2X2 Matrix Multiplication

UCS2311---Digital Design Lab

Submitted by

Monishkumar Balaji

3122235001085

Munish Madhav

3122235001085

SRI SIVASUBRAMANIYA NADAR COLLEGE OF ENGINEERING,

KALVAKKAM- 603 110

DECEMBER 2024

1. Problem Statement

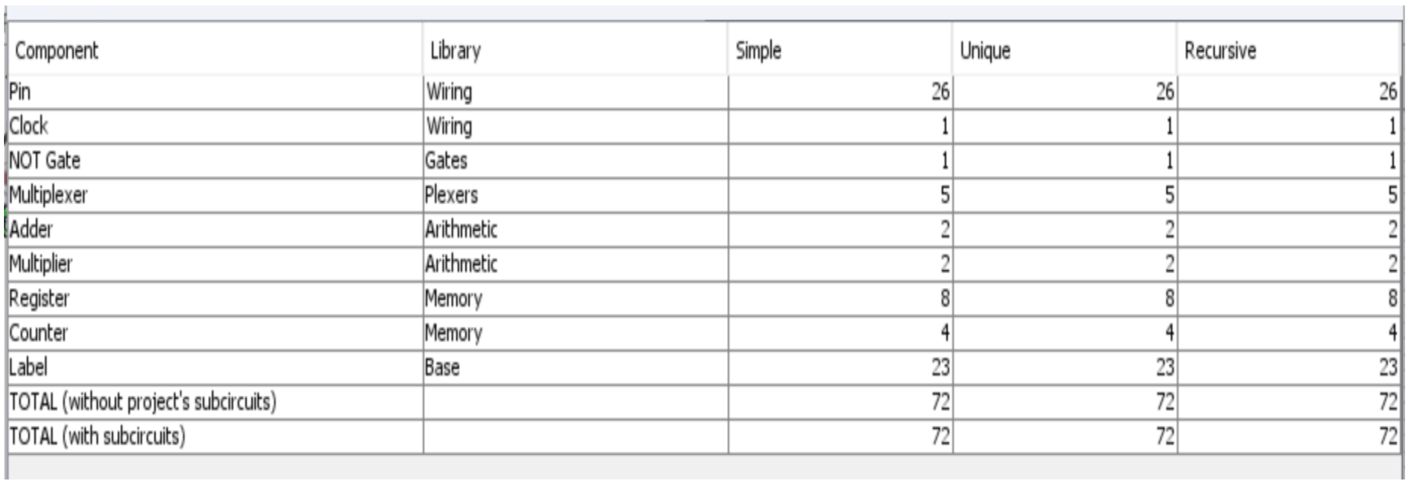
This project aims to design and implement a digital circuit in Logisim to perform 2x2 matrix multiplication with results represented in up to 6 bits. Matrix multiplication is fundamental in many applications across computing, engineering, and mathematics. By implementing it in Logisim, this project showcases digital design techniques for binary operations and highlights bit constraint management for accurate result representation. The goal is to compute each element of the resulting matrix by combining multiplication and addition of corresponding elements from two 2x2 input matrices.

2. Abstract

This project involves constructing a digital logic circuit to perform 2x2 matrix multiplication using Logisim. The primary objective is to develop a circuit that accurately computes the product of two matrices with each element represented in binary form, allowing results up to 6 bits in length. Components such as adders, binary multipliers, and logic gates are utilized to accomplish this. The project provides insight into managing bit constraints in digital logic, particularly in implementing arithmetic operations like multiplication and addition within defined bit limits. Overall, this project demonstrates the capability of digital design to solve mathematical problems.

3. Components Used

The circuit design uses several key components to implement matrix multiplication in Logisim. The following components are used,



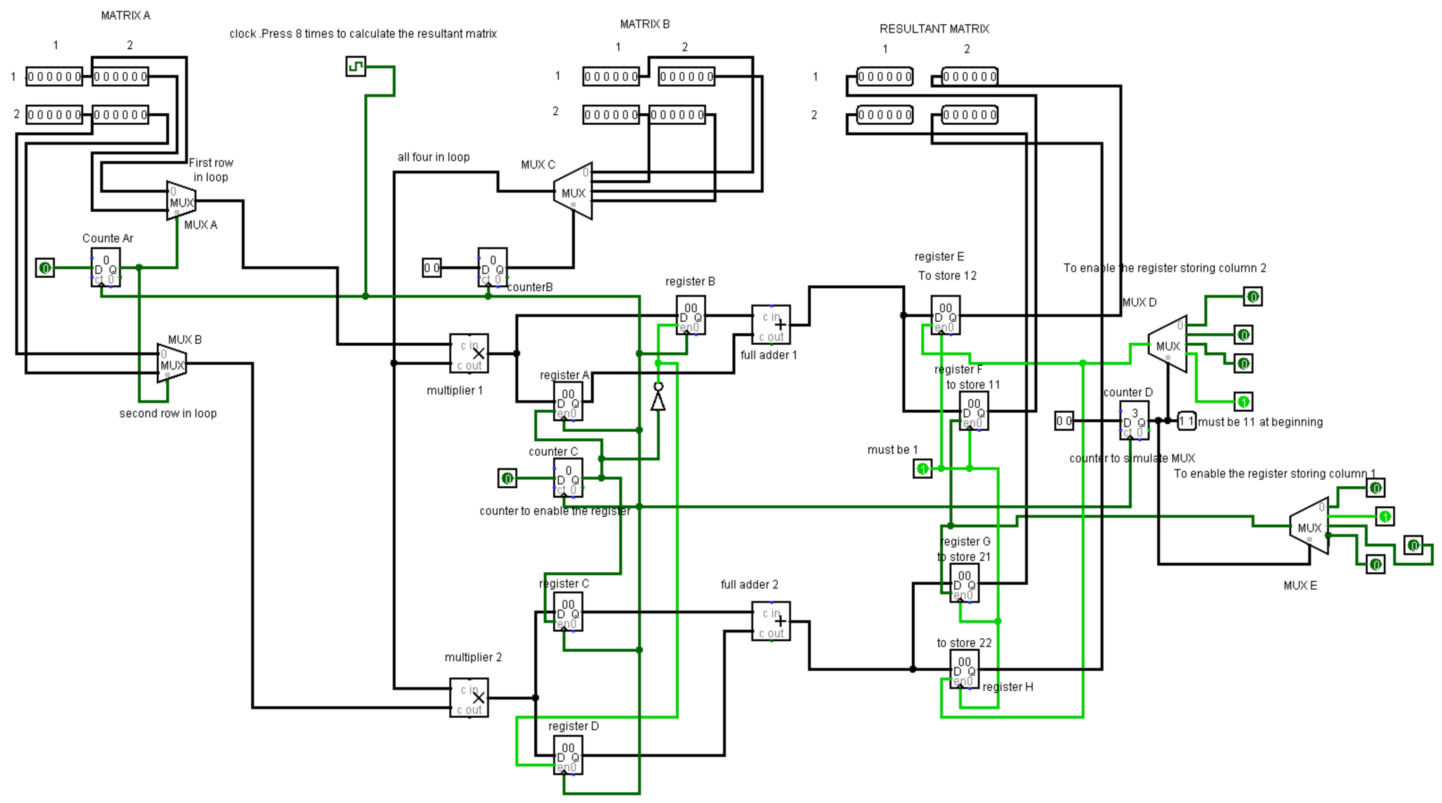
Component	Library	Simple	Unique	Recursive
Pin	Wiring	26	26	26
Clock	Wiring	1	1	1
NOT Gate	Gates	1	1	1
Multiplexer	Plexers	5	5	5
Adder	Arithmetic	2	2	2
Multiplier	Arithmetic	2	2	2
Register	Memory	8	8	8
Counter	Memory	4	4	4
Label	Base	23	23	23
TOTAL (without project's subcircuits)		72	72	72
TOTAL (with subcircuits)		72	72	72

- **Pin**: Provides input or output points for data signals in the circuit.
- **Clock**: Generates a periodic signal to synchronize sequential circuit operations.
- **NOT Gate**: Inverts the input signal, producing a logic high (1) if the input is low (0), and vice versa.
- **Multiplexer**: Selects one of multiple input signals to pass through based on selector inputs.
- **Adder**: Performs binary addition on input values, generating a sum and a carry-out.
- **Multiplier**: Computes the product of two binary numbers.
- **Register**: Stores binary data temporarily, allowing data to be read or written on clock cycles.
- **Counter**: Sequentially counts in binary, typically incrementing or decrementing with each clock pulse.

- **Label**: Adds text annotations to help identify and organize parts of the circuit diagram.

Each component was chosen to facilitate accurate multiplication and addition while maintaining the constraint of a 6-bit resultant value.

4.Circuit Diagram and Explanation



- **Matrix A and Matrix B Inputs**:

  These are the inputs for the 2 2x2 matrices that will be multiplied. Each matrix has two rows and columns, with binary values as elements. Each detail can be in my view set for testing distinctive matrix values.

- **Clock :**

  The clock signal (indicated at the top center) is used to synchronize the operation of the circuit. The counter works with the clock, iterating through the elements of the matrices in a loop.

  You need to press the clock 8 times(4 clock cycles.One press to make it high another to make it low again) to complete the computation of the resultant matrix, as each clock pulse helps in moving through each multiplication and addition step required for the matrix product.

- **Counters (Counter A, Counter B, Counter C, Counter D)**:

  Counter A is used to control the row choice for Matrix A, cycling between the primary and 2d rows all through each multiplication step.

  Counter B manages column selection in Matrix B, iterating thru the columns for the multiplication technique.

  Counter C permits the storage registers sequentially, allowing accurate timing for the storage of intermediate and final consequences.

  Counter D is used to simulate the selection technique in MUX D and MUX E, aiding in accurate routing of statistics to the correct garage registers.

- **Multiplexers (MUX A, MUX B, MUX C, MUX D, MUX E)**:

  MUX A and MUX B are used to select specific rows in Matrix A, with Counter A controlling which row is active for each clock cycle.

  MUX C selects elements from Matrix B based on Counter B’s state, ensuring the right column is accessed for each multiplication.

  MUX D and MUX E control the enabling of registers for storing results in specific positions of the resultant matrix. They route the final values into the correct storage location.

- **Multipliers (Multiplier 1 and Multiplier 2)**:

  Multiplier 1 performs element-wise multiplication of elements from the first row of Matrix A and all elements of Matrix B to produce partial products.

  Multiplier 2 performs element-wise multiplication of elements from the second row of Matrix A and all elements of Matrix B to produce partial products.

.• **Adders (Full Adder 1 and Full Adder 2)**:

  Full Adder 1 adds the partial products from Register A and Register B to produce the sum needed for one element of the resultant matrix.

  Full Adder 2 performs a similar operation for the partial products in Register C and Register D.

- **Registers**:

  Registers temporarily store intermediate results (like individual products and sums) as the circuit progresses through each multiplication and addition. This allows the circuit to accumulate results correctly without losing any data at each clock cycle.

Register A:

- Temporarily stores the first partial product from **Multiplier 1** for each element in the resultant matrix.
- Holds the intermediate multiplication result of the first row of **Matrix A** and the first column of **Matrix B** .
- This value is later sent to **Full Adder 1** to be added with the second partial product.

Register B:

- Temporarily stores the second partial product from **Multiplier 1** for each element in the resultant matrix.
- Holds the intermediate multiplication result of the second row of **Matrix A** and the second column of **Matrix B**.
- This value is added with the partial product from **Register A** in **Full Adder 1**.

Register C:

- Similar to Register A, **Register C** stores the first partial product from **Multiplier 2** for each element in the resultant matrix.
- Holds the intermediate multiplication result of the second d row of **Matrix A** and the first column of **Matrix B**.
- This value is later sent to **Full Adder 2** to be added with the second partial product for the element in the resultant matrix.

Register D:

- Similar to Register B, **Register D** stores the second partial product from **Multiplier 2** for each element in the resultant matrix.
- Holds the intermediate multiplication result of the second row of **Matrix A** and the second column of **Matrix B**.
- This value is added with the partial product from **Register C** in **Full Adder 2**.

Register E:

- Stores the final computed value of the top-left element (position [1,1]) in the resultant matrix.
- Receives the summed output from **Full Adder 1** for this position after all necessary multiplications and additions are complete.

Register F:

- Stores the final computed value of the top-right element (position [1,2]) in the resultant matrix.
- Receives the summed output from **Full Adder 1** for this position after the required multiplications and additions.

Register G:

- Stores the final computed value of the bottom-left element (position [2,1]) in the resultant matrix.
- Receives the summed output from **Full Adder 2** for this position after the necessary multiplications and additions.

Register H:

- Stores the final computed value of the bottom-right element (position [2,2]) in the resultant matrix.
- Receives the summed output from **Full Adder 2** for this position once all multiplications and additions are complete.

Registers A, B, C, and D are used for temporary storage of partial products and intermediate results during the multiplication process.

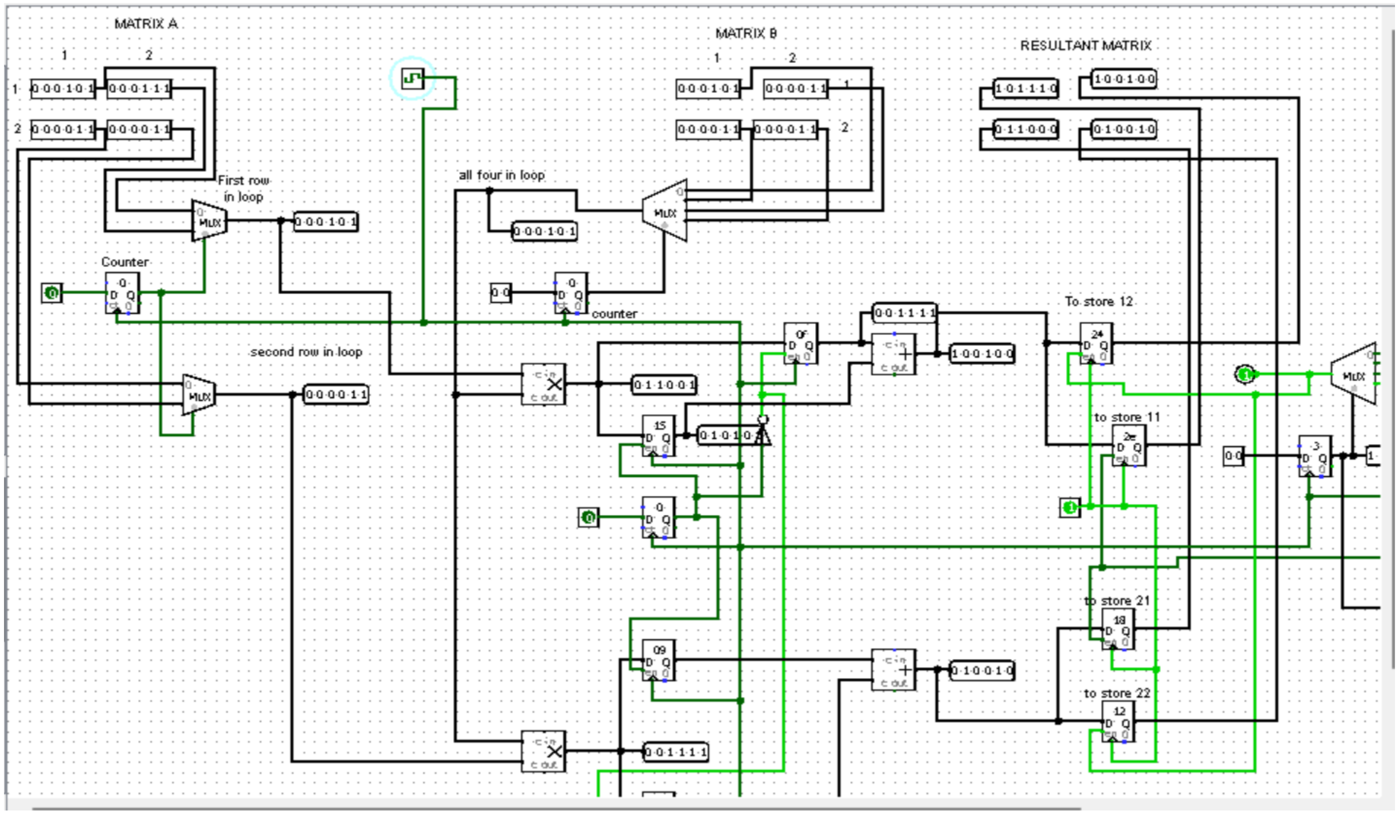
Registers E, F, G, and H hold the final values of each element in the resultant 2x2 matrix after all computations are finished.

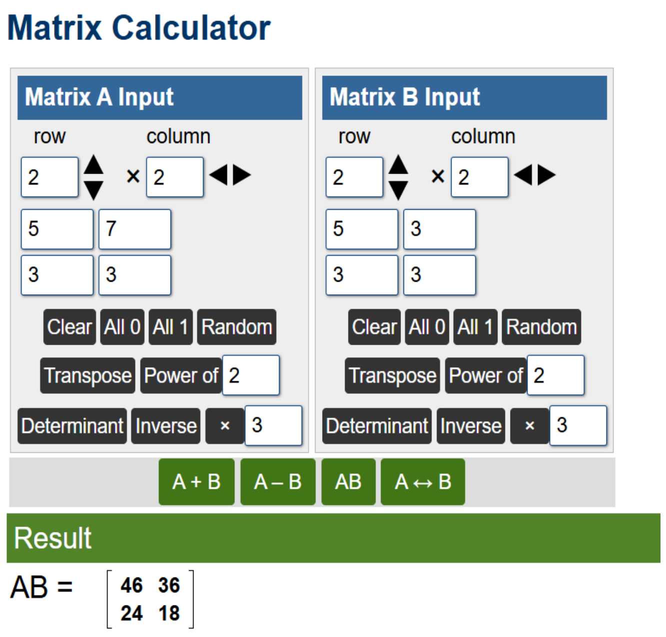
- **Resultant Matrix**:

  This section displays the computed elements of the 2x2 result matrix after multiplication and addition.

Each result is stored in one of the registers (E, F, G, or H), corresponding to the positions in the 2x2 output matrix.

5.Output & Truthtable :





Matrix Calculator

Matrix A Input

row 2 × column 2

5	7
3	3

Matrix B Input

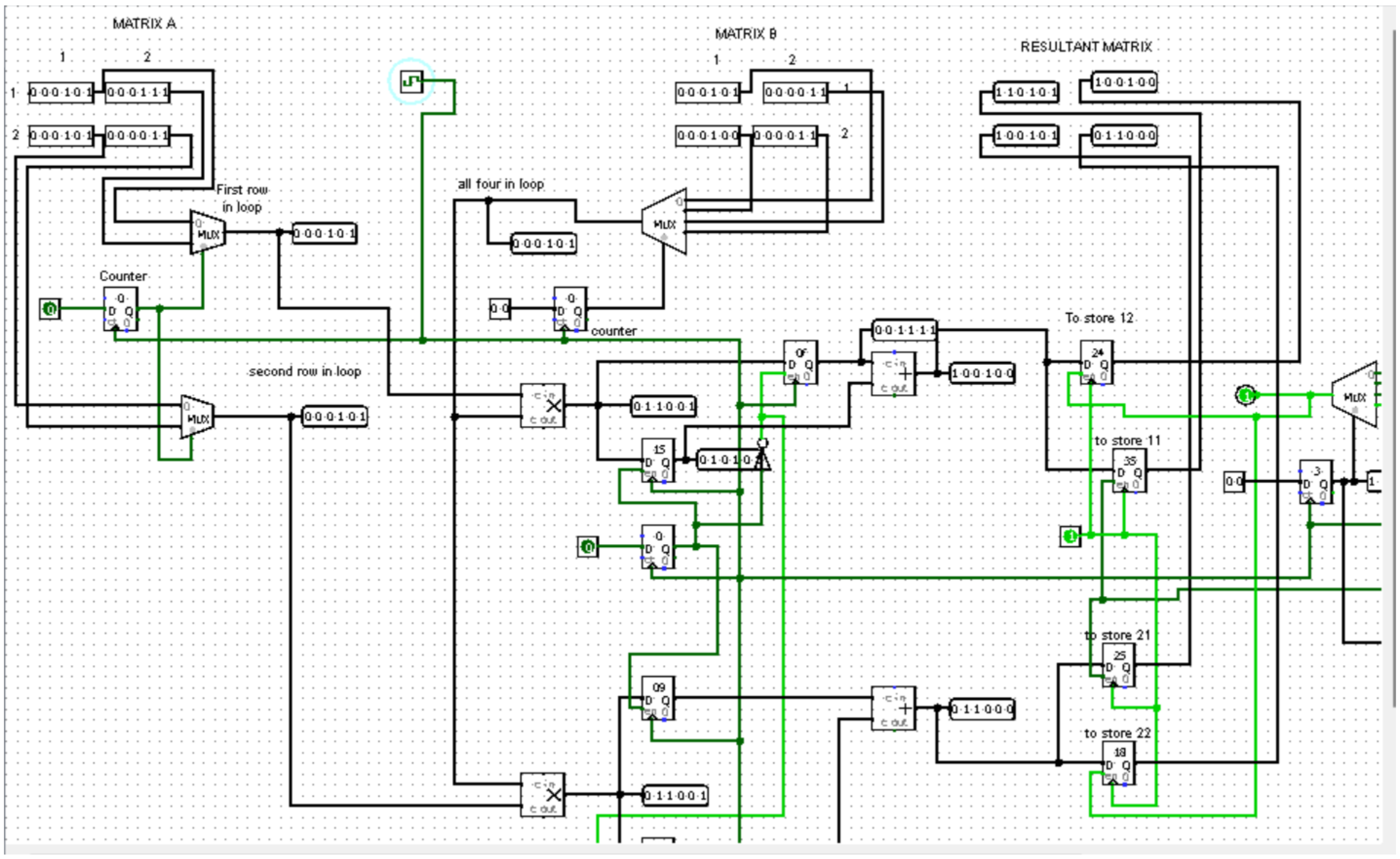
row 2 × column 2

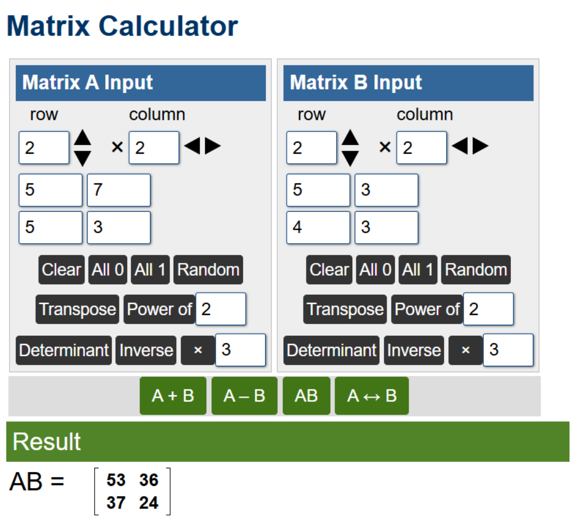
5	3
3	3

Result

$$AB = \begin{bmatrix} 46 & 36 \\ 24 & 18 \end{bmatrix}$$

Decimal	Binary
46	101110
36	100100
24	011000
18	010010





Matrix Calculator

Matrix A Input

row column
2 × 2

5	7
5	3

Clear | All 0 | All 1 | Random
Transpose | Power of 2
Determinant | Inverse | × 3

Matrix B Input

row column
2 × 2

5	3
4	3

Clear | All 0 | All 1 | Random
Transpose | Power of 2
Determinant | Inverse | × 3

A + B | A – B | AB | A ↔ B

Result

$$AB = \begin{bmatrix} 53 & 36 \\ 37 & 24 \end{bmatrix}$$

Decimal	Binary
53	110101
36	100100
37	100101
24	011000

(NOTE: the output screenshots does not cover the whole circuit rather only cover input and output areas)

Here is a truthtable of how the output is computed for every clock cycle

	1	2	3	4	5	6	7	8	9	10	11
Clk	MUX A O/P	MUX B O/P	MUX C O/P	Register A {1*2}	Register B {1*2}	Register F R[11] {4+5}	Register E R[12] {4+5}	RegisterC {2*3}	Register D {2*3}	Register G R[21] {8+9}	Register F R[22] {8*9}
0	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx	xxxxxx
1	000101	000101	000101	011001 (en=1)	xxxxxx (en=0)	xxxxxx (en=0)	xxxxxx (en=0)	011001 (en=1)	xxxxxx (en=0)	xxxxxx (en=0)	xxxxxx (en=0)
2	000111	000011	000100	011001 (en=0)	011100 (en=1)	110101 (en=1)	xxxxxx (en=0)	011001 (en=0)	001100 (en=1)	100101 (en=1)	xxxxxx (en=0)
3	000101	000101	000011	011001 (en=1)	011100 (en=0)	110101 (en=0)	xxxxxx (en=0)	001111 (en=1)	001100 (en=0)	100101 (en=0)	xxxxxx (en=0)
4	000111	000011	000011	011001 (en=0)	011000 (en=1)	110101 (en=0)	100100 (en=1)	001111 (en=0)	001001 (en = 1)	100101 (en=0)	011000 (en=1)

Here the enable (en) value of 0 and 1 is switched with the help of counters and multiplexers (MUX) depending on the requirement.

6.OUTPUT ANALYSIS

Matrix A:

- a11=000101 (5)
- a12=000111 (7)
- a21=000101 (5)
- a22=000011 (3)

Matrix B:

- b11=000101 (5)
- b12=000011 (3)
- b21=000100 (4)
- b22=000011 (3)

Resultant Matrix C (Target Calculation)

To calculate each element of the resultant matrix **C**, we use the matrix multiplication formula for 2x2 matrices as discussed before.

Step-by-Step Calculation with Binary Values

1. **Calculate c11:**
   - Formula: c11=a11·b11+a12·b21
   - Substituting values:
     - a11·b11=000101 (5) · 000101 (5) = 011001 (25)
     - a12·b21=000111 (7) · 000100 (4) = 011100 (28)
   - Add the results:
     - c11=011001 (25) + 011100 (28) = 110101 (53)
2. **Calculate c12:**
   - Formula: c12=a11·b12+a12·b22
   - Substituting values:
     - a11·b12=000101 (5) · 000011 (3) = 001111 (15)
     - a12·b22=000111 (7) · 000011 (3) = 010101 (21)
   - Add the results:
     - c12=001111 (15) + 010101 (21) = 100100 (36)
3. **Calculate c21:**
   - Formula: c21=a21·b11+a22·b21
   - Substituting values:
     - a21·b11=000101 (5) · 000101 (5) = 011001 (25)
     - a22·b21=000011 (3) · 000100 (4) = 001100 (12)
   - Add the results:
     - c21=011001 (25) +001100 (12) =100101 (37)
4. **Calculate c22:**
   - Formula: c22=a21·b12+a22·b22
   - Substituting values:
     - a21·b12=000101 (5) · 000011 (3) = 001111 (15)
     - a22·b22=000101 (3) · 000011 (3) = 001001 (9)
   - Add the results:
     - c22=001111 (15) + 001001 (9) = 011000 (24)

Summary of Each Output Register

Each of the registers in the circuit would store the following binary values:

- **Register E** (for c11): `110101` (binary for 53)
- **Register F** (for c12): `100100` (binary for 36)
- **Register G** (for c21): `100101` (binary for 37)
- **Register H** (for c22):`011000` (binary for 24)

Each value in the registers corresponds to the elements of the resultant matrix after performing the multiplication and addition operations as outlined in the circuit's logic. This breakdown shows how the circuit completes the matrix multiplication operation step-by-step based on the given inputs.

7.LIMITATIONS:

- The resultant matrix elements are limited to a maximum of 6 bits, restricting the range of values that can be represented.
- The circuit is unable to handle negative numbers, limiting its application to unsigned integer matrices..
- Requires manual initialization and configuration before performing calculations, which may reduce efficiency and scalability.

8.LEARNING OUTCOMES:

- **Binary Arithmetic in Hardware**: Learned to implement matrix multiplication using binary multiplication and addition in circuits.

- **Sequential Logic and Clock Cycles**: Gained understanding of step-by-step operations controlled by clock cycles.

- **Data Flow Control**: Used multiplexers and counters to manage and control data flow through the circuit.

- **Component Integration**: Experienced in linking adders, registers, and multiplexers for functionality.

- **Debugging and Verification**: Developed skills in systematic troubleshooting by examining individual outputs in each step.

- **Looping Operations in Hardware**: Learned to handle iterative processes in hardware with counters and looping controls.

- **Simulation Tools Proficiency**: Acquired hands-on experience with Logisim for circuit design and testing.