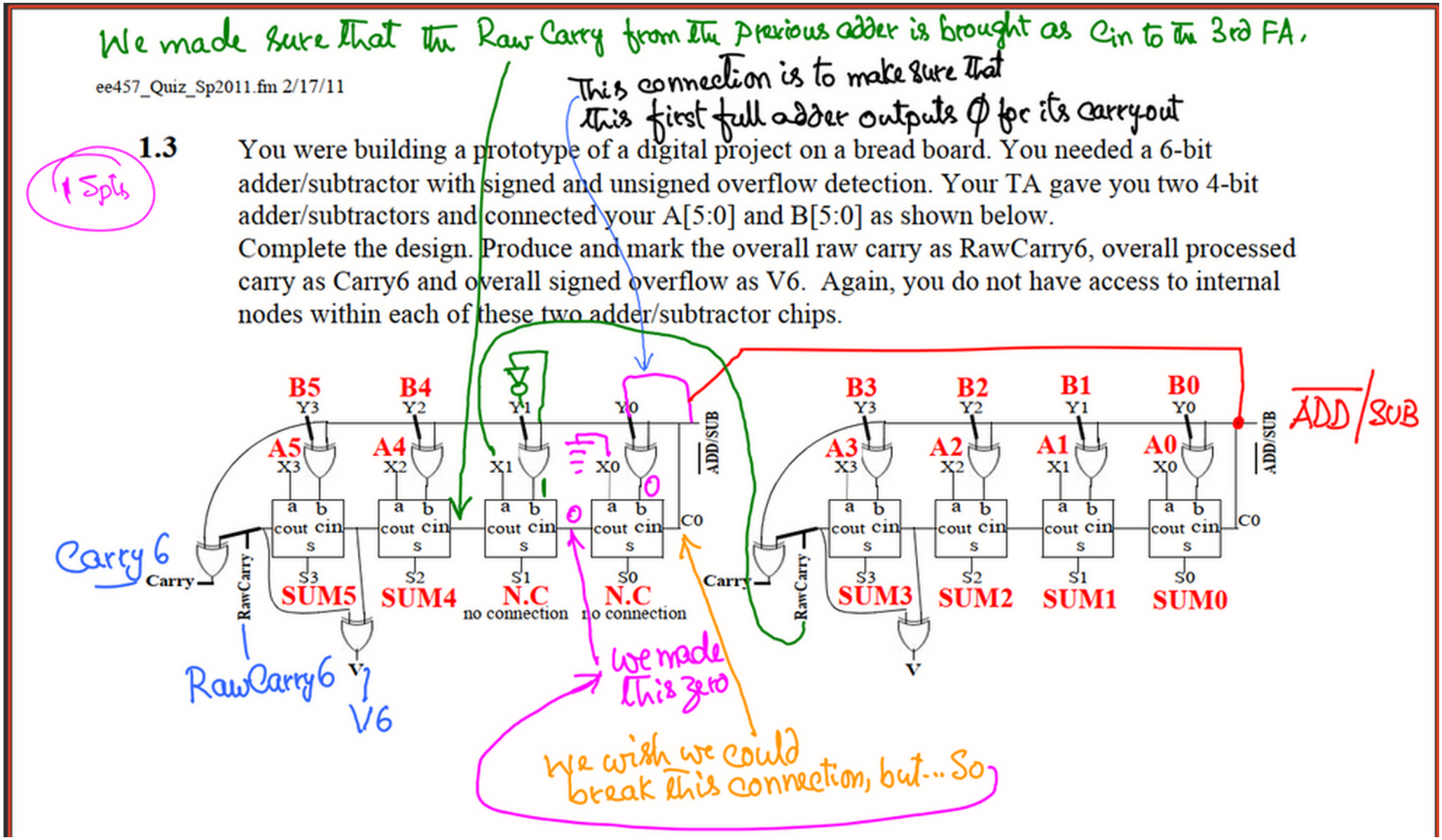
ee457_Quiz_Sp2011.fm 2/18/11

(15 pts) **1.3** You were building a prototype of a digital project on a bread board. You needed a 6-bit adder/subtractor with signed and unsigned overflow detection. Your TA gave you two 4-bit adder/subtractors and connected your A[5:0] and B[5:0] as shown below.
Complete the design. Produce and mark the overall raw carry as RawCarry6, overall processed carry as Carry6 and overall signed overflow as V6. Again, you do not have access to internal nodes within each of these two adder/subtractor chips.

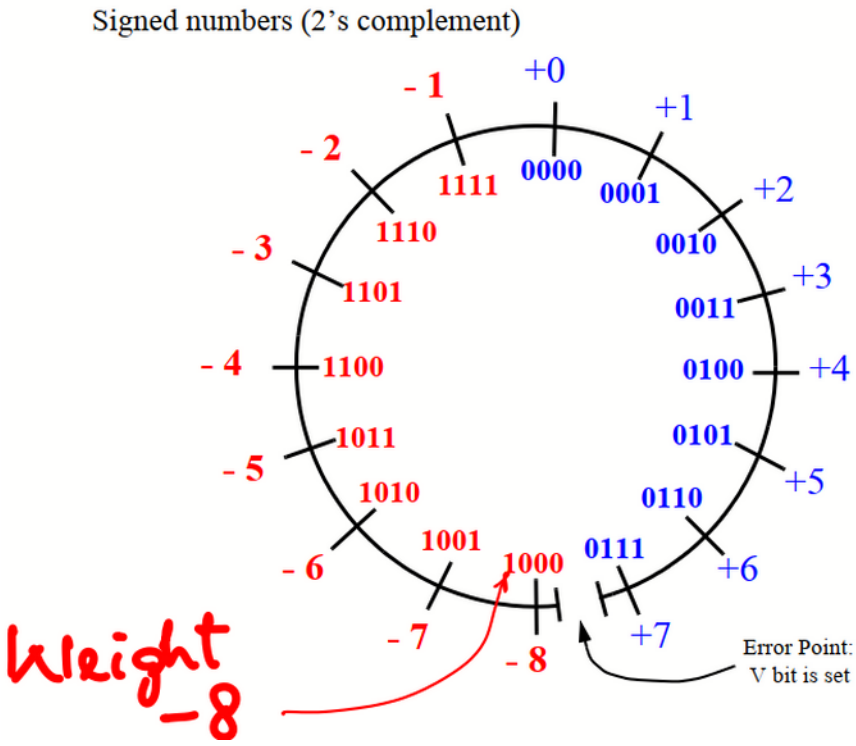


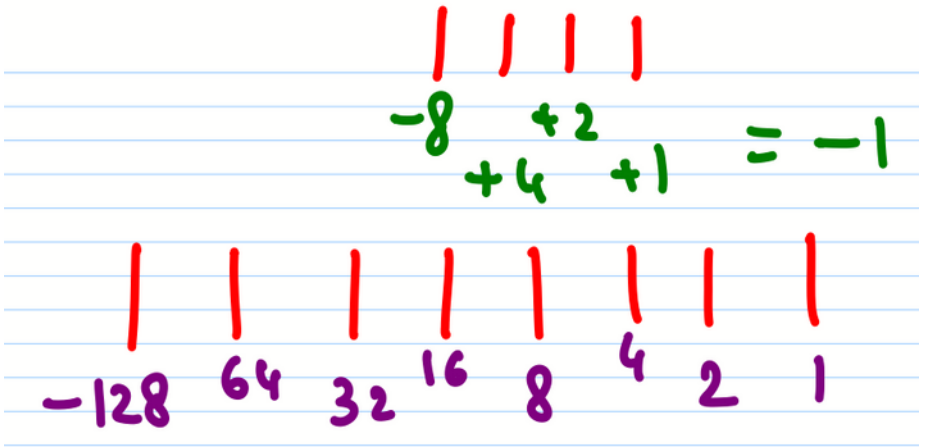
Solution

Spring 2011 Quiz Q#1.3



3. Positional weights -- sign extension -- Multiply by 2 or 4 or any power of 2, Division by 2 or 4 or any power of 2 and the integer quotient





Exercise

Spring 2011 Quiz Q#1.1

1.1 The two signed numbers P, Q (8-bit P and 6-bit Q, represented in 2's complement system) are equal, if P[5:0] is identical to Q[5:0] (i.e. lower six bits of the 8-bit P are identical to the six bits of the 6-bit Q) and further (select one)

(i) (P[7:6] == 2'b00)
(ii) (P[7:6] == 2'b11)
(iii) (P[7:6] == {P[5],P[5]}) // R.H.S is a 2-bit item obtained by concatenating P[5] to P[5] itself.
(iv) none of the above

Exercise

Fall 2009 Quiz Q#2.4

2.4 Compare these two 32-bit signed numbers (in 2's complement system). The 16 X's and the 16 Y's can be replaced by any bit combinations. Example the 16 X's can be 16 ones and the 16 Y's can be 16 zeros. A is _____ (greater than / less than / can't be compared with) B.

```
A:  1111_0000_1010_XXXX_XXXX_1111_XXXX_XXXX
B:  1111_0000_1011_YYYY_YYYY_0000_YYYY_YYYY
```

Does your answer to the above question for signed numbers applicable for unsigned binary numbers also???

Solution

Spring 2011 Quiz Q#1.1

1.1 The two signed numbers P, Q (8-bit P and 6-bit Q, represented in 2's complement system) are equal, if P[5:0] is identical to Q[5:0] (i.e. lower six bits of the 8-bit P are identical to the six bits of the 6-bit Q) and further (select one)

(i) (P[7:6] == 2'b00)

(ii) (P[7:6] == 2'b11)

(iii) (P[7:6] == {P[5],P[5]}) // R.H.S is a 2-bit item obtained by concatenating P[5] to P[5] itself.

(iv) none of the above

Solution

Fall 2009 Quiz Q#2.4

2.4 Compare these two 32-bit signed numbers (in 2's complement system). The 16 X's and the 16 Y's can be replaced by any bit combinations. Example the 16 X's can be 16 ones and the 16 Y's can be 16 zeros. A is _____ (greater than / less than / can't be compared with) B.

```
A:   1111_0000_1010_XXXX_XXXX_1111_XXXX_XXXX
B:   1111_0000_1011_YYYY_YYYY_0000_YYYY_YYYY
```

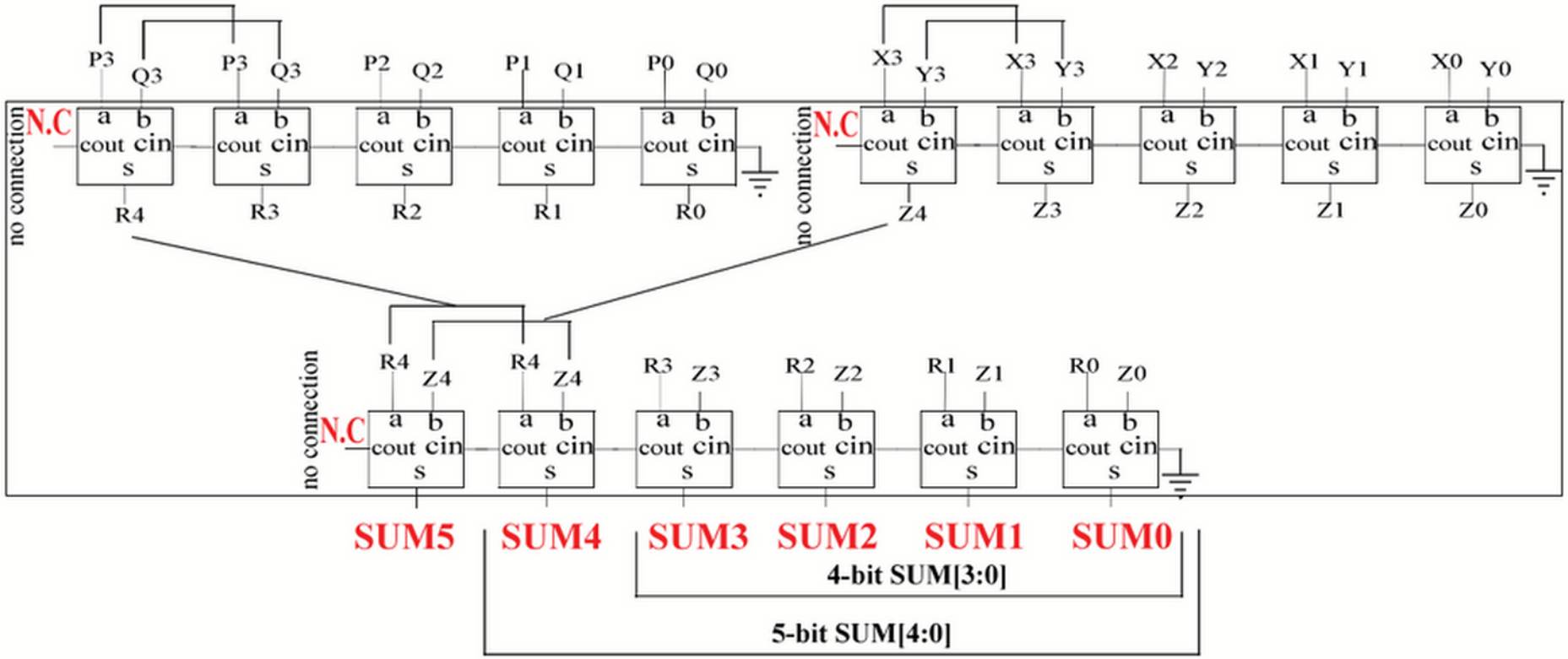
Does your answer to the above question for signed numbers applicable for unsigned binary numbers also???

Yes, for both signed and unsigned numbers, the lower 31 bits of the above 32 bit numbers bear positive weights!

Exercise

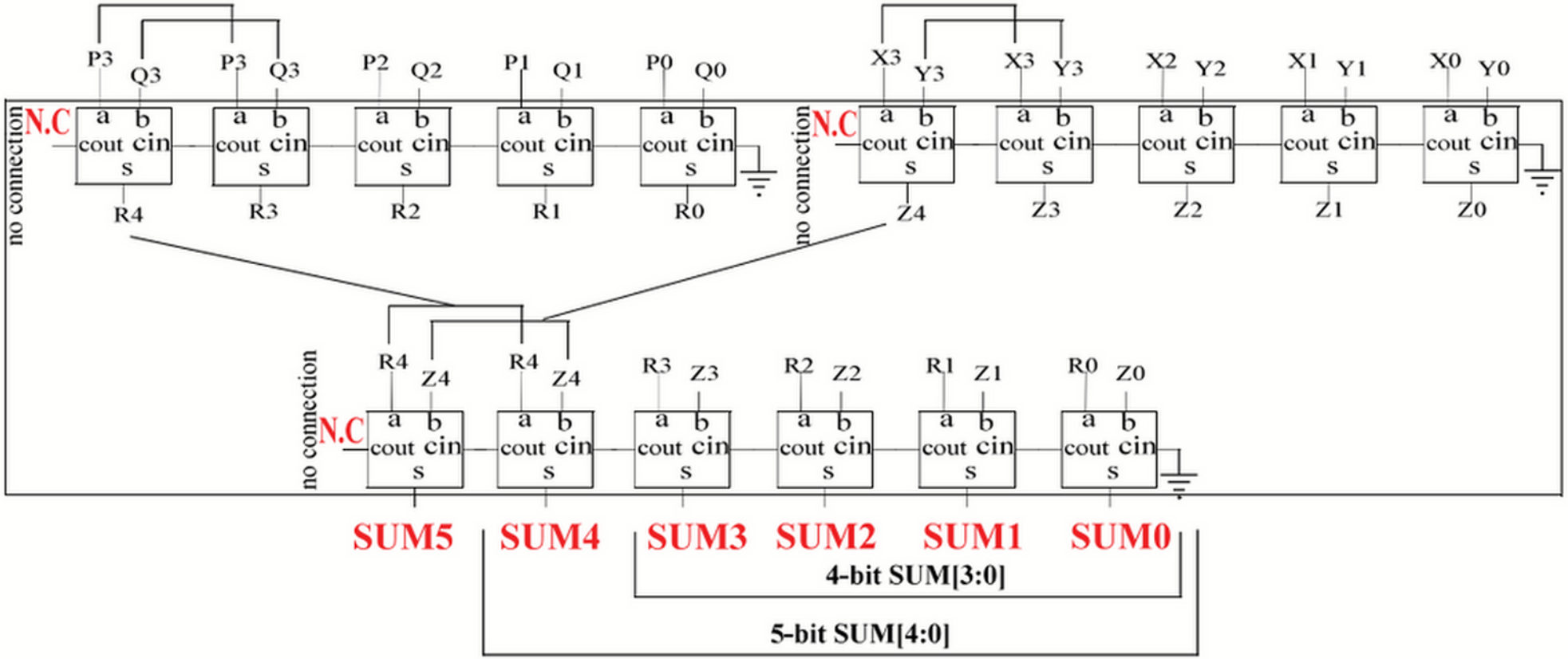
Spring 2011 Quiz Q#1.2

(15 pts) **1.2** Four 4-bit signed numbers P[3:0], Q[3:0], X[3:0], Y[3:0] are added using a tree of 3 adders (two 5-bit adders followed by a 6-bit adder) as shown below. These adders do not have any overflow detection logic. The 4-bit items are sign-extended to 5 bits and the two pairs are added using the two 5-bit adders to produce 5-bit intermediate sums, R[4:0] and Z[4:0]. These two are further added using the 6-bit adder to produce SUM[5:0]. Internal nodes of these three adders are not accessible. Produce two overflow outputs called V5, V4. The condition (V5 == 1) indicates that the 5-bit sum SUM[4:0] could not represent the sum of P, Q, X, Y. Similarly, the condition (V4 == 1) indicates that the 4-bit sum SUM[3:0] could not represent the sum of P, Q, X, Y.
Note: P,Q,R,S can be any mix of positive and negative numbers.



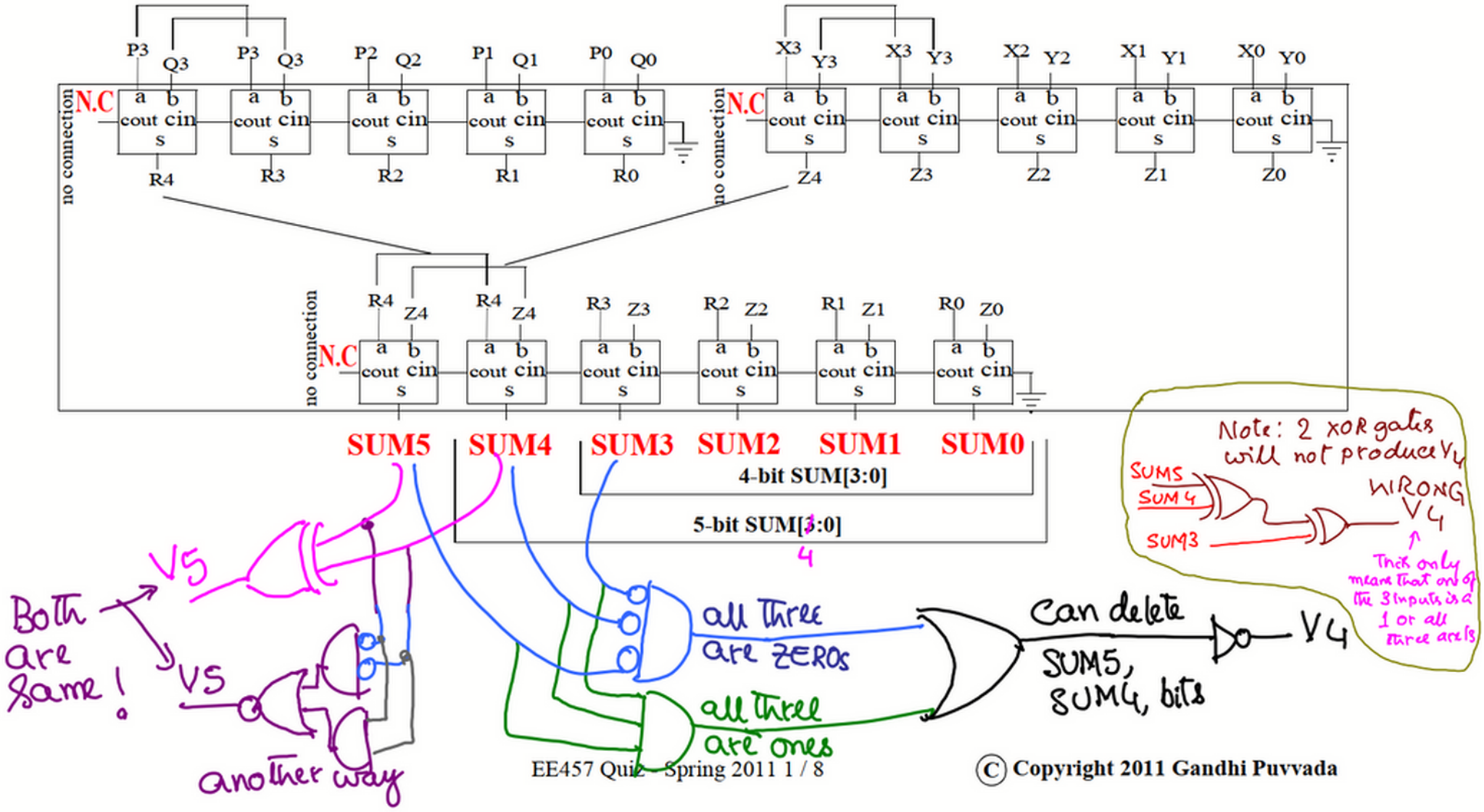
Exercise

Spring 2011 Quiz Q#1.2



Solution

Spring 2011 Quiz Q#1.2



4. ALU Lab -- SLT -- SLTU --
Comparison unit design using a subtractor

SLT

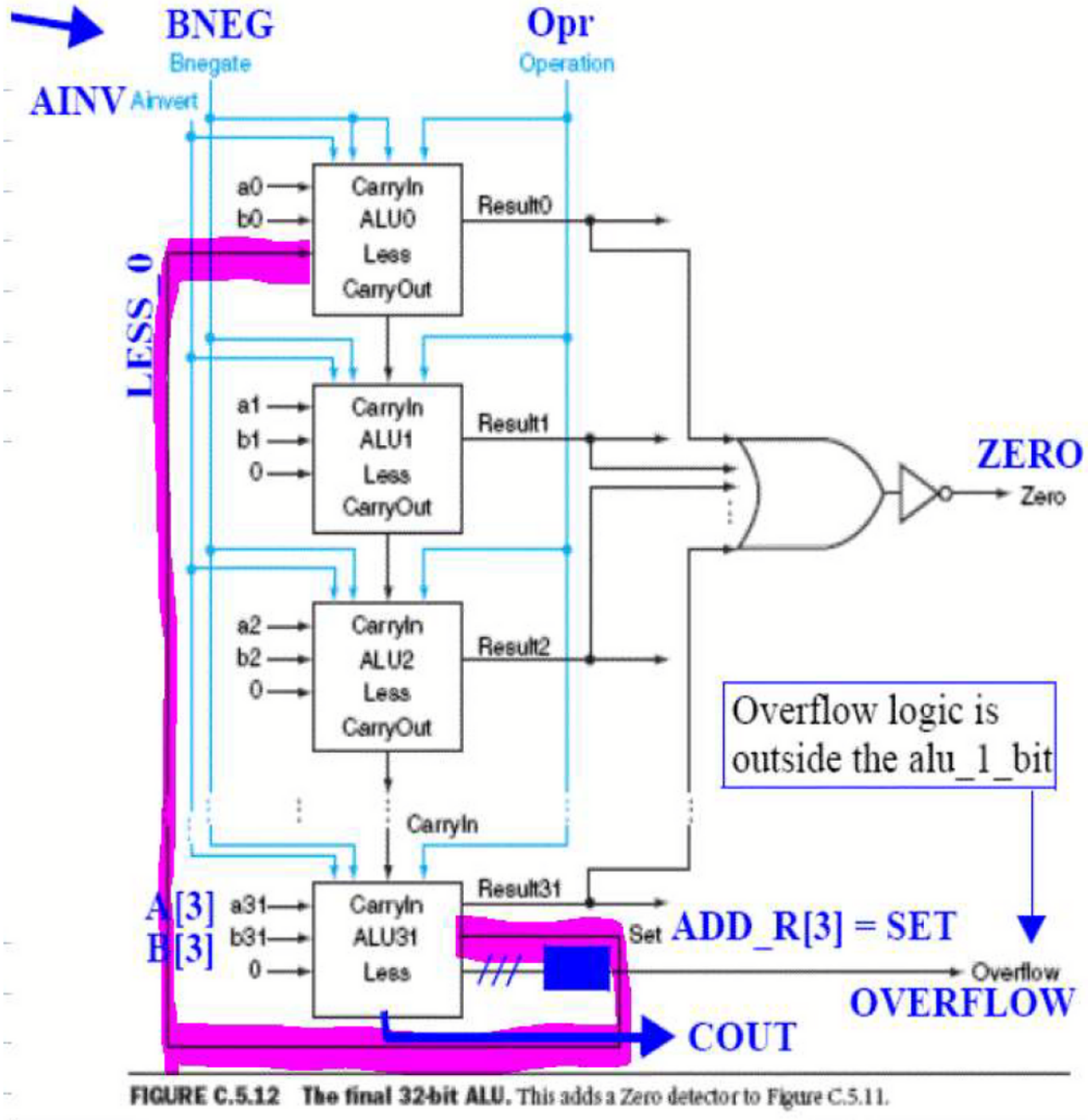
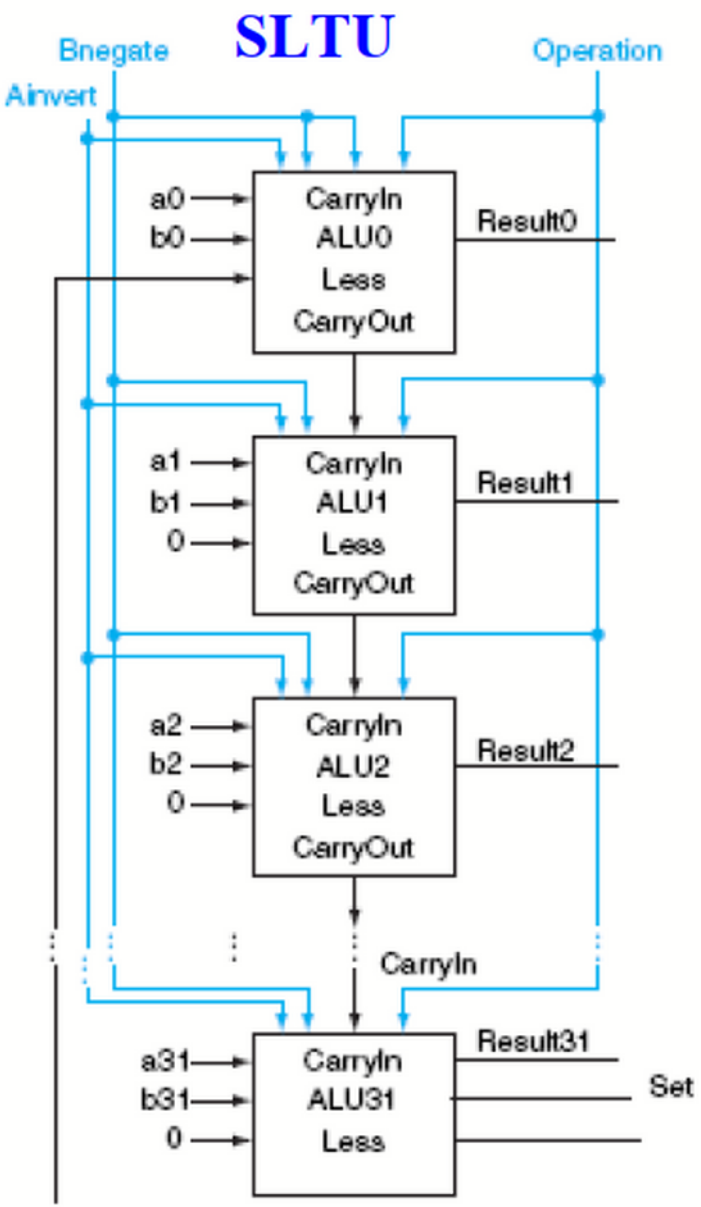
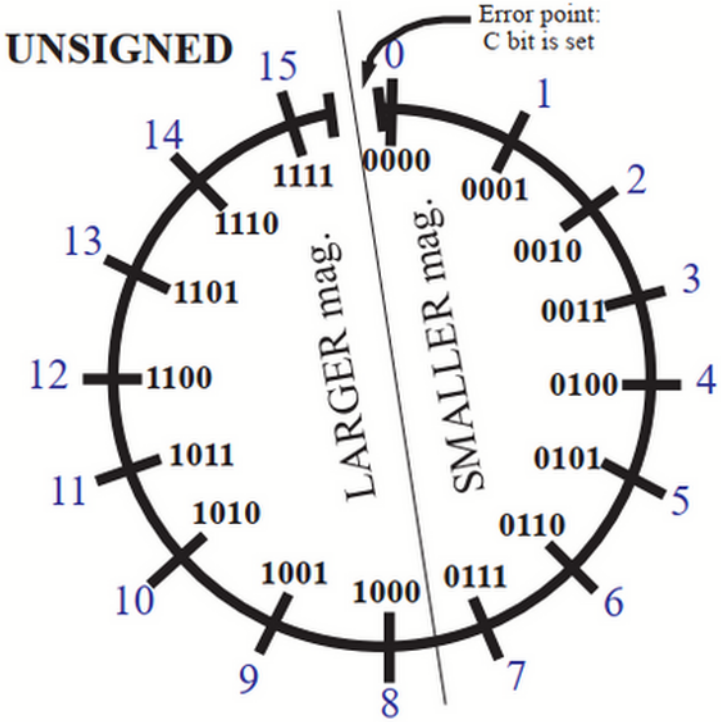


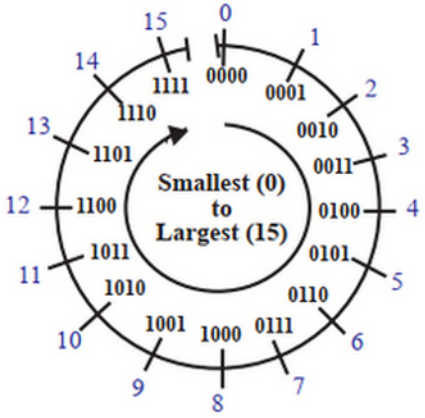
FIGURE C.5.12 **The final 32-bit ALU.** This adds a Zero detector to Figure C.5.11.

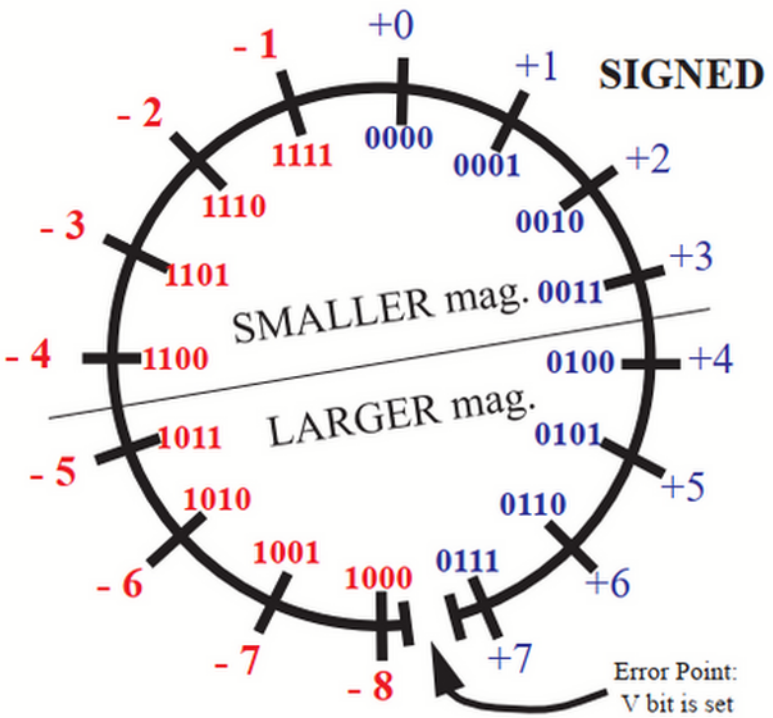
SLTU

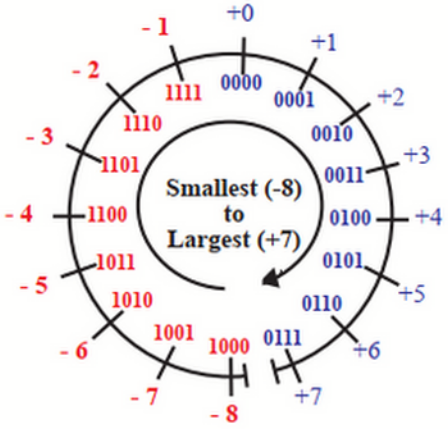

SLTU
Bnegate
Ainvert
Operation
a0
b0
CarryIn
ALU0
Less
CarryOut
Result0
a1
b1
0
CarryIn
ALU1
Less
CarryOut
Result1
a2
b2
0
CarryIn
ALU2
Less
CarryOut
Result2
CarryIn
a31
b31
0
CarryIn
ALU31
Less
Result31
Set

Selecting 4-bit numbers





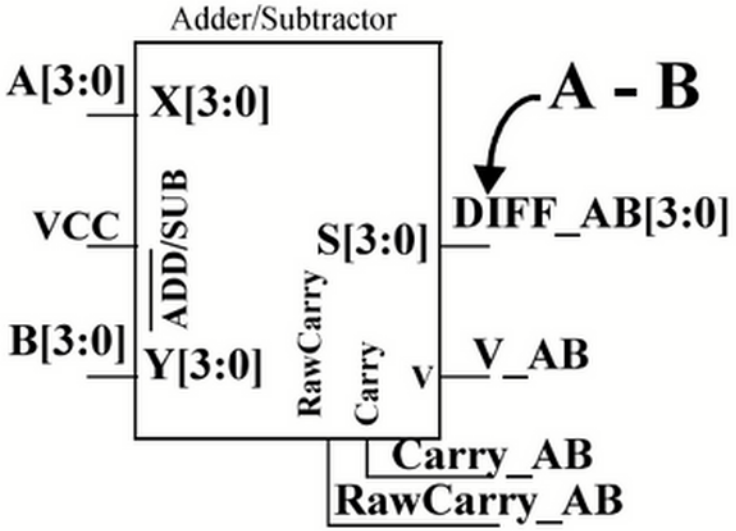


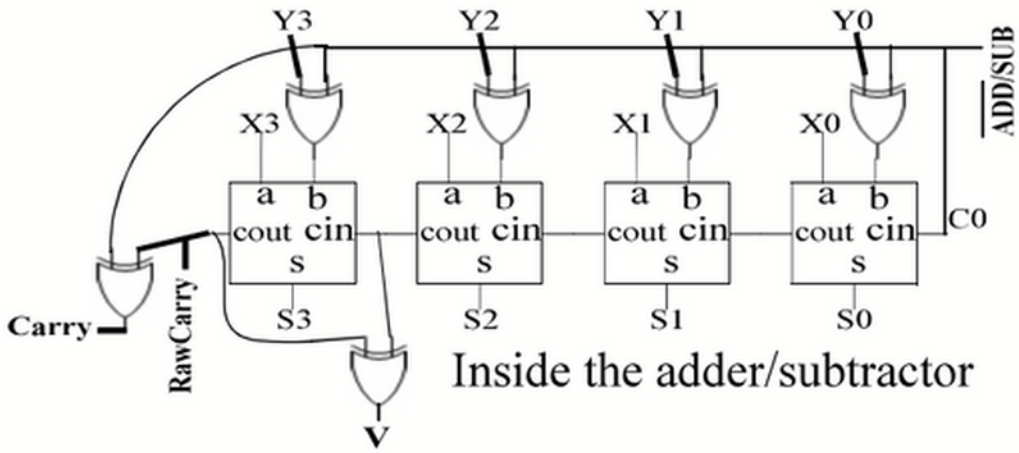


Exercise

Spring 2009 Quiz Q#1.1

1.1 Signed and unsigned numbers: Use the subtractor below and produce
(a) **`A_EQ_B`** (A ***equal*** B),
(b) **`A_GT_B`** (A ***greater than*** B treating A and B as ***signed*** numbers), and
(c) **`A_HI_B`** (A ***higher than*** B treating A and B as ***unsigned*** numbers).





Exercise continuation

Spring 2009 Quiz Q#1.1

4 pts

A_EQ_B

6 pts

A_GT_B

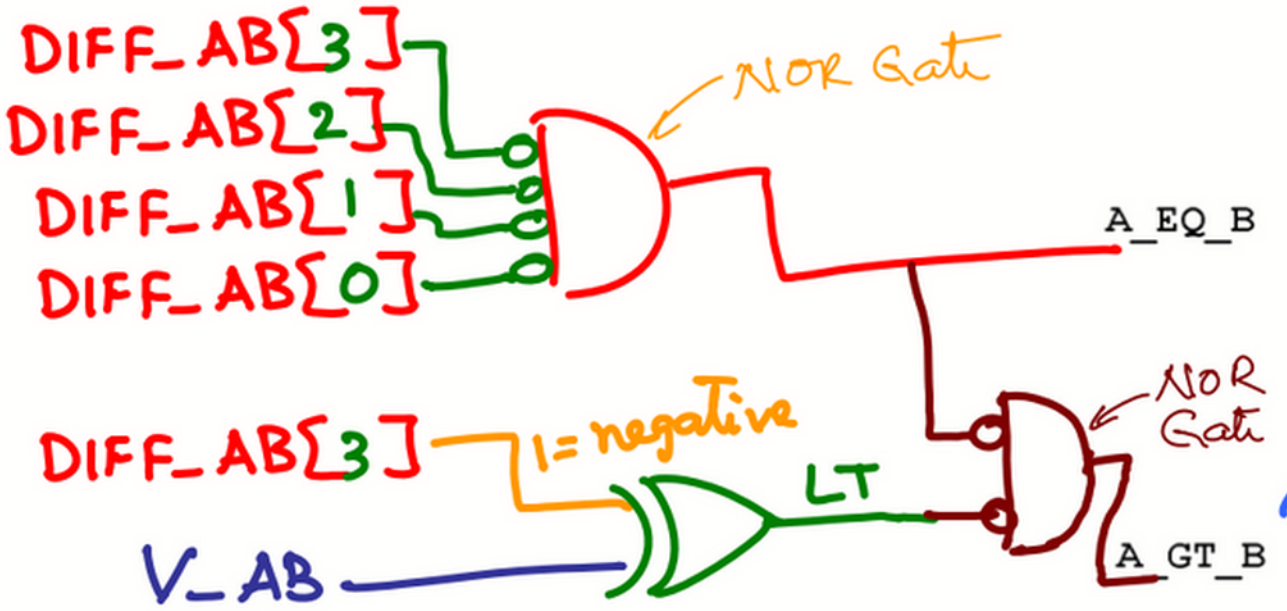
6 pts

A_HI_B

Solution

Spring 2009 Quiz Q#1.1

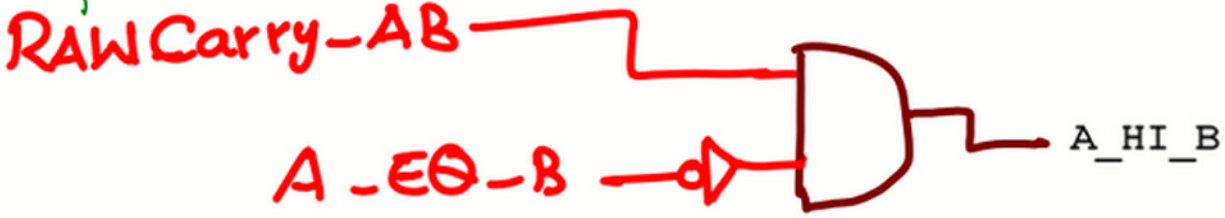
4 pts



6 pts

presence of RawCarry indicates that A is HIGHER or EQUAL to B.

6 pts



Exercise

Fall 2008 Quiz Q#1

1 (8 + 12 + 10.5 = 30.5 points) 20 min.

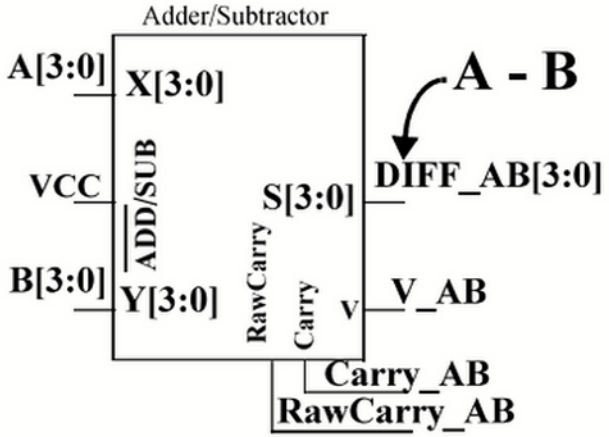
Signed and unsigned numbers, combinational logic design: Find the difference between the given two numbers by subtracting the *smaller* number from the *larger* number treating the numbers (a) as *unsigned* and (b) as *signed.*

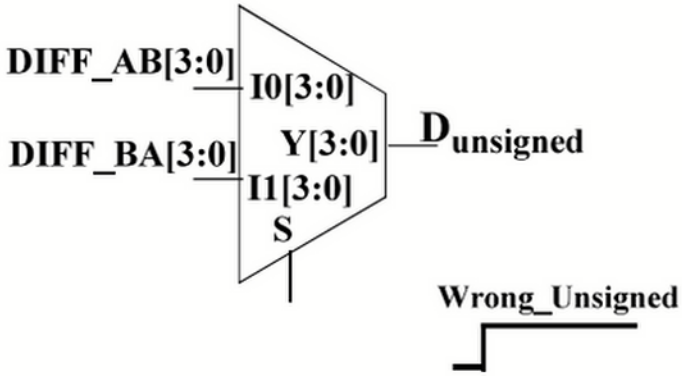
Note: -5 is smaller than -3. So if A = -5 and B = -3, the signed subtraction shall be B - A = +2

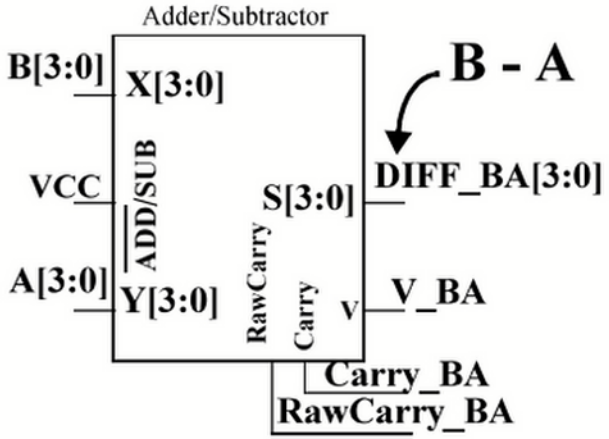
Given below on the left are two subtractors to produce A - B as well as B - A and all other inferences (RawCarry, Carry, and V bits). Complete the **select lines** on the two muxes on the right to select the right difference for $D_{unsigned}$ (treating A and B as unsigned) and D_{signed} (treating A and B as signed). Also produce "Wrong_Unsigned" and "Wrong_Signed" if such subtraction produced wrong difference due to overflow.

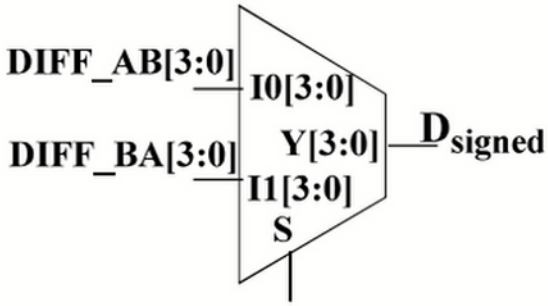
Exercise continuation

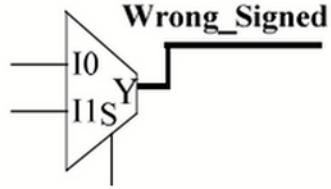
Fall 2008 Quiz Q#1

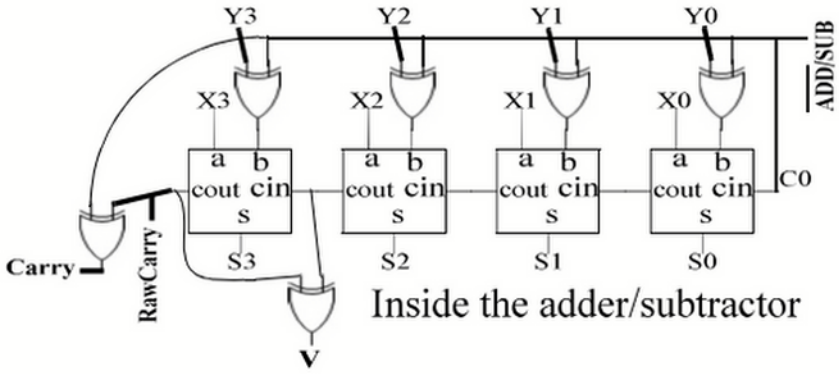








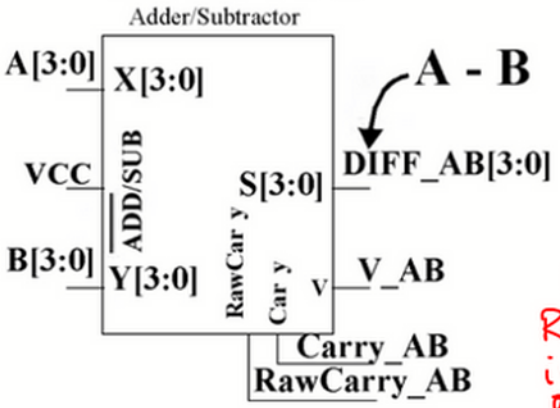


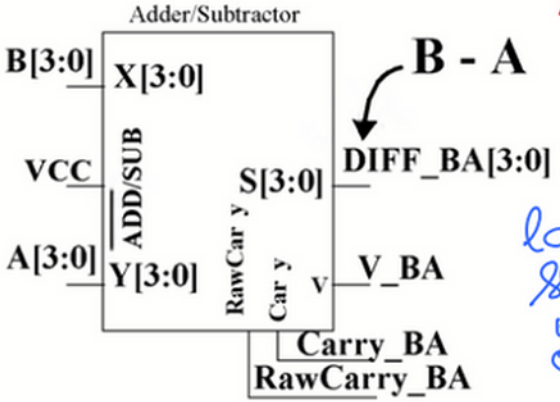


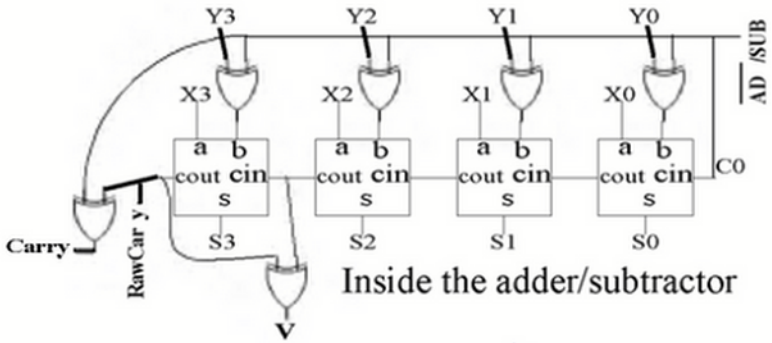
Solution

Fall 2008 Quiz Q#1

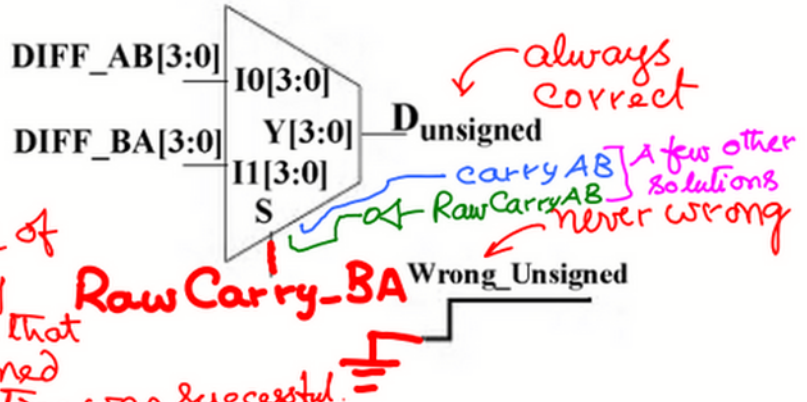
answer questions on page3.

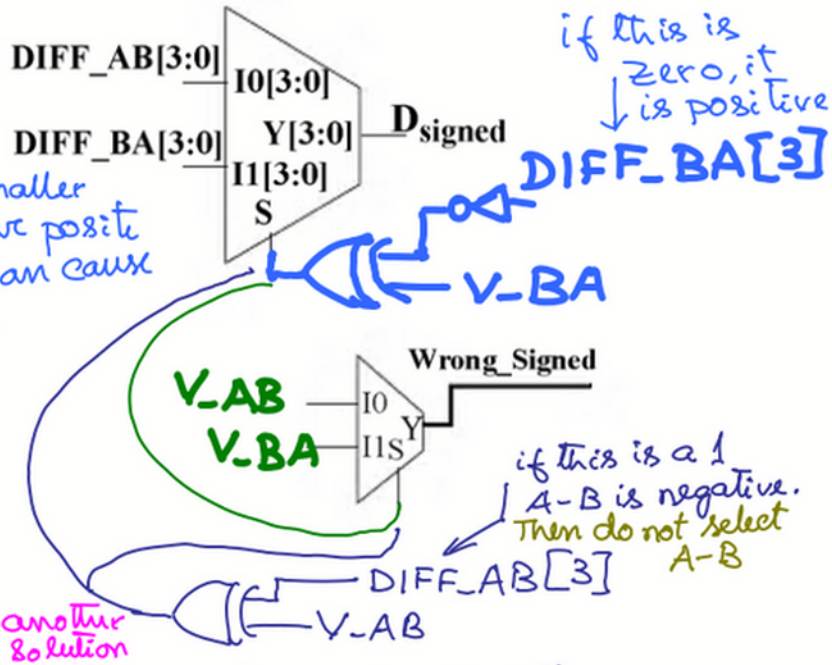




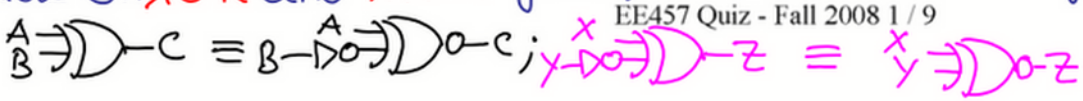


Inside the adder/subtractor





Note on XOR and XNOR gates:



Based on XOR and XNOR gate transformation rules, there are a few more equivalent solutions.

Exercise

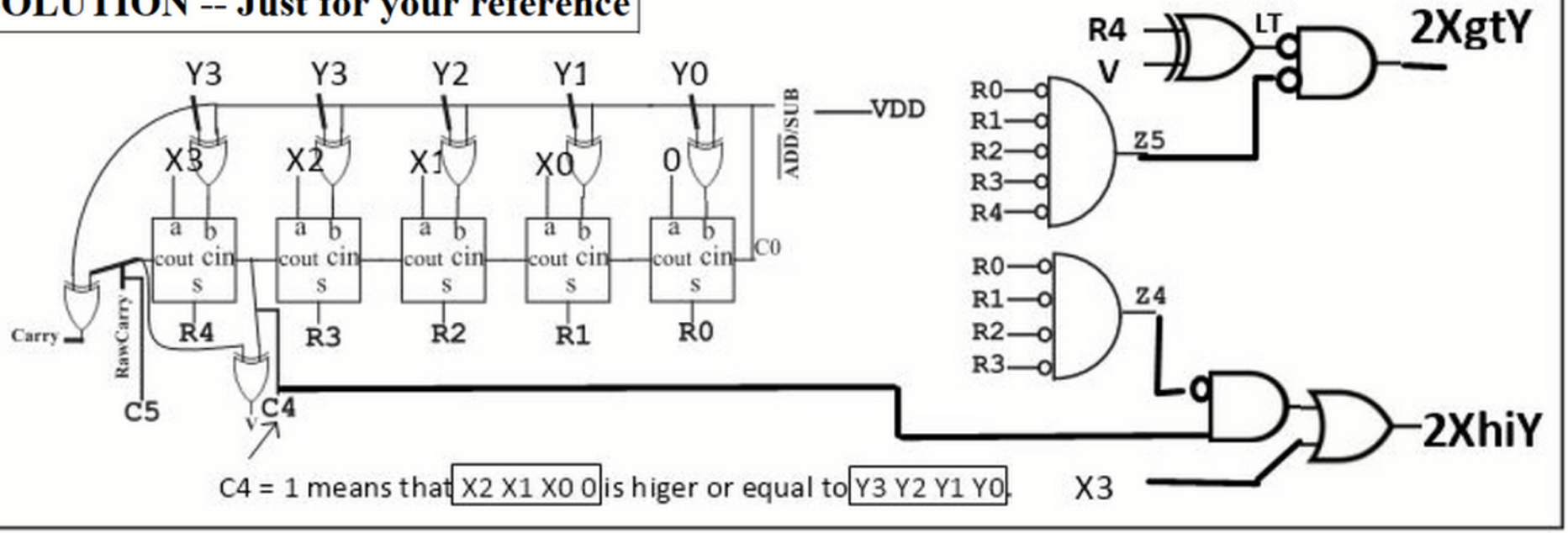
Fall 2018 Quiz Q#2.1

2.1 Given below is the Q#2 (statement of the question and the solution) from Spring 2018 Quiz that you were asked to go through

Q#2 of Spring 2018 Quiz

Given two 4-bit numbers **`X`** (**`X3 X2 X1 X0`**) and **`Y`** (**`Y3 Y2 Y1 Y0`**), we need to produce **`2XgtY`** (**`2X`** is *greater* than **`Y`**, treating **`X`** and **`Y`** as *signed* numbers represented in 2's complement notation) and also **`2XhiY`** (**`2X`** is *higher* than **`Y`**, treating **`X`** and **`Y`** as *unsigned* binary numbers). Use the following 5-bit ripple-carry adder/subtracter as a subtracter. Notice that **`C4`** is also brought out. Is **`2X`** higher, if **`X3`** is a **`1`**? If so, for unsigned comparison, we can manage with the lower 4-bit part of the 5-bit adder/subtracter! Note, that we are trying to produce, "**`gt`**", [not "**`gte`**" (greater or equal)] and it is "**`hi`**", [not "**`his`**" (higher or same)].

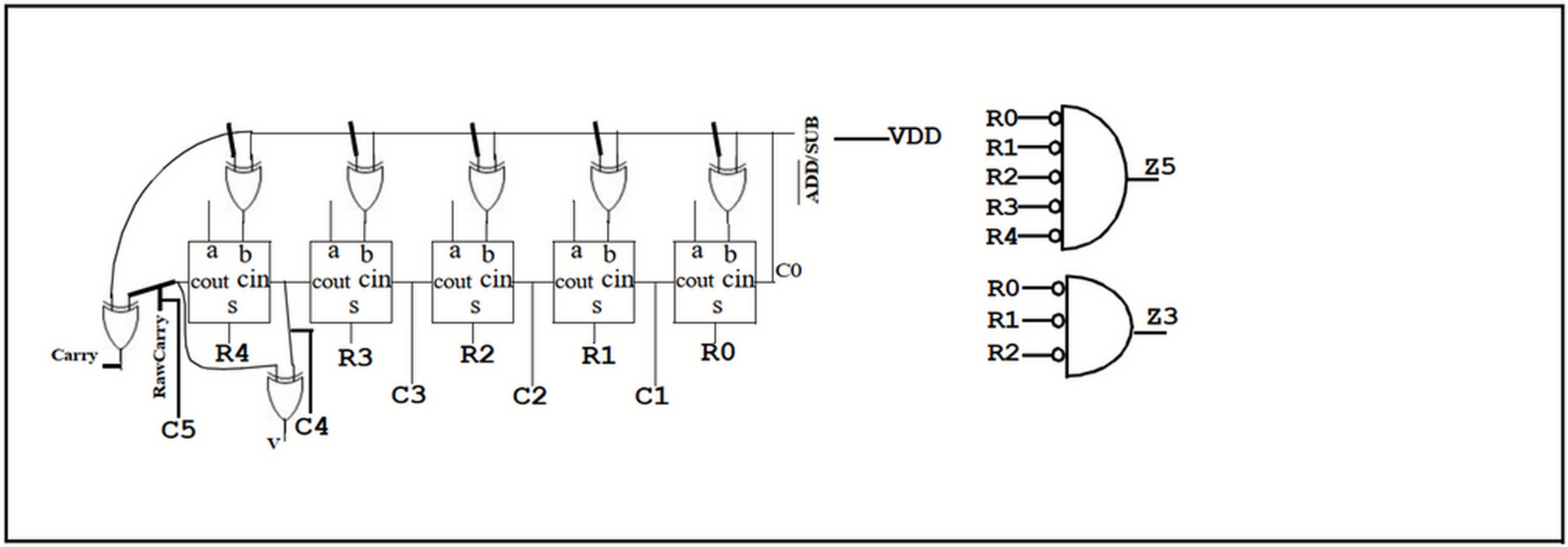
SOLUTION -- Just for your reference



Exercise continuation

Fall 2018 Quiz Q#2.1

2.1.1 Now we need to produce **`4XgtY`** and **`4XhiY`** instead of **`2XgtY`** and **`2XhiY`** as shown above. If we were given a 6-bit subtracter in place of the 5-bit subtracter, it would have been fairly straight forward but you are given the same 5-bit subtracter below.
Perhaps you may be able to ***ignore*** **`Y0`** (or deal with it ***outside*** the subtracter as needed) so that the rest of the bits can be handled by the 5-bit subtracter. Notice that the internal carries **`C3`**, **`C2`**, and **`C1`** are also brought out for your possible use and **`Z3`** is produced instead of the previous **`Z4`**.



Solution

Fall 2018 Quiz Q#2.1

2.1.1 Now we need to produce **`4XgtY`** and **`4XhiY`** instead of **`2XgtY`** and **`2XhiY`** as shown above. If we were given a 6-bit subtracter in place of the 5-bit subtracter, it would have been fairly straight forward but you are given the same 5-bit subtracter below.
Perhaps you may be able to ***ignore*** **`Y0`** (or deal with it ***outside*** the subtracter as needed) so that the rest of the bits can be handled by the 5-bit subtracter. Notice that the internal carries **`C3`**, **`C2`**, and **`C1`** are also brought out for your possible use and **`Z3`** is produced instead of the previous **`Z4`**.