CS2100

TUTORIAL #7

COMBINATIONAL CIRCUITS

(PREPARED BY: AARON TAN)

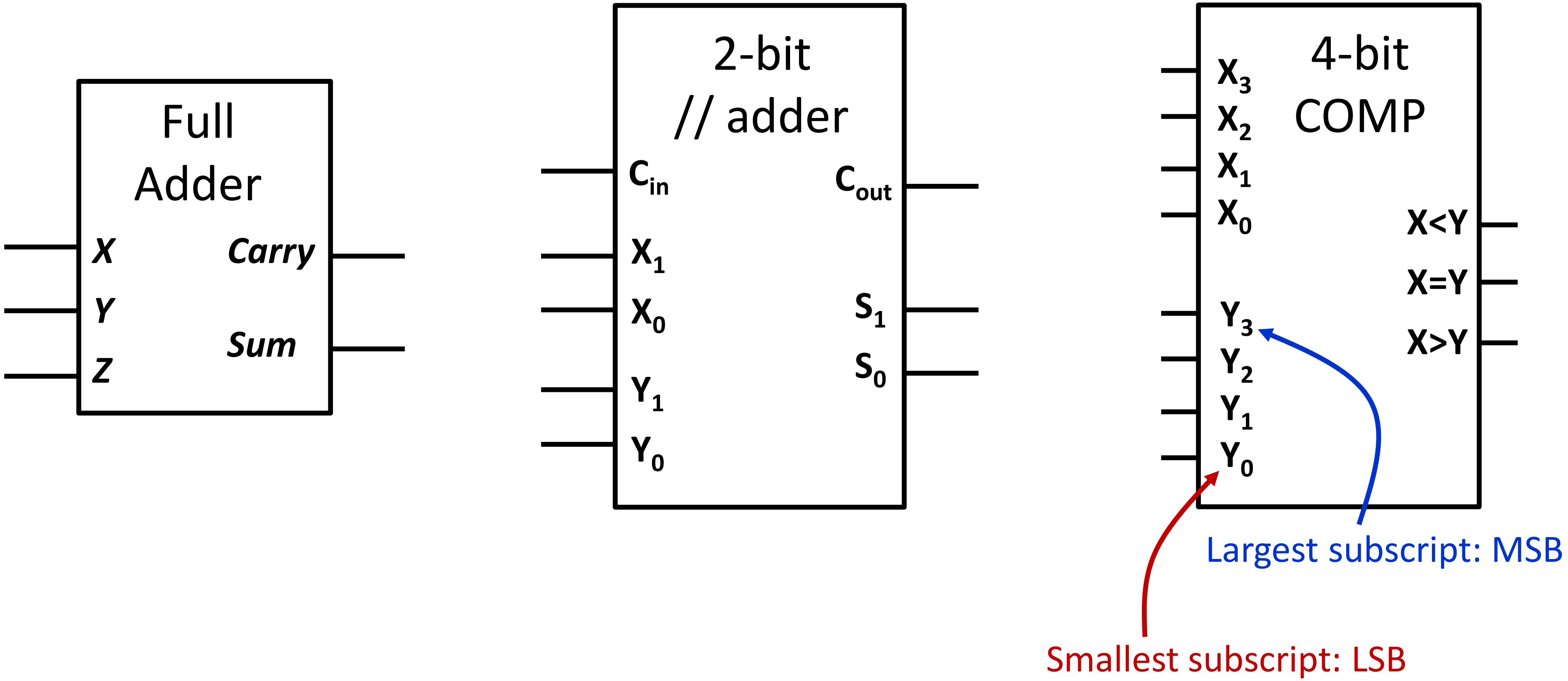
D3. A combinational circuit takes in a 5-bit input *ABCDE* and generates a 2-bit value *PQ* such that *PQ* represents the *distance* between the two closest 1s in the input. The distance is defined to be the number of 0s between the two closest 1s.

You may assume that the distance is always determinable from the given input. Therefore, inputs such as 00000 and 01000 will not be supplied to this circuit.

Q1. You are to design a circuit to implement a function *V*(*A*,*B*,*C*,*D*,*E*) that takes in input *ABCDE* and generates output 1 if *ABCDE* is a valid input for the circuit in question D3 above, or 0 if *ABCDE* is an invalid input.

Q1.

Idea?


Full
Adder
X
Y
Z
Carry
Sum
2-bit
// adder
C_{in}
C_{out}
X_1
X_0
S_1
S_0
Y_1
Y_0
4-bit
COMP
X_3
X_2
X_1
X_0
$X<Y$
$X=Y$
$X>Y$
Y_3
Y_2
Y_1
Y_0
Largest subscript: MSB
Smallest subscript: LSB

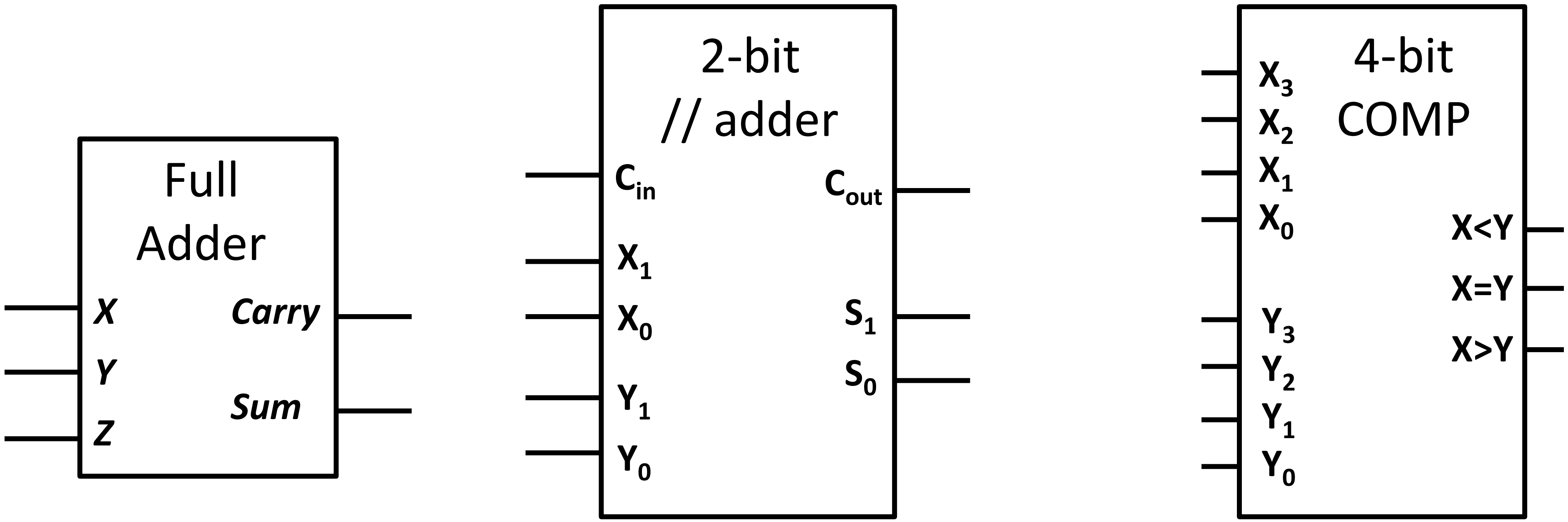
Q1.

Idea?

Need at least two 1s in the input for distance to be valid.

Count the number of 1s in *ABCDE*.
If ≥ 2 then $V = 1$, else $V = 0$.

Available blocks:

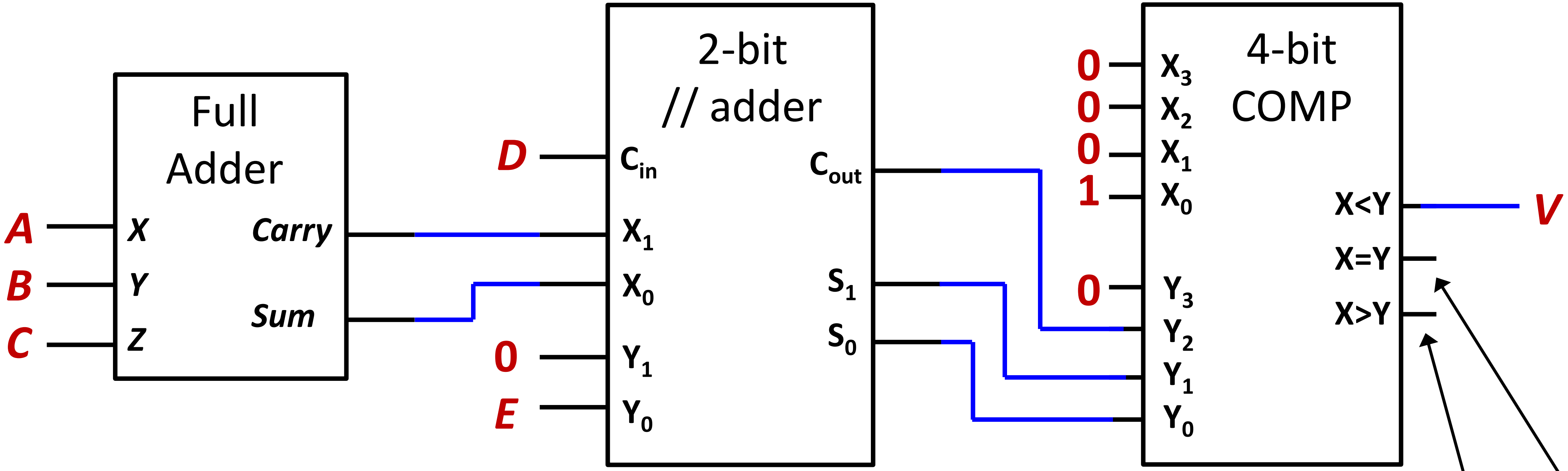


Q1.

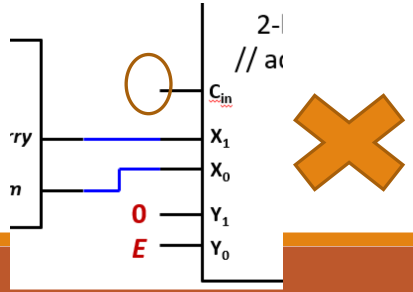
Idea?

Count the number of 1s in *ABCDE*.

A + B + C + D + E = (A + B + C) + D + E (integer addition)

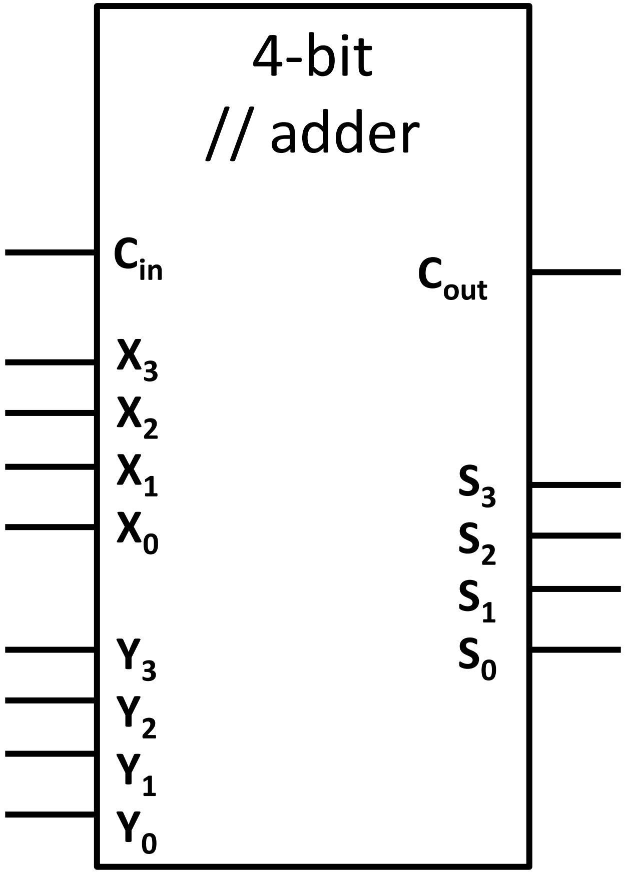


Important: All inputs must be connected to some value! Leave no inputs "hanging".



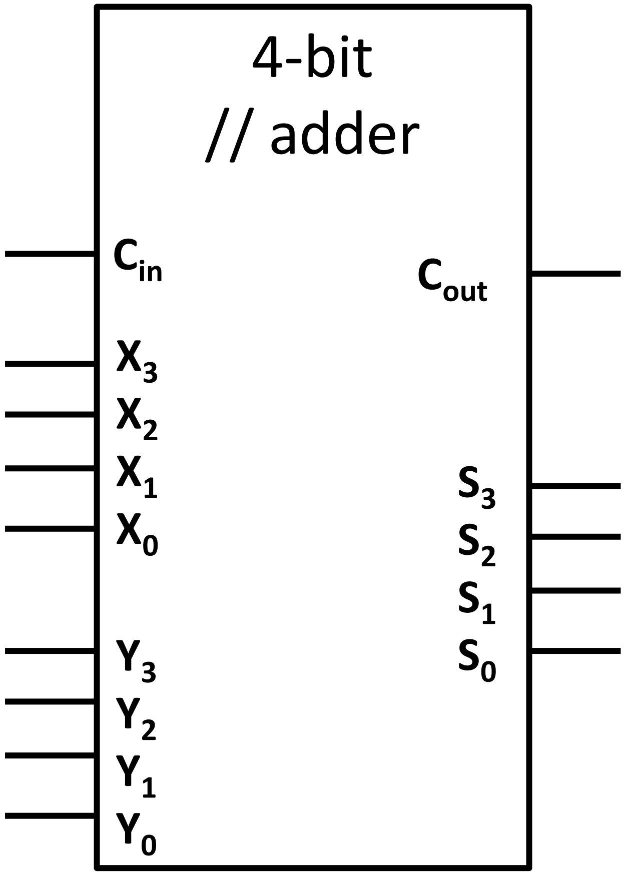
Unused outputs may be left alone.

Q2(a) $EFGH = (ABCD+1)/2$



A	B	C	D	E	F	G	H
0	0	0	0				
0	0	0	1				
0	0	1	0				
0	0	1	1				
0	1	0	0				
0	1	0	1				
0	1	1	0				
0	1	1	1				
1	0	0	0				
1	0	0	1				
1	0	1	0				
1	0	1	1				
1	1	0	0				
1	1	0	1				
1	1	1	0				
1	1	1	1				

Q2(a) $EFGH = (ABCD+1)/2$



How to do $ABCD$+1?

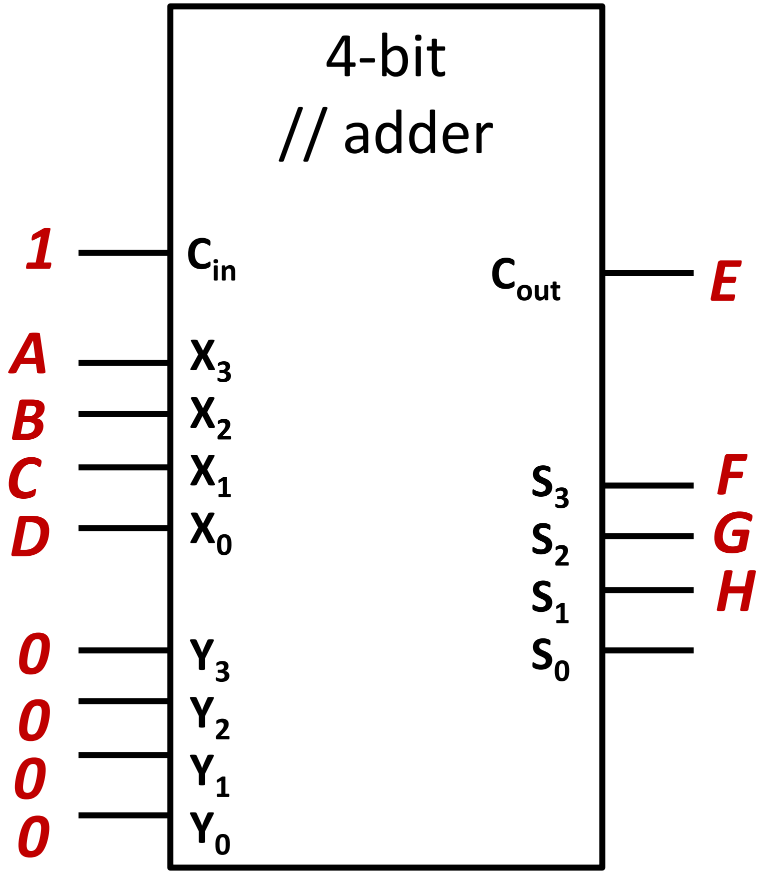
How to do divide by 2?

Is there anything special about the number 2?

Available blocks:

4-bit parallel adder and at most 1 logic gate

Q2(a) *EFGH* = (*ABCD*+1)/2 Dividing and multiplying by a fixed power of 2 is easy!

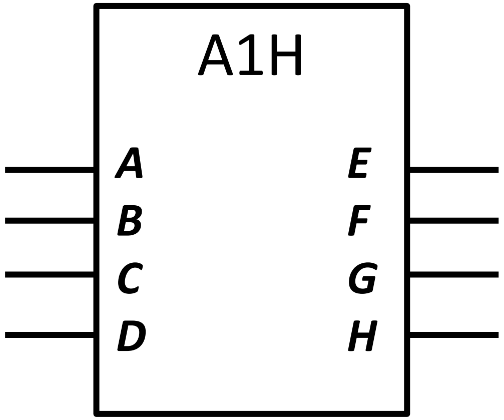


Just change the wiring:
Instead of S_0 -> H,
connect S_1 -> H …

Q2(b) 4221-to-8421 decimal code converter

Hint: Might have to use this

Digit	4221 Code	8421 Code
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0110	0100
5	1001	0101
6	1100	0110
7	1101	0111
8	1110	1000
9	1111	1001

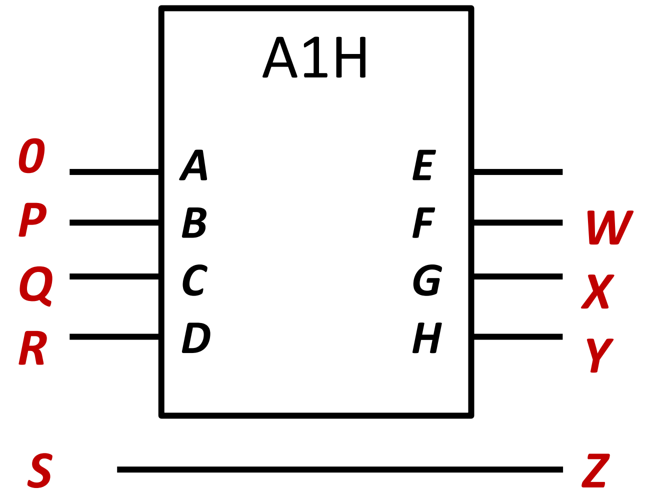


Q2(b) 4221-to-8421 decimal code converter

Digit	4221 Code	8421 Code
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0110	0100
5	1001	0101
6	1100	0110
7	1101	0111
8	1110	1000
9	1111	1001

1 is 1. LSB of 4221 code is the same as LSB of 8421 code.

Add-1-then-Half



Q3

BCD code

Digits:	0	1	2	3	4	5	6	7	8	9
Code:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Given two decimal digits *A* and *B*, represented by their BCD codes $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ respectively, implement a circuit without using any logic gates to calculate the BCD code of the 3-digit output of $(51\times A) + (20\times(B\%2))$, where % is the modulo operator. Name the outputs $F_{11}F_{10}F_9F_8\ F_7F_6F_5F_4\ F_3F_2F_1F_0$.

Hint: Fill in the table on the right.

A				5×*A*							
A_3	A_2	A_1	A_0								

Q3

BCD code

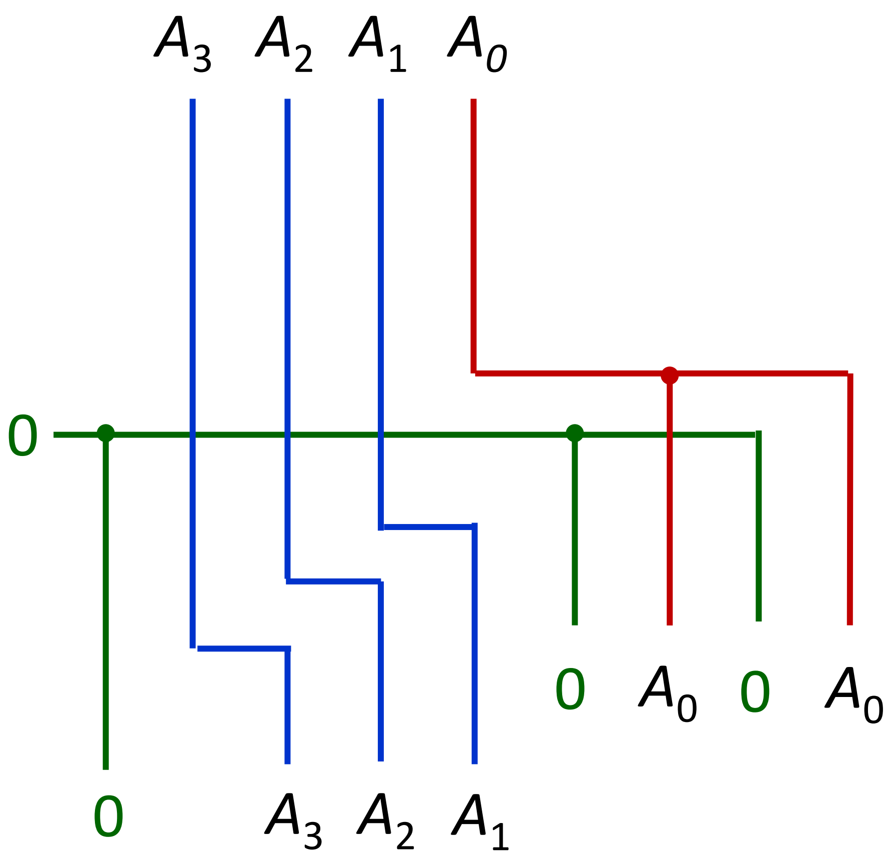
Digits:	0	1	2	3	4	5	6	7	8	9
Code:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

A				$5\times A$							
A_3	A_2	A_1	A_0								
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	1
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1	0	1	0	1
0	1	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	1	0	1
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	1	0	1

Q3

BCD code

Digits:	0	1	2	3	4	5	6	7	8	9
Code:	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

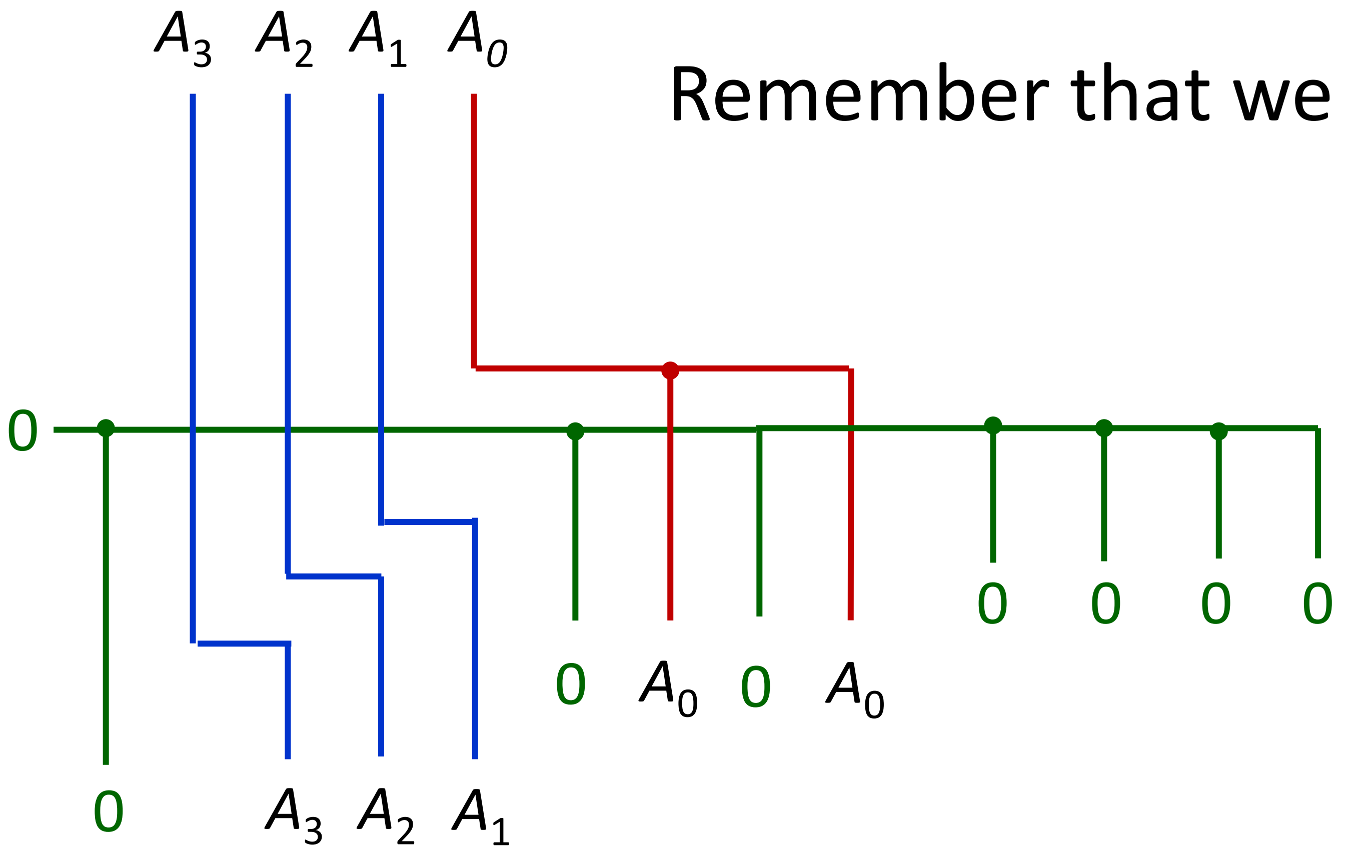


A				$5 \times A$							
A_3	A_2	A_1	A_0								
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	1	0	1
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	1	0	1	0	1
0	1	0	0	0	0	1	0	0	0	0	0
0	1	0	1	0	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0	0	0	0
0	1	1	1	0	0	1	1	0	1	0	1
1	0	0	0	0	1	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	1	0	1

Q3

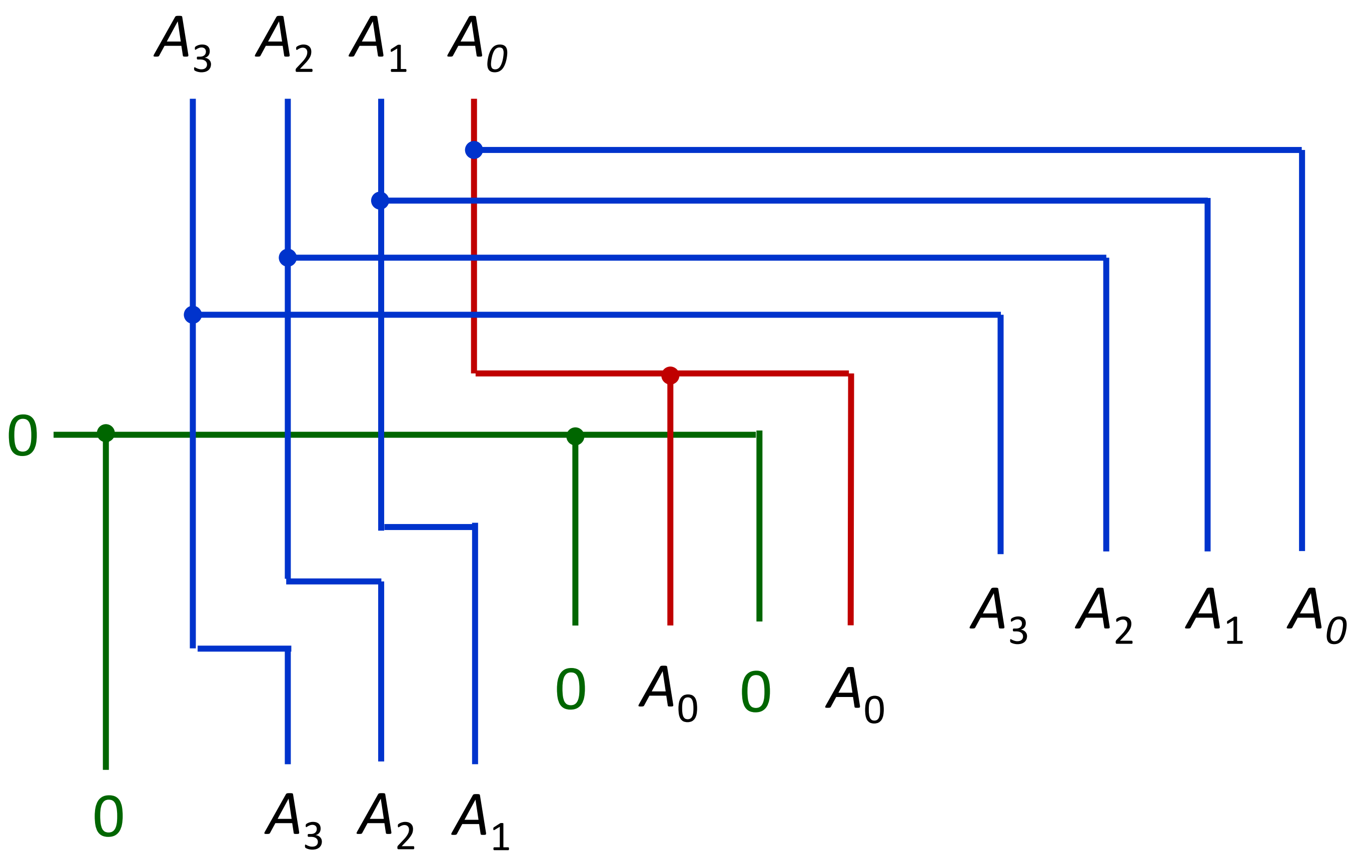
~~$5\times A$~~ ~~$50\times A$~~ $51\times A$

Remember that we are working with BCD



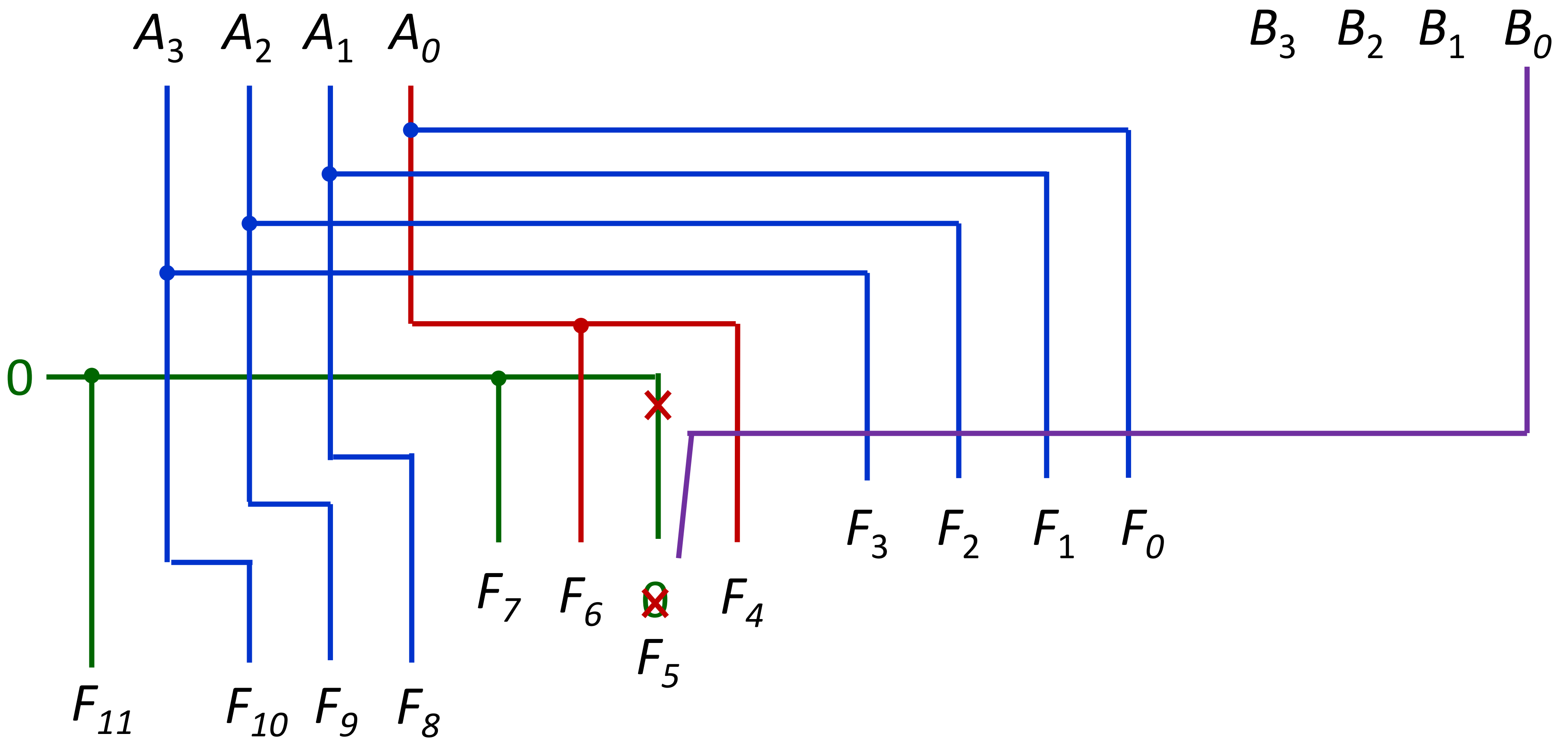
Q3

$\cancel{5\times A}$ $\cancel{50\times A}$ $\cancel{51\times A}$ $51\times A + (20\times(B\%2))$

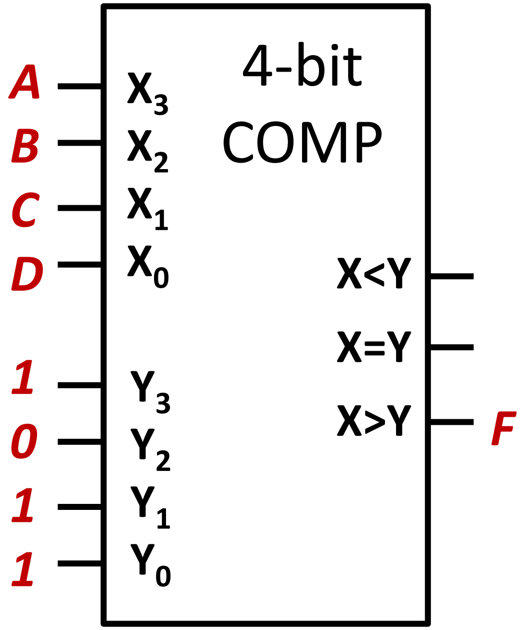


Q3

$\cancel{5\times A}$ $\cancel{50\times A}$ $\cancel{51\times A}$ $51\times A + (20\times(B\%2))$



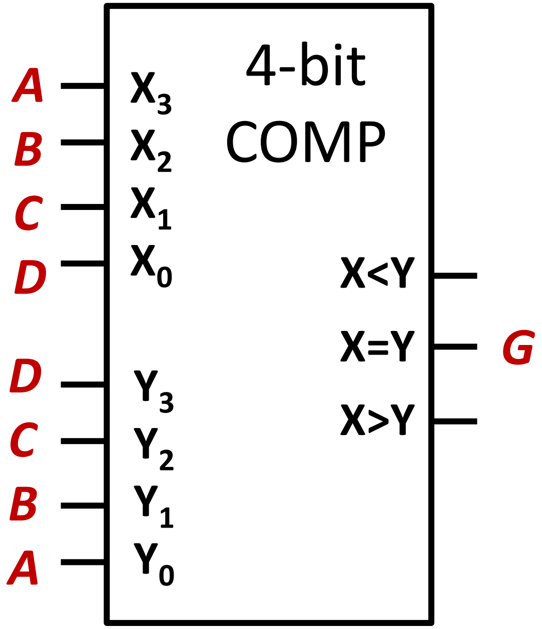
Q4(a) $F(A,B,C,D) = \Sigma m(12 - 15)$.



Q4(b) $G(A,B,C,D) = \Sigma m(0, 6, 9, 15)$.

A	*B*	*C*	*D*	*G*
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

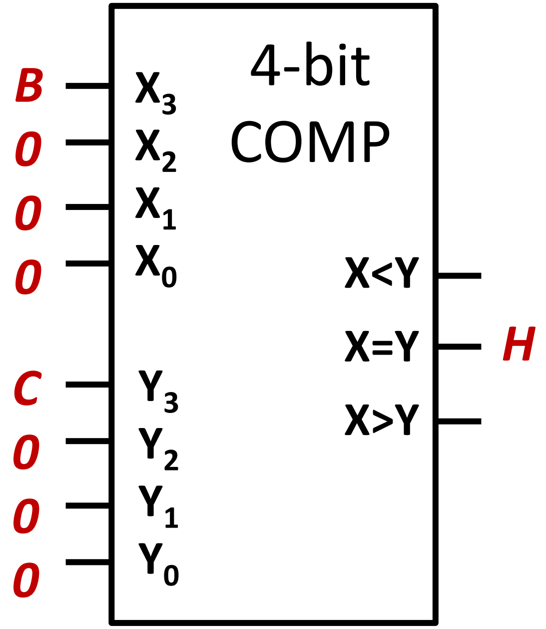
Observe the pattern of *ABCD* where *G*=1.



Q4(c) $H(A,B,C,D) = \Sigma m(0, 1, 6, 7, 8, 9, 14, 15)$.

A	*B*	*C*	*D*	*H*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

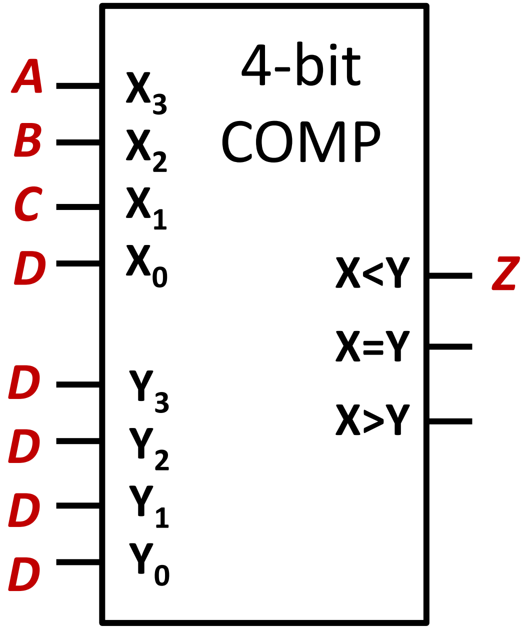
Observe the pattern of *ABCD* where *H*=1. Slightly tougher.



Q4(d) $Z(A,B,C,D) = \Sigma m(1, 3, 5, 7, 9, 11, 13)$.

A	B	C	D	Z
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Tough!

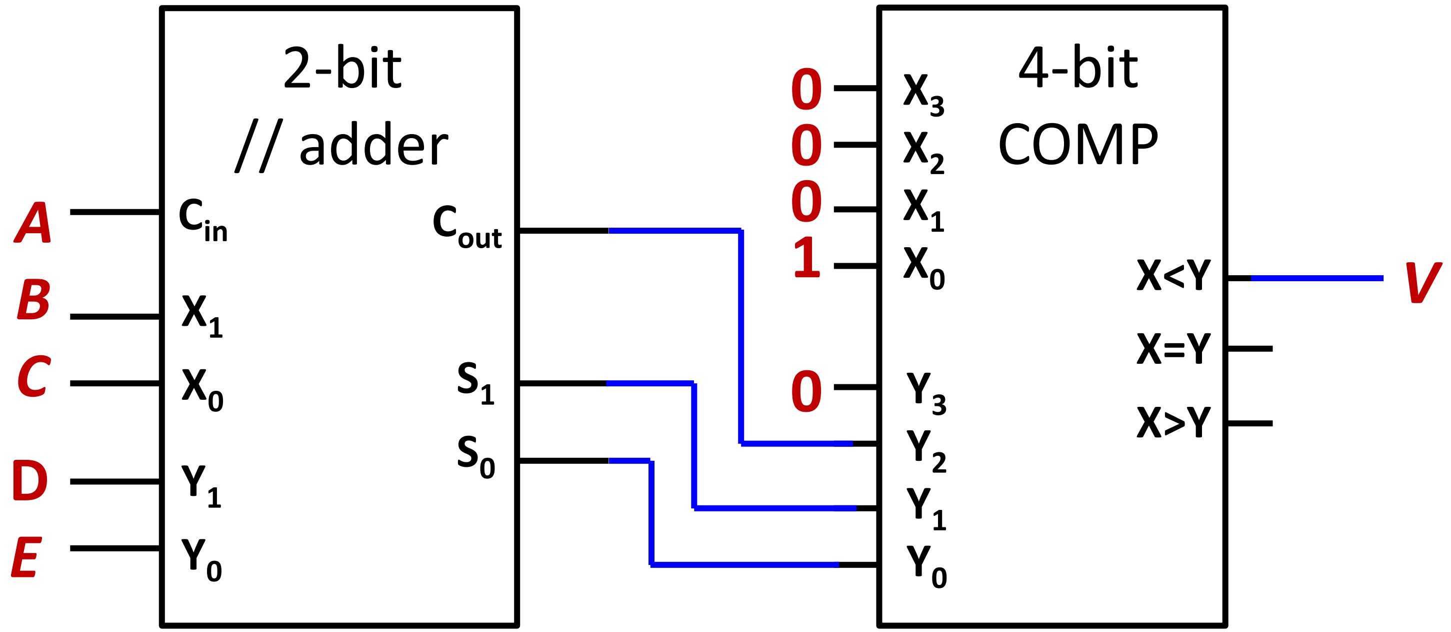


If this is 1 then it becomes very easy.

END OF FILE

Additional Question

1. Refer to tutorial Q1. Is the following combinational circuit another possible solution? Justify your answer.

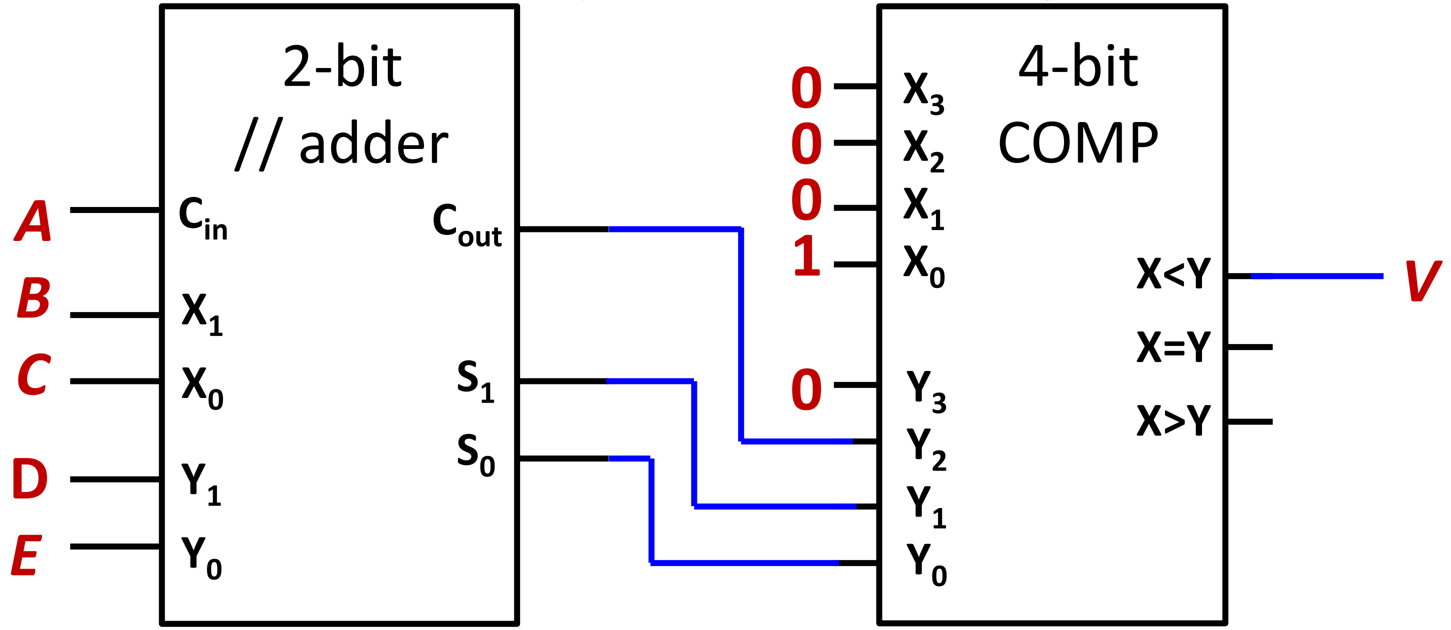


Answer

1. Refer to tutorial Q1. Is the following combinational circuit another possible solution? Justify your answer.

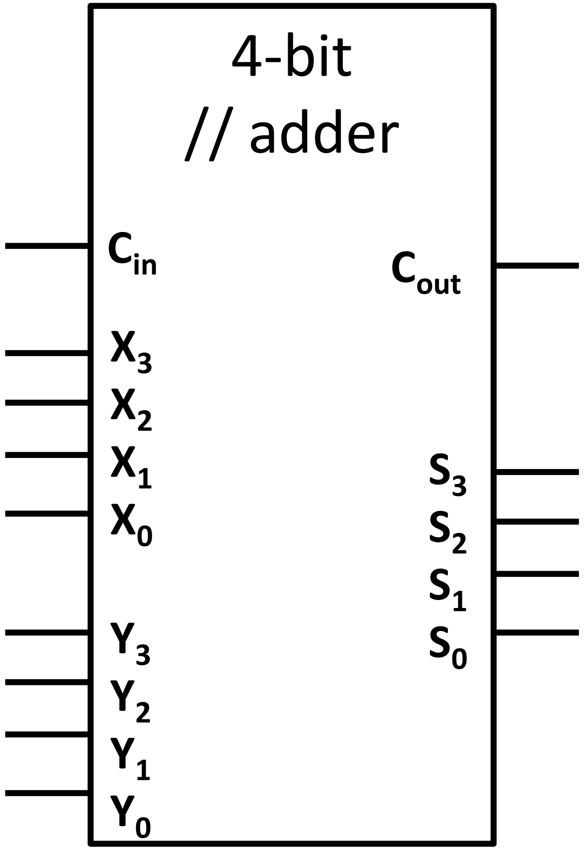
No, one problem is that we get $S_0 = A + C + E$ instead of $S_0 = A + B + C + D + E$.

Counterexample: ABCDE = 01000 is invalid but the 2-bit parallel adder would give 2 as the result.



Additional Question

2. Is it possible to implement $VWXYZ = 3(ABC) + 1$ using only one 4-bit parallel adder? If yes, how? If no, why not?



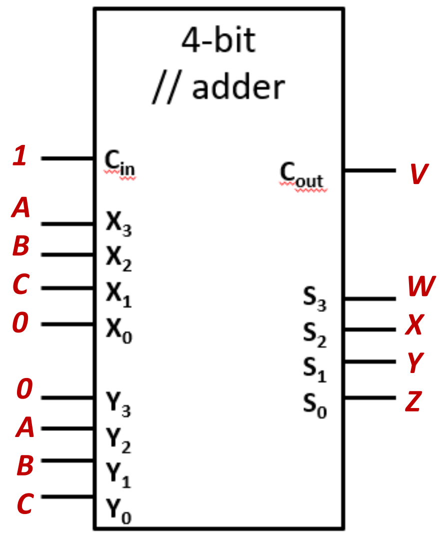
Answer

2. Is it possible to implement $VWXYZ = 3(ABC) + 1$ using only one 4-bit parallel adder? If yes, how? If no, why not?

Yes.

Recall how we implemented multiply by 3 using the sll and add

instructions in tutorial 2.



Additional Question

3. The table below gives the binary coded base-6 values for the six base-6 digits:

Digits:	**0**	**1**	**2**	**3**	**4**	**5**
Code:	0000	0001	0010	0011	0100	0101

For example, the decimal value 18 in base-6 is 30 which is represented as 011 000 in binary coded base-6 code.

Fill in the table below to find the binary coded base-6 code for 3*A when A is 5.

A				**3×*A***					
A_3	A_2	A_1	A_0						
0	1	0	1						

Answer

When filling in the table for tutorial question 3, we first perform 5×A in decimal and use the code to convert the result into BCD.

3. The table below gives the binary coded base-6 values for the six base-6 digits:

Digits:	0	1	2	3	4	5
Code:	0000	0001	0010	0011	0100	0101

For example, the decimal value 18 in base-6 is 30 which is represented as 011 000 in binary coded base-6 code.

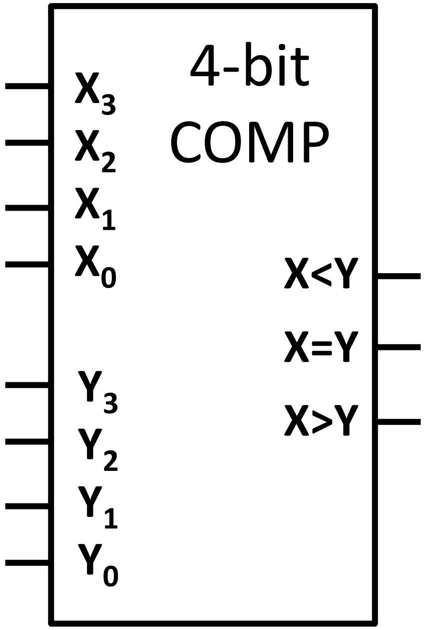
Fill in the table below to find the binary coded base-6 code for 3*A when A is 5.

A				**3×*A***					
A_3	A_2	A_1	A_0						
0	1	0	1	0	1	0	0	1	1

For this question, we can calculate in decimal 3×5 = 15 (from kindergarten). Then we convert 15_{10} to 23_6 and use the above table to get the binary code.

Additional Question

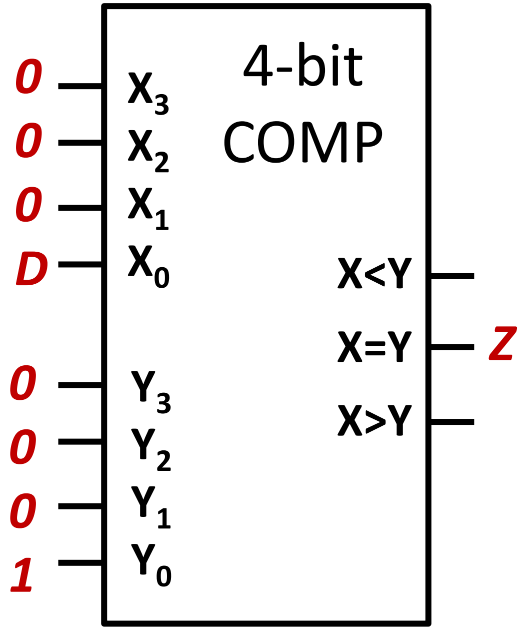
4. Follow the requirements for tutorial question 4 for X(A, B, C, D) = $\Sigma m(1, 3, 5, 7, 9, 11, 13, 15)$.



Answer

4. Follow the requirements for tutorial question 4 for X(A, B, C, D) = $\Sigma m(1, 3, 5, 7, 9, 11, 13, 15)$.

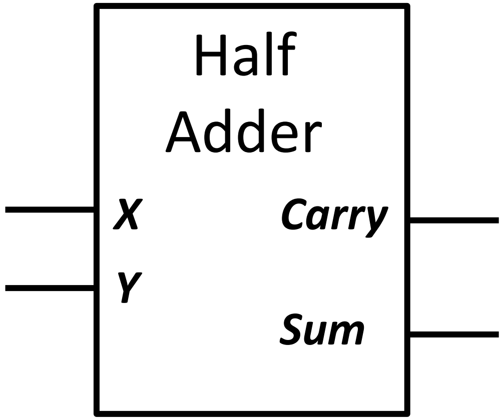
These are all the odd numbers.



Additional Question

5. Using only the 1-bit half-adder as shown in the diagram below, implement the function

$Y(A, B, C, D) = \Sigma m(1, 3, 5, 7, 8, 10, 12, 14)$.



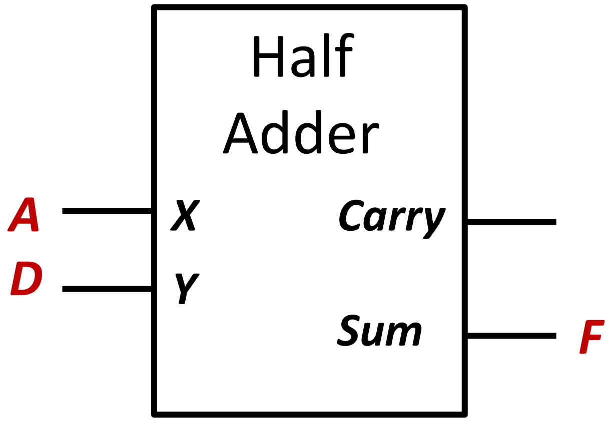
Answer

5. Using only the 1-bit half-adder as shown in the diagram below, impleme

F(A, B, C, D) = $\Sigma m(1, 3, 5, 7, 8, 10, 12, 14)$.

Since we have 8 minterms in the formula for F, once we find out that all the minterms in F have A $\neq$ D, we know that for the other 8 Minterms (those not in F), A = D. Therefore, we use F = A xor D.

A	B	C	D	F
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0



Answer

5. Using only the 1-bit half-adder as shown in the diagram below, implement the function

F(A, B, C, D) = $\Sigma m(1, 3, 5, 7, 8, 10, 12, 14)$.

If we want to notice that F = A $\neq$ D without drawing truth tables, we can also see that we want all the odd numbers < 8 and all the even numbers $\geq$ 8.

All numbers < 8 have A = 0 and all numbers $\geq$ 8 have A = 1.

We should know what D is for odd and even numbers.