Lab Report: Integer Multiplier Circuit Design in Logisim

Course: CS201
Lab Title: Integer Multiplier Circuit
Objective: To investigate the implementation of integer multiplication using hardware components in Logisim.

Introduction

The goal of this lab is to build a 4-bit multiplier circuit using Logisim and simulate the hardware approach for multiplication. The design will use registers, adders, control logic, and a shift register to execute an integer multiplication. We will explore how multiplication is implemented in hardware, particularly focusing on a step-by-step execution of multiplication using binary arithmetic and registers.

The multiplier circuit will demonstrate how machines can multiply numbers using control logic, adders, and registers. This circuit will be based on an algorithm that uses shifting and adding techniques to compute the product of two binary numbers.

Algorithm of Integer Multiplication (Hardware Approach)

We are going to implement a binary multiplication using the following steps:

1. **Initialization:**
   - Set `A` (Accumulator) to 0.
   - Set `C` (Carry Bit) to 0.
   - Load the Multiplicand `M` into register `M`.
   - Load the Multiplier `Q` into register `Q`.
   - Initialize the Counter to `n`, where `n` is the bit size of the registers (4 bits for this lab).
2. **Repeat the following steps until Counter = 0:**
   - **Step 1:** If the least significant bit (`Q0`) of the Multiplier `Q` is 1:
     - Add the Multiplicand `M` to the Accumulator `A`.
     - Shift the Carry `C`, Accumulator `A`, and Multiplier `Q` to the right (C→A→Q).
   - **Step 2:** If `Q0` is 0:
     - Shift the Carry `C`, Accumulator `A`, and Multiplier `Q` to the right.
   - **Step 3:** Decrease the Counter by 1.
3. **Stop the process when the Counter reaches 0.**
   The product of the multiplication will be stored in the `A` and `Q` registers combined, where `A` holds the higher bits and `Q` holds the lower bits of the result.

Circuit Components

The circuit comprises the following essential components:

- **M Register (4-bit):** Holds the Multiplicand.
- **Q Register (4-bit):** Holds the Multiplier.
- **A Register (4-bit):** Initialized to 0 to store the partial products.
- **C Register (1-bit):** Stores the carry from the adder.
- **4-bit Adder:** Adds the contents of the `A` and `M` registers when necessary.

- **Shift Register:** Simultaneously shifts the `C`, `A`, and `Q` registers to the right after every step.
- **Control Logic:** Manages the shifting, loading, and adding operations.

Step-by-Step Implementation in Logisim

1. Build the 4-bit Multiplier Circuit:

1. **Multiplicand (`M`):**
   - Use a 4-bit input device to load the multiplicand into the `M` register.
2. **Multiplier (`Q`):**
   - Use another 4-bit input device to load the multiplier into the `Q` register.
3. **C Register (Carry Bit):**
   - Implement a 1-bit `D flip-flop` to act as the carry register `C`.
   - It will store the carry-out bit from the adder.
4. **A Register (Accumulator):**
   - Initialize the `A` register to zero at the start.
   - Use a 4-bit `D flip-flop` for storing the partial sums.
5. **Adder:**
   - Build a 4-bit adder using Logisim's arithmetic components.
   - This will add the contents of `M` and `A` when `Q0 = 1`.
6. **Shift Registers:**
   - Implement shift registers for the `C`, `A`, and `Q` registers.
   - After each iteration, shift the contents of `C`, `A`, and `Q` to the right.
7. **Control Logic:**
   - Use control input devices (1-bit) to manage the loading, shifting, and adding of the registers.
   - A counter (4 iterations for 4-bit inputs) will keep track of the number of shifts performed.

Steps to Run the Multiplication:

1. **Step 1: Set Initial Values**
   - Load `M = 1100` (multiplicand) into the `M` register.
   - Load `Q = 1110` (multiplier) into the `Q` register.
   - Set `C = 0` (carry bit) and `A = 0000` (accumulator).
2. **Step 2: Perform Multiplication**
   - Follow the multiplication algorithm described above. For each step, check the least significant bit (`Q0`).
   - If `Q0 = 1`, add the contents of `M` to `A`.
   - Shift the `C`, `A`, and `Q` registers to the right.
   - Repeat for 4 cycles.

Simulation: Multiplication of 1100 × 1110

Initial Values:

- `M = 1100` (Multiplicand)
- `Q = 1110` (Multiplier)
- `A = 0000` (Accumulator)

- `C = 0` (Carry bit)

Step-by-Step Operations (Simulation Table):

Iteration	C	A (Accumulator)	Q (Multiplier)	Operation

Iteration	C	A (Accumulator)	Q (Multiplier)	Operation
1	0	1100	1110	Add M to A (Q0=1)
	0	0110	1111	Shift right
2	0	0110	1111	No add (Q0=0)
	0	0011	1111	Shift right
3	0	0011	1111	Add M to A (Q0=1)
	1	0001	1111	Shift right
4	1	1000	1111	Add M to A (Q0=1)
	1	1000	1111	Final result

Final Result:

The final product is stored in both `A` and `Q`. The register `A` contains the higher 4 bits, and `Q` contains the lower 4 bits of the result. The final product of 1100 × 1110 is `11110000` in binary, equivalent to 240 in decimal.