4-Bit Ripple Carry Adder Explanation and Diagrams

Below is an explanation and a diagram (using text-based formatting) of a full 4-bit ripple carry adder. In this design, four full adder blocks are cascaded. Each full adder adds the corresponding bits of two 4-bit numbers (A and B) along with a carry input. The carry-out from each stage "ripples" to the next stage's carry-in.

Full 4-Bit Ripple Carry Adder Overview

- **Stage 0 (LSB):**
  - **Inputs:** A[0], B[0]
  - **Carry-in:** Fixed at 0 (since it's the LSB)
  - **Outputs:**
    - $\text{Sum}_0 = A[0] \oplus B[0]$
    - $\text{Carry}_0 = A[0] \cdot B[0]$
- **Stages 1 to 3 (Full Adders):** For each bit position i (where $i = 1, 2, 3$):
  - **Inputs:** A[i], B[i] and the carry-in from the previous stage (Carry_{i-1})
  - **Outputs:**
    - $\text{Sum}_i = A[i] \oplus B[i] \oplus \text{Carry}_{i-1}$
    - $\text{Carry}_i = (A[i] \cdot B[i]) + (B[i] \cdot \text{Carry}_{i-1}) + (A[i] \cdot \text{Carry}_{i-1})$

Text-Based Diagram

```
4-bit Ripple Carry Adder
-------------------------

  A[0] -----\
             \
              [ Full Adder 0 ]--- Sum[0]
  B[0] -----/       |
             (Cin=0)  |--> Carry0

  A[1] -----\
             \
              [ Full Adder 1 ]--- Sum[1]
  B[1] -----/       |
           Carry0--->|--> Carry1

  A[2] -----\
             \
              [ Full Adder 2 ]--- Sum[2]
  B[2] -----/       |
           Carry1--->|--> Carry2

  A[3] -----\
             \
              [ Full Adder 3 ]--- Sum[3]
  B[3] -----/       |
           Carry2--->|--> Carry3  (Final Carry-Out)
```

Details of Each Full Adder Block

1. **Full Adder Logic Equations:** For stages 1–3 (and for stage 0 with the note that Cin = 0), the full adder uses these logic equations:

$$\text{Sum}[i] = A[i] \oplus B[i] \oplus \text{Cin}$$

$$\text{Carry-out} = (A[i] \cdot B[i]) + (B[i] \cdot \text{Cin}) + (A[i] \cdot \text{Cin})$$

2. **LSB Special Case (Stage 0):** Since Cin is 0, the equations for Stage 0 simplify to:

$$\text{Sum}_0 = A[0] \oplus B[0]$$

$$\text{Carry}_0 = A[0] \cdot B[0]$$

How It Works:

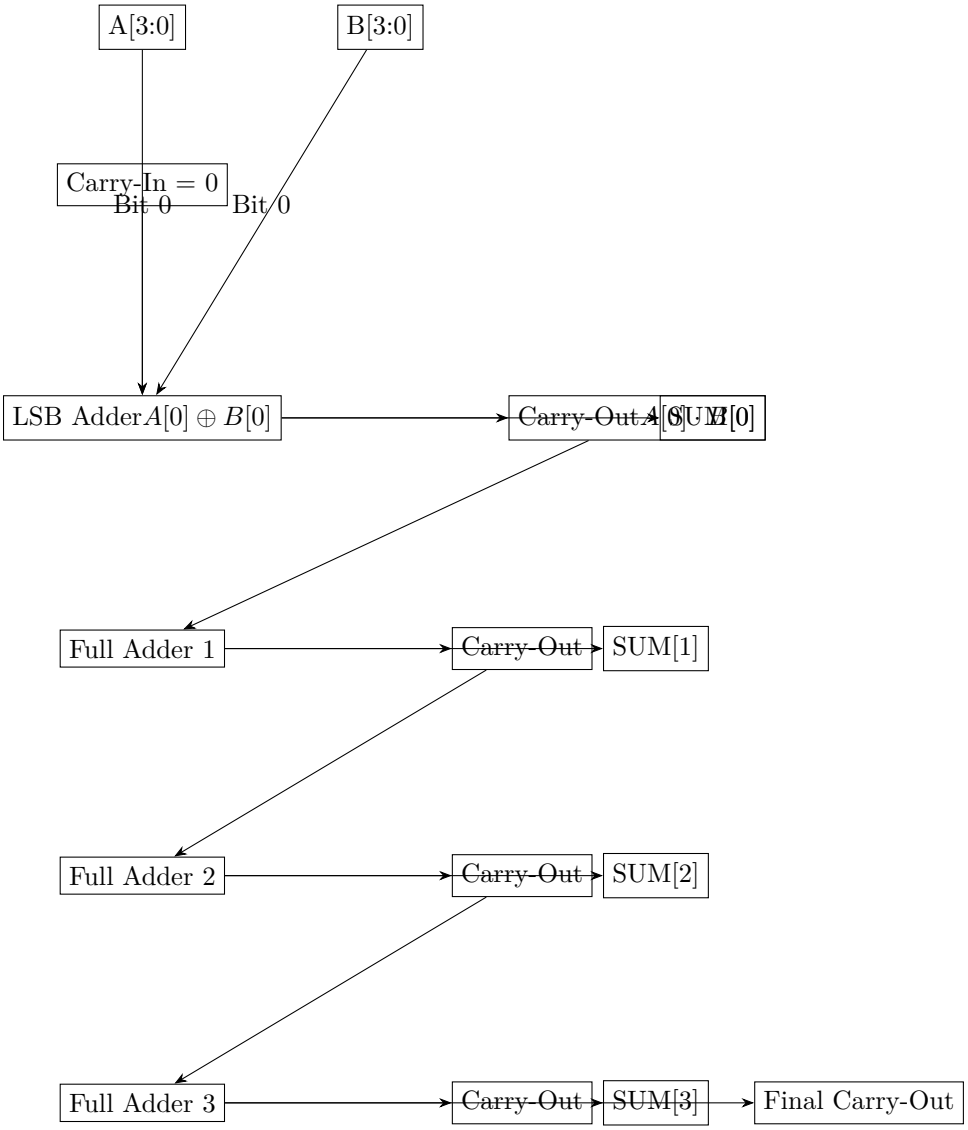
- **Stage 0:** The adder for the least significant bit (LSB) computes the sum of A[0] and B[0] without needing a carry-in. If both A[0] and B[0] are 1, a carry is produced for the next stage.
- **Stages 1–3:** Each subsequent full adder takes its carry-in from the previous stage's carry-out. This chain of carry propagation is why the circuit is called a "ripple carry adder" — the carry "ripples" from the LSB to the MSB.
- **Final Output:** The 4-bit sum is given by the outputs Sum[3:0]. An extra carry-out (Carry_3) is available if an overflow occurs (i.e., if the sum exceeds 4 bits).

This design meets the lab's requirements by cascading the basic full adder circuits (with the simplified LSB) to create a complete 4-bit ripple carry adder. You can implement this design in LTSpice by creating the individual full adder blocks (or using your previously designed LSB block for Stage 0) and then connecting them as shown above.

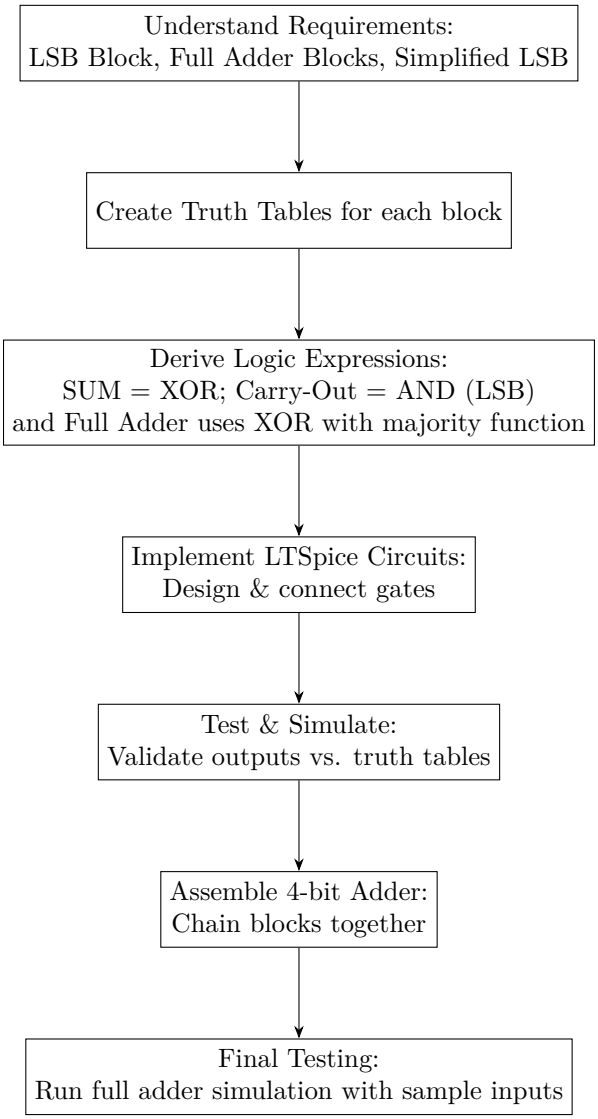
LaTeX/TikZ Diagrams

Below are nine diagrams corresponding to the design process of the 4-bit ripple carry adder.

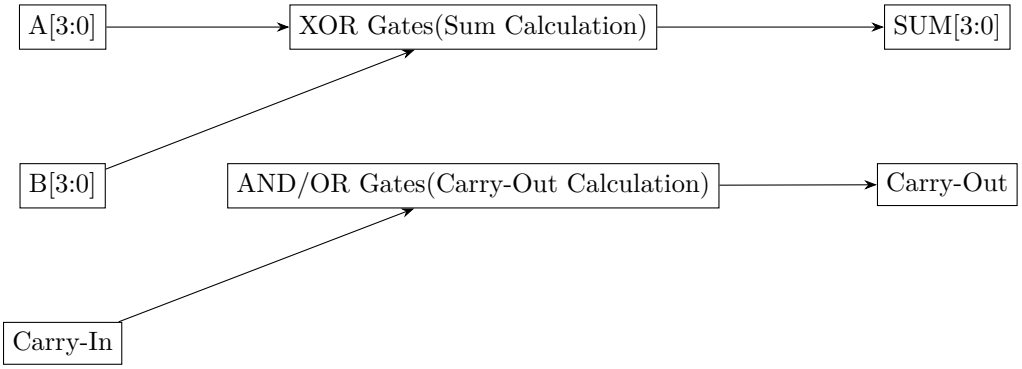
1. Block Diagram of the 4-Bit Ripple Carry Adder



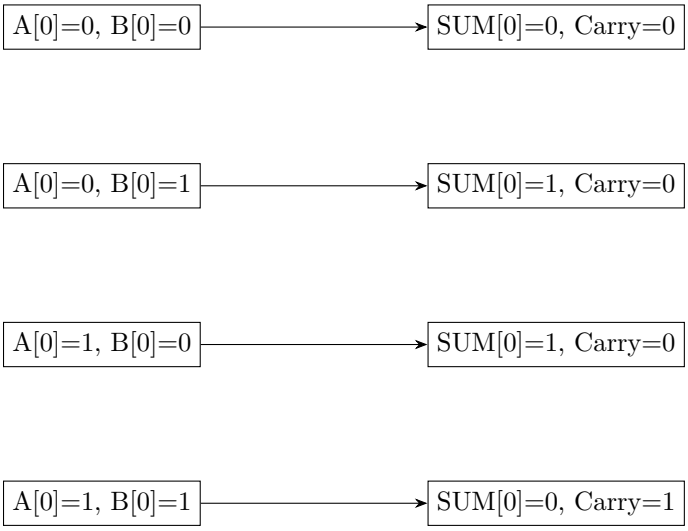
2. Flowchart for the Design Process



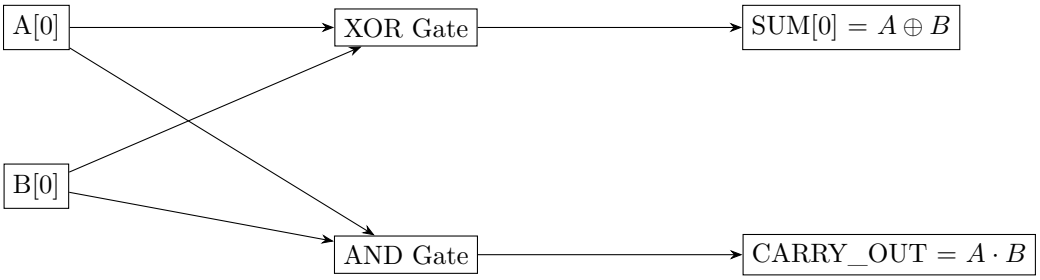
3. Data Flow Diagram (DFD)



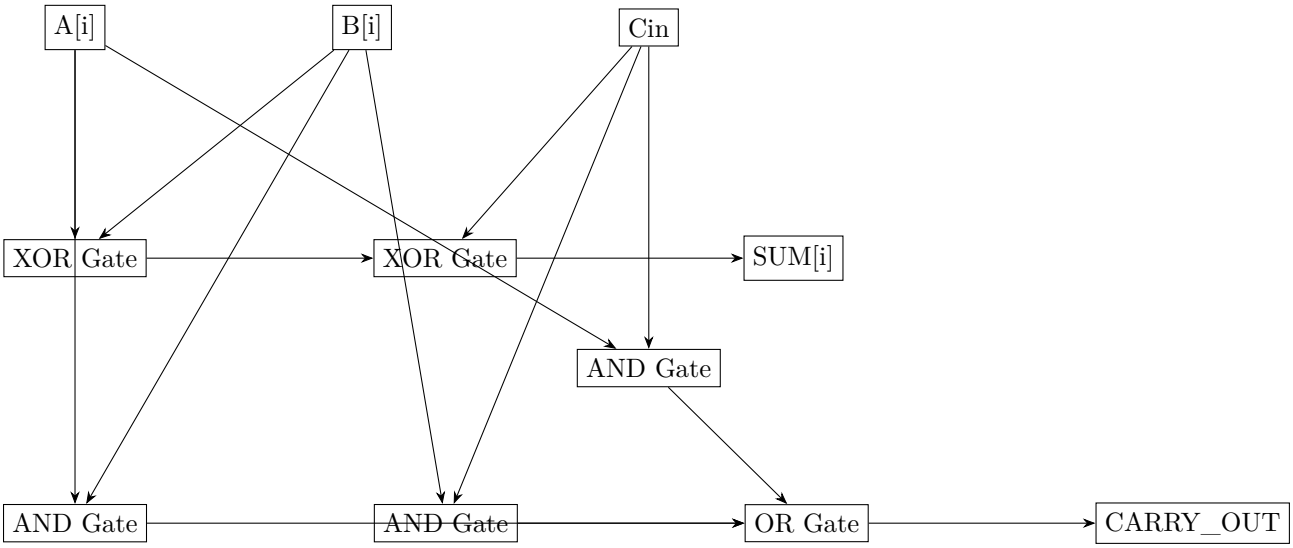
4. Truth Table for LSB Block (Carry-In = 0)



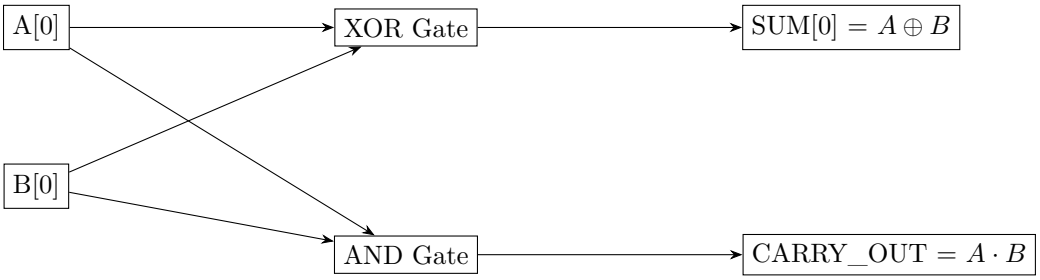
5. Logic Circuit Diagram for LSB Adder



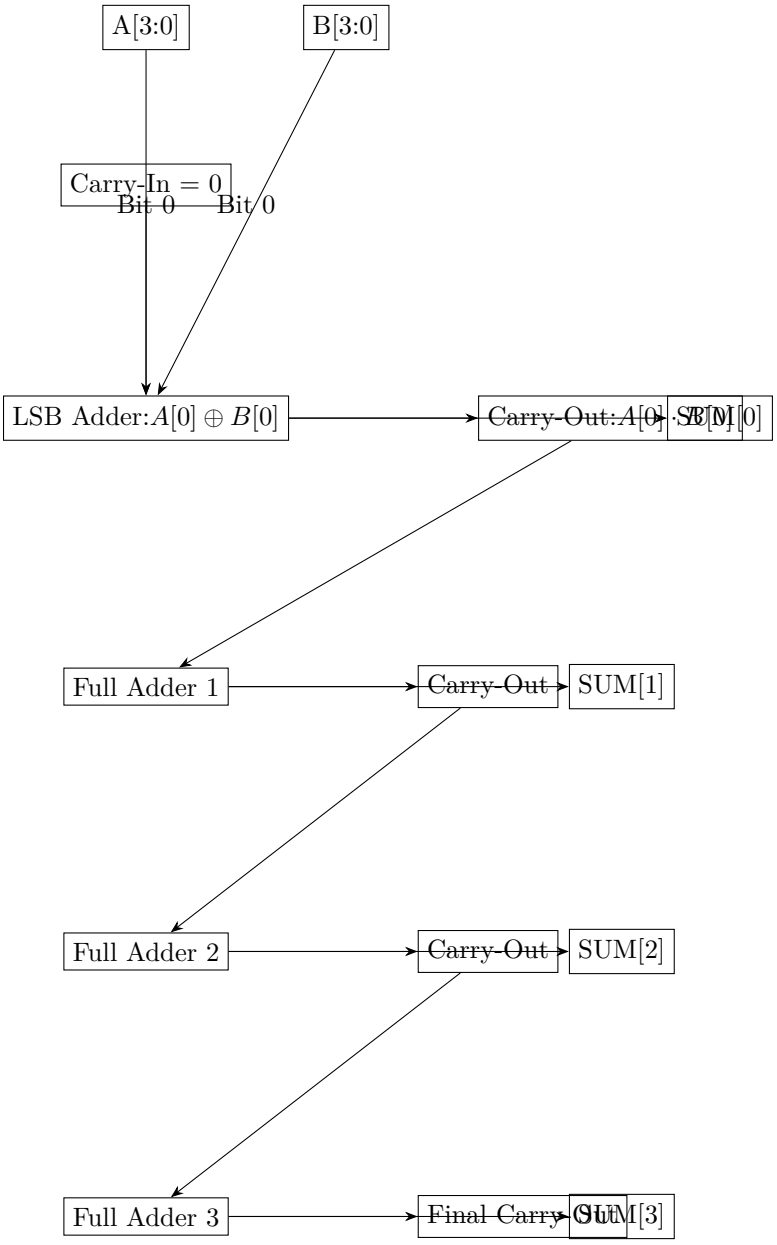
6. Full Adder Circuit Diagram



7. Simplified LSB Block Diagram



8. Integrated 4-Bit Ripple Carry Adder Diagram



9. Simulation Waveform Diagram (Sequence Diagram)

Test Case 4-bit Adder

Input: 0000 + 0000 ⟷ Output: 00000(No carry propagation)

Input: 0001 + 1111 ⟷ Output: 10000(Carry propagates)

Input: 1010 + 0101 ⟷ Output: 11111(Alternating bits addition)