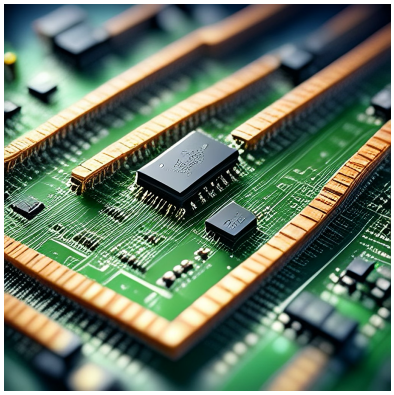
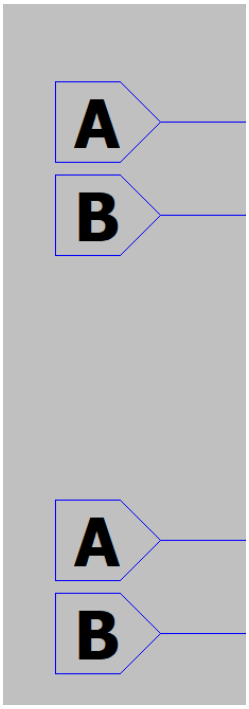


Lab Two Building with Gates

Monday Submission



Steps

1. Design the Block (Carry-In = 0)

• Overview:

Build the simpler block described in the last paragraph on the final page —the version with a carry-in fixed at zero.

• Tasks:

• Understand the Problem:

A. Describe what the block does.

1. I started by adding the inputs from the 'ground' component, right clicking them, selecting 'inputs' and labeling them A and B.

2. Second, I added the inverters for each opposite input source. In the situation, I added an inverter at B and an inverter at A.

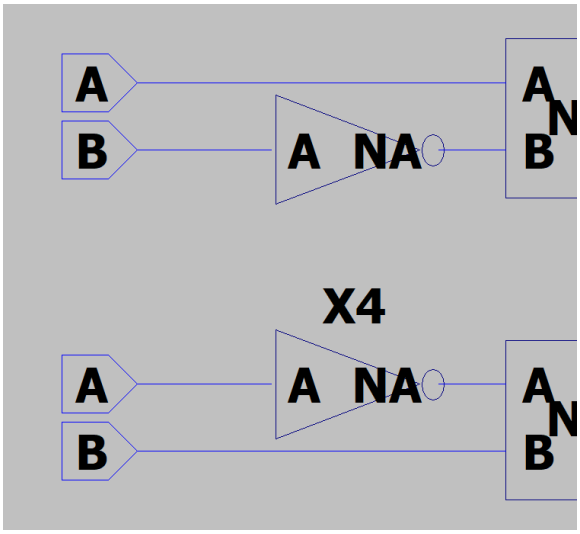
3. This choice was made because of the choice I made below to utilize two NAND gates to each 'handle' the two inputs.

4. This is demonstrated by the two half adder circuits below on the subsequent page.

5. Note the use of the AND gate and not a NAND.

6. Remember the notation "1 ⊕ 1 = Like adding 1+1 in binary, sum digit is 0 1 ⊕ 0 = 1 # Like adding 1+0 in binary, sum is 1"

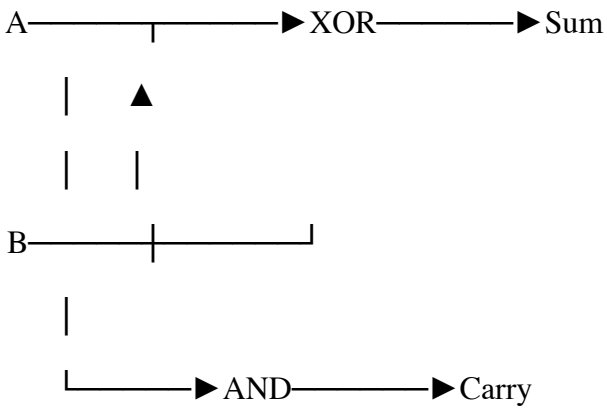
a. If it was a NAND, what would happen to the carry?



=== Half Adder Circuit ===

Inputs: A = 0, B = 1

Circuit Diagram:



Gate Operations:

XOR Gate: 0 ⊕ 1 = 1

Note the notation “A ⊕ B = (A AND NOT B) OR (NOT A AND B)”

AND Gate: 0 & 1 = 0

Note : “A & B = (A AND B)

Outputs:

Sum: 1

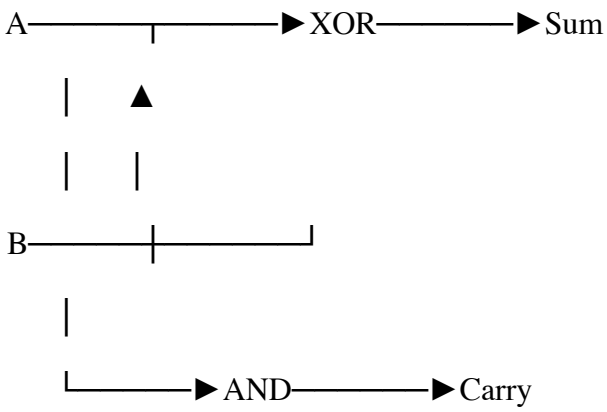
Carry: 0

Second Half Adder Circuit.

=== Half Adder Circuit ===

Inputs: A = 1, B = 0

Circuit Diagram:



Gate Operations:

XOR Gate: 1 ⊕ 0 = 1

AND Gate: 1 & 0 = 0

Outputs:

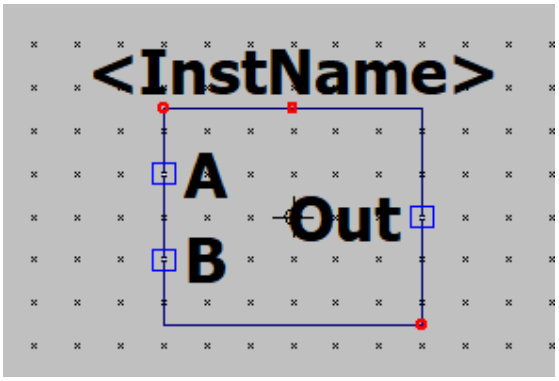
Sum: 1

Carry: 0

4. I chose the NAND gate because it will have a output that is on(1) if either input is on(1) as well as as a on(1) if both are low(0).

a. If both input ANB are zero I would still like the gate to produce an on result that can be handled by the ORx gate downstream.

b. A constraint of this is that if both inputs are one, then the inverter causes an off(o) output.



c. Then shows the exclusive or gay because it will differentiate if both inputs are dissimilar. This logic gate attribute is appealing.

5. Then this exclusive or gate leads to an output of Sum[0].

--- Component (NAND) ---

Truth Table:

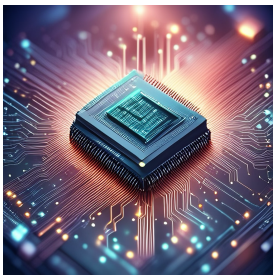
A B | NAND

0 0 | 1

0 1 | 1

1 0 | 1

1 1 | 0



K-map:

	B=0	B=1
A=0:	1	1
A=1:	1	0

SOP Expression:

Output = A'B' + A'B + AB’

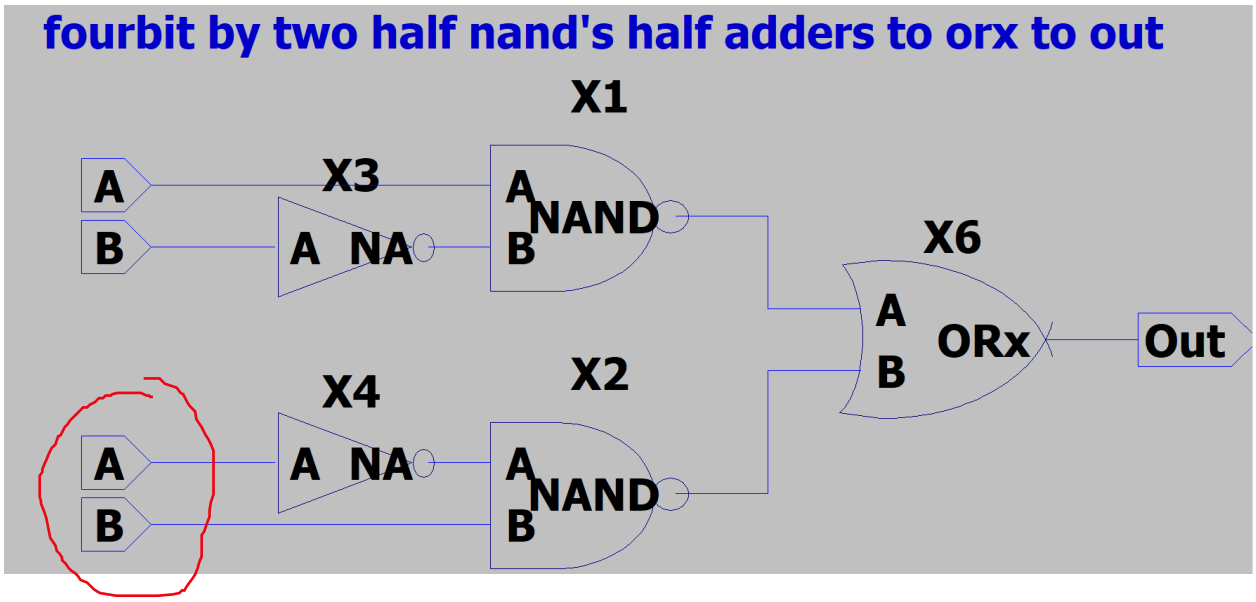
a. Unfortunately, I wasn't sure how to handle a potential carry during my initial attempt. To to be more specific I wasn't sure where the wire would be placed as well as whether or not an additional C bi-directional input would be needed. I explore this concept during my implementation of the subsequent logic gates to handle the Sum[n:0] as well as the carry[n+1:0].

b. I'm interested to see because of the properties of the two NAND gates utilizing an inverter previously to the gate, thus creating a circuit that is likely to move forward an on(1) or more, I am hopeful that the truth table as well as the KMap analysis logic will show where there is a carry in this initial stagein addition to the output sum[0].

This is the first of hierarchy symbol.

Note : I have labled the ‘output’ as Sum[0] as well as A[0] and B[0] inputs although this screenshot unfortunately didn't reflect that. I will improve that documentation error as soon as possible.

Below is my first attempt at a circuit to start the chain!



I have included in (text’ the description

Since you already have I/O pads for A and B (going into X1), these can be "none" directionality. Style-

This was my initial try.

Unfortunately I included four inputs rather than two.

Oops. Nevermind my comment above. Good catch

Additionally, I did not have a implementation devised yet for implementing the carry from the input’s A and B.

I revised this through these steps:

a. Removing the duplicate A and B inputs

b. Adding a C input.

c. Adding an additional AND gate to the configuration.

1. Why?

This additional AND gate was added because of the need for the carry operation.

--- Advanced Details for Component 3 (XNOR) ---

Truth Table:

A B | XNOR

0 0 | 1

For sum, it should be an XOR.

0 1 | 0

1 0 | 0

1 1 | 1

K-map:

	B=0	B=1
A=0:	1	0
A=1:	0	1

SOP Expression:

Output = A'B' + AB

• Create Truth Tables is the next exciting step.

Prepare two truth tables—one for SUM and one for CARRY-OUT.

- Derive the Logic:
  - This is listed under K-Map for each gate choice

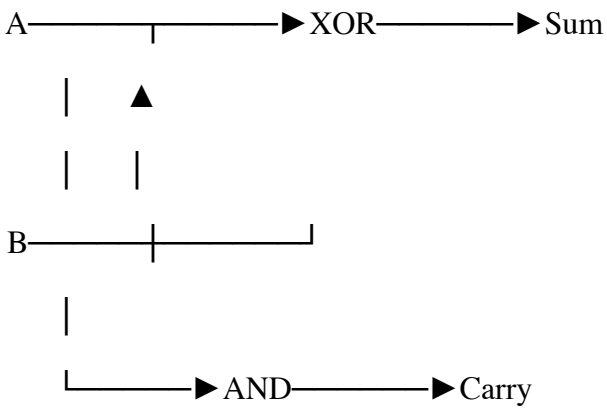
Use the Sum of Products (SOP) method to determine the logic for both SUM and CARRY-OUT.

CARRY-OUT Logic

=== Half Adder Circuit ===

Inputs: A = 1, B = 1

Circuit Diagram:



Gate Operations:

XOR Gate: 1 ⊕ 1 = 0

AND Gate: 1 & 1 = 1

Outputs:

Sum: 0

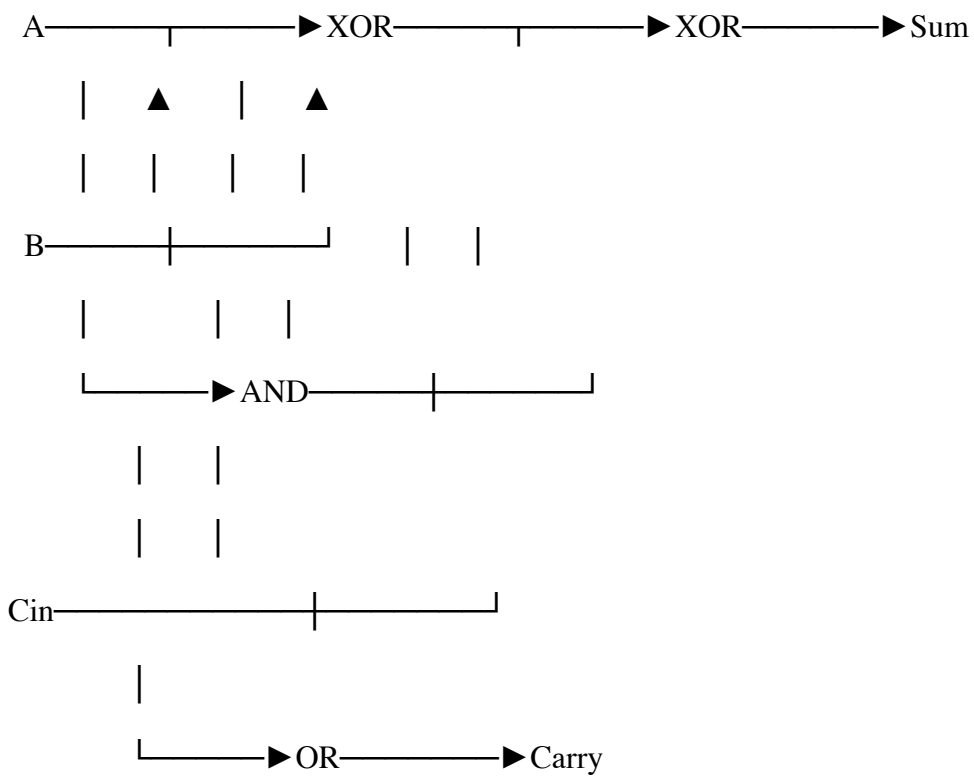
Carry: 1

=== Full Adder Circuit ===

Under construction.

Inputs: A = 1, B = 1, Carry_in = 1

Circuit Diagram:



Gate Operations:

First Half Adder:

XOR1: $1 \oplus 1 = 0$

AND1: $1 \& 1 = 1$

Second Half Adder:

XOR2: $0 \oplus 1 = 1$

AND2: $0 \& 1 = 0$

Final Carry:

OR: 1 | 0 = 1

Outputs:

Sum: 1

Carry_out: 1