Lab Two: Building with Gates (Carry-In = 0)

Monday Submission

February 4, 2025

1 Introduction

This document outlines the design and implementation of a digital logic block with a carry-in fixed at zero (LSB position in a ripple-carry context). The lab steps include creating half adders, full adders, truth tables, K-Maps, and final SOP expressions. Circuit simulations are conducted in LTSpice, and final blocks are tested with all possible input combinations.

2 Design Steps (Carry-In = 0)

2.1 Overview

This section describes a simpler block derived from the final page instructions, where the carry-in is constantly set to zero.

- **Task:** Build a half adder (HA) or simpler LSB block with carry-in = 0.
- **Implementation Observations:**
  1. Inputs labeled A and B are connected to ground or VCC in the schematic.
  2. Inverters can be used on inputs if using NAND gates to build simpler sub-blocks.
  3. Two NAND gates are considered so each handles a pair of inputs (or their complements).
  4. The logic to handle carry is not required here because *Carry-In* = 0.
  5. XOR logic is required for the sum bit, and an AND or NAND-based approach can generate *Carry-Out*.

Note on NAND Gate Use. Choosing a NAND gate can be beneficial because:

- A NAND output is 1 if any input is 0, and only 0 if all inputs are 1.
- This can feed into additional gates (e.g., OR or XOR) to handle certain logic downstream.

2.2 Half Adder Circuits

2.2.1 Half Adder Example 1

Inputs: A = 0, B = 1
Circuit Diagram:

$$\begin{tikzcd}
\text{A} \arrow[r] & \oplus \arrow[r] & \text{Sum} & \\
 & & & \uparrow \\
\text{B} \arrow[r] & \oplus & & \\
 & & & \downarrow \\
 & & \text{AND} \arrow[r] & \text{Carry}
\end{tikzcd}$$

Gate Operations:

- XOR Gate: $0 \oplus 1 = 1$.
$$A \oplus B = (A \wedge \neg B) \vee (\neg A \wedge B)$$
- AND Gate: $0 \wedge 1 = 0$.

Outputs:

$$\text{Sum} = 1, \quad \text{Carry} = 0$$

2.2.2 Half Adder Example 2

Inputs: A = 1, B = 0
Circuit Diagram:

$$\begin{tikzcd}
\text{A} \arrow[r] & \oplus \arrow[r] & \text{Sum} & \\
 & & & \uparrow \\
\text{B} \arrow[r] & \oplus & & \\
 & & & \downarrow \\
 & & \text{AND} \arrow[r] & \text{Carry}
\end{tikzcd}$$

Gate Operations:

- XOR Gate: $1 \oplus 0 = 1$
- AND Gate: $1 \wedge 0 = 0$

Outputs:

$$\text{Sum} = 1, \quad \text{Carry} = 0$$

Commentary on NAND Gates.

- A NAND output can be high (1) even if both inputs are 0, which can be useful downstream in OR or XOR gates.
- If both inputs are 1, the NAND output will be 0.
- Such configurations can feed an XOR gate for sum logic.

2.2.3 NAND Gate Details

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:

$$\text{NAND} = A'B' + A'B + AB'$$

2.2.4 Additional Considerations

- The potential carry wire in a half adder emerges from the AND gate; for more significant bits, a *Carry-In* would be needed.
- When building multi-bit adders, we chain the output carry of each block to the carry input of the next.

3 Half Adder and Full Adder Examples

3.1 Another Half Adder Example: (A = 1, B = 1)

Inputs: $A = 1, B = 1$
Circuit Diagram:

$$\begin{tikzcd}
\text{A} \arrow[r] & \oplus \arrow[r] & \text{Sum} & \\
& & & \uparrow \\
\text{B} \arrow[r] & \oplus & & \\
& & & \downarrow \\
& & \text{AND} \arrow[r] & \text{Carry}
\end{tikzcd}$$

Gate Operations:

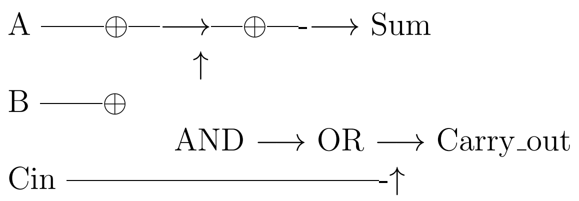
- XOR Gate: $1 \oplus 1 = 0$
- AND Gate: $1 \wedge 1 = 1$

Outputs:

$$\text{Sum} = 0, \quad \text{Carry} = 1$$

3.2 Full Adder Example (A = 1, B = 1, Carry_in = 1)

Circuit Diagram:



Gate Operations (Step-by-Step):

1. *First Half Adder:*
   - XOR1: $1 \oplus 1 = 0$
   - AND1: $1 \wedge 1 = 1$
2. *Second Half Adder:*
   - XOR2: (XOR1 output $= 0) \oplus ($Carry_in $= 1) = 1$
   - AND2: $0 \wedge 1 = 0$
3. *Final Carry:*

$$\mathrm{OR}(\mathrm{AND1}, \mathrm{AND2}) = 1 \vee 0 = 1$$

Outputs:

$$\mathrm{Sum} = 1, \quad \mathrm{Carry\_out} = 1$$

4 Implementation in LTSpice: Next Steps for me to complete

- Enter the circuits into LTSpice.
- Create a symbol for the block.
- Develop a new schematic to test the block.
- Simulate all possible inputs, and include screen captures of your simulation results.
- Ensure that all design files (`*.asc` and `*.asy`) are backed up.

5 LSB Block Design (Simplified, Carry-In = 0)

5.1 Overview

Once the main circuit is working, a simplified version for the least significant bit (LSB) can be designed. Since the carry-in is always zero, it is not a required input.

5.2 Tasks

- **Understand the Problem:** Describe what this LSB block does.
- **Create a Truth Table:** Summarize inputs/outputs for the new block (A, B, sum, carry-out).
- **Derive the Logic:** Use the Sum of Products (SOP) method (or other) to define the block's behavior.

5.3 Implementation in LTSpice

- Enter the new block into LTSpice.
- Create a symbol for the new block.
- Build a schematic to test this block.
- Simulate the block to confirm its operation.

5.4 Integration

- Replace the original LSB block with the new design in the overall circuit.
- Verify the integrated circuit still works as intended.

6 Lab 2 Requirements

6.1 Part 1: Main Block Design (Carry-In = 0)

1. **Design the Block (Carry-In = 0):**
   - Understand and describe block function.
   - Create truth tables for SUM and CARRY-OUT.
   - Derive logic using the SOP method.
2. **Implementation:**
   - Enter circuits in LTSpice.
   - Create a block symbol.
   - Develop a test schematic.
   - Simulate all inputs (include screenshots).
3. **File Backup:**
   - Backup all design files (`*.asc` and `*.asy`).

6.2 Part 2: LSB Block Optimization

1. **Design the Simplified LSB Block:**
    - Analyze requirements.
    - Create a truth table.
    - Derive logic (SOP or other).
2. **Implementation:**
    - Enter new block into LTSpice.
    - Create a block symbol.
    - Develop a test schematic.
    - Simulate the block.
3. **Integration:**
    - Replace original LSB block with new design.
    - Verify the circuit functions correctly.

7 Documentation and Final Report

All steps mentioned above must be documented thoroughly. The final, combined documentation (including Wednesday's work and today's additional steps) is due on Monday, February 3. Make sure to include:

- Explanations of each design choice.
- Truth tables and K-Maps for the key logic components.
- Detailed simulation results with screenshots.
- All relevant files in a well-organized folder.