4-Bit Divider Using Repeated Subtraction Algorithm

Group Members: Arnold Joseph C. Najera Jr.
Ike Kian G. Abiera
Lev Altair Imetri S. Aguirre

Course: CPE 2301 - LOGIC CIRCUITS AND DESIGN

Schedule: MTWThFS 10:30AM-1:30PM

Table of Contents

1 Design Description

This project shows a 4-bit digital divider that calculates the quotient and remainder from two binary numbers. It follows the rules of binary division by repeatedly subtracting the divisor from the dividend and shifting bits as needed. The circuit keeps track of how many times the divisor fits into the dividend and what value is left over.

2 Scope and Limitations

This 4-bit divider circuit is for unsigned 4-bit numbers so both the dividend and divisor must be whole numbers from 0 to 15. The main function of the circuit is to output two values: the quotient which is the result of the division and the remainder which is the leftover value. It's good for demonstrating how binary division works in digital systems so it's a great educational tool for students learning digital logic and computer architecture. However the circuit has some limitations. It can't handle negative numbers or signed values, it doesn't support decimal or floating point results so only whole number outputs are possible. It also doesn't have a mechanism to handle division by zero so it will produce incorrect or undefined results if the divisor is zero. The fixed 4-bit design also limits the range of numbers it can process so it's not suitable for more advanced or real world applications. But it does introduce the basic principles of digital division and helps build a strong foundation in binary arithmetic.

3 System Operation

3.1 List of Materials

- 74LS157 Quad Multiplexer IC
- 74LS194 4 Bit Shift Register IC
- 74LS83 Binary Full Adder IC
- 74LS93 4 Bit Binary Counter IC
- 74LS48 BCD to 7 Segment Decoder IC
- 74LS04 NOT Gate
- 74LS08 AND Gate
- 74LS32 OR Gate
- Red LEDs
- 100 Ohm Resistors
- 100 Hz Clock
- 5V Power

3.2 Truth Table

Dividend				Divisor				Quotient			
A3	A2	A1	A0	B3	B2	B1	B0	Q3	Q2	Q1	Q0
x	x	x	x	0	0	0	0	Invalid			
0	0	0	0	x	x	x	x	0	0	0	0
0	0	0	1	0	0	0	1	0	0	0	1
0	0	0	1	x	x	x	x	0	0	0	0
0	0	1	0	0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	0	0	0	0	1
0	0	1	0	x	x	x	x	0	0	0	0
0	0	1	1	0	0	0	1	0	0	1	1
0	0	1	1	0	0	1	0	0	0	0	1
0	0	1	1	0	0	1	1	0	0	0	1
0	0	1	1	x	x	x	x	0	0	0	0
0	1	0	0	0	0	0	1	0	1	0	0
0	1	0	0	0	0	1	0	0	0	1	0
0	1	0	0	0	0	1	1	0	0	0	1
0	1	0	0	0	1	0	0	0	0	0	1
0	1	0	0	x	x	x	x	0	0	0	0

0	1	0	1	0	0	0	1	0	1	0	1
0	1	0	1	0	0	1	0	0	0	1	0
0	1	0	1	0	0	1	1	0	0	0	1
0	1	0	1	0	1	0	0	0	0	0	1
0	1	0	1	0	1	0	1	0	0	0	1
0	1	0	1	x	x	x	x	0	0	0	0
0	1	1	0	0	0	0	1	0	1	1	0
0	1	1	0	0	0	1	0	0	0	1	1
0	1	1	0	0	0	1	1	0	0	1	0
0	1	1	0	0	1	0	0	0	0	0	1
0	1	1	0	0	1	0	1	0	0	0	1
0	1	1	0	0	1	1	0	0	0	0	1
0	1	1	0	x	x	x	x	0	0	0	0
0	1	1	1	0	0	0	1	0	1	1	1
0	1	1	1	0	0	1	0	0	0	1	1
0	1	1	1	0	0	1	1	0	0	1	0
0	1	1	1	0	1	0	0	0	0	0	1
0	1	1	1	0	1	0	1	0	0	0	1
0	1	1	1	0	1	1	0	0	0	0	1

0	1	1	1	0	1	1	1	0	0	0	1
0	1	1	1	x	x	x	x	0	0	0	0
1	0	0	0	0	0	0	1	1	0	0	0
1	0	0	0	0	0	1	0	0	1	0	0
1	0	0	0	0	0	1	1	0	0	1	0
1	0	0	0	0	1	0	0	0	0	1	0
1	0	0	0	0	1	0	1	0	0	0	1
1	0	0	0	0	1	1	0	0	0	0	1
1	0	0	0	0	1	1	1	0	0	0	1
1	0	0	0	1	0	0	0	0	0	0	1
1	0	0	0	x	x	x	x	0	0	0	0
1	0	0	1	0	0	0	1	1	0	0	1
1	0	0	1	0	0	1	0	0	1	0	0
1	0	0	1	0	0	1	1	0	0	1	1
1	0	0	1	0	1	0	0	0	0	1	0
1	0	0	1	0	1	0	1	0	0	0	1
1	0	0	1	0	1	1	0	0	0	0	1
1	0	0	1	0	1	1	1	0	0	0	1
1	0	0	1	1	0	0	0	0	0	0	1

1	0	0	1	1	0	0	1	0	0	0	1
1	0	0	1	x	x	x	x	0	0	0	0
1	0	1	0	0	0	0	1	1	0	1	0
1	0	1	0	0	0	1	0	0	1	0	1
1	0	1	0	0	0	1	1	0	0	1	1
1	0	1	0	0	1	0	0	0	0	1	0
1	0	1	0	0	1	0	1	0	0	1	0
1	0	1	0	0	1	1	0	0	0	0	1
1	0	1	0	0	1	1	1	0	0	0	1
1	0	1	0	1	0	0	0	0	0	0	1
1	0	1	0	1	0	0	1	0	0	0	1
1	0	1	0	1	0	1	0	0	0	0	1
1	0	1	0	x	x	x	x	0	0	0	0
1	0	1	1	0	0	0	1	1	0	1	1
1	0	1	1	0	0	1	0	0	1	0	1
1	0	1	1	0	0	1	1	0	0	1	1
1	0	1	1	0	1	0	0	0	0	1	0
1	0	1	1	0	1	0	1	0	0	1	0
1	0	1	1	0	1	1	0	0	0	0	1

1	0	1	1	0	1	1	1	0	0	0	1
1	0	1	1	1	0	0	0	0	0	0	1
1	0	1	1	1	0	0	1	0	0	0	1
1	0	1	1	1	0	1	0	0	0	0	1
1	0	1	1	1	0	1	1	0	0	0	1
1	0	1	1	x	x	x	x	0	0	0	0
1	1	0	0	0	0	0	1	1	1	0	0
1	1	0	0	0	0	1	0	0	1	1	0
1	1	0	0	0	0	1	1	0	1	0	0
1	1	0	0	0	1	0	0	0	0	1	1
1	1	0	0	0	1	0	1	0	0	1	0
1	1	0	0	0	1	1	0	0	0	1	0
1	1	0	0	0	1	1	1	0	0	0	1
1	1	0	0	1	0	0	0	0	0	0	1
1	1	0	0	1	0	0	1	0	0	0	1
1	1	0	0	1	0	1	0	0	0	0	1
1	1	0	0	1	0	1	1	0	0	0	1
1	1	0	0	1	1	0	0	0	0	0	1
1	1	0	0	x	x	x	x	0	0	0	0

1	1	0	1	0	0	0	1	1	1	0	1
1	1	0	1	0	0	1	0	0	1	1	0
1	1	0	1	0	0	1	1	0	1	0	0
1	1	0	1	0	1	0	0	0	0	1	1
1	1	0	1	0	1	0	1	0	0	1	0
1	1	0	1	0	1	1	0	0	0	1	0
1	1	0	1	0	1	1	1	0	0	0	1
1	1	0	1	1	0	0	0	0	0	0	1
1	1	0	1	1	0	0	1	0	0	0	1
1	1	0	1	1	0	1	0	0	0	0	1
1	1	0	1	1	0	1	1	0	0	0	1
1	1	0	1	1	1	0	0	0	0	0	1
1	1	0	1	1	1	0	1	0	0	0	1
1	1	0	1	x	x	x	x	0	0	0	0
1	1	1	0	0	0	0	1	1	1	1	0
1	1	1	0	0	0	1	0	0	1	1	1
1	1	1	0	0	0	1	1	0	1	0	0
1	1	1	0	0	1	0	0	0	0	1	1
1	1	1	0	0	1	0	1	0	0	1	0

1	1	1	0	0	1	1	0	0	0	1	0
1	1	1	0	0	1	1	1	0	0	1	0
1	1	1	0	1	0	0	0	0	0	0	1
1	1	1	0	1	0	0	1	0	0	0	1
1	1	1	0	1	0	1	0	0	0	0	1
1	1	1	0	1	0	1	1	0	0	0	1
1	1	1	0	1	1	0	0	0	0	0	1
1	1	1	0	1	1	0	1	0	0	0	1
1	1	1	0	1	1	1	0	0	0	0	1
1	1	1	0	x	x	x	x	0	0	0	0
1	1	1	1	0	0	0	1	1	1	1	1
1	1	1	1	0	0	1	0	0	1	1	1
1	1	1	1	0	0	1	1	0	1	0	1
1	1	1	1	0	1	0	0	0	0	1	1
1	1	1	1	0	1	0	1	0	0	1	1
1	1	1	1	0	1	1	0	0	0	1	0
1	1	1	1	0	1	1	1	0	0	1	0
1	1	1	1	1	0	0	0	0	0	0	1
1	1	1	1	1	0	0	1	0	0	0	1

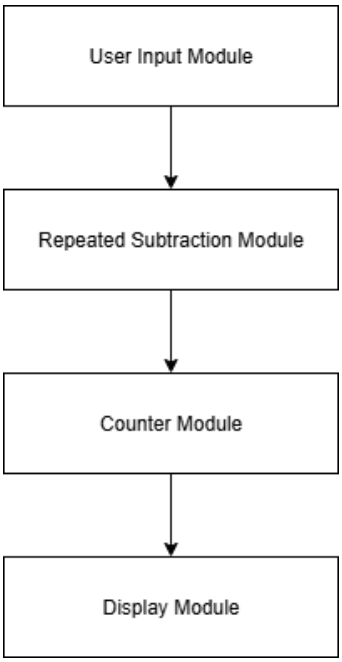
1	1	1	1	1	0	1	0	0	0	0	1
1	1	1	1	1	0	1	1	0	0	0	1
1	1	1	1	1	1	0	0	0	0	0	1
1	1	1	1	1	1	0	1	0	0	0	1
1	1	1	1	1	1	1	0	0	0	0	1
1	1	1	1	1	1	1	1	0	0	0	1

3.3 Logic Functions

- **Multiplexer (U1 - 74LS157)**:
  - Selects between inputs A and B for data routing
- **Shift Register (U2 - 74LS194)**
  - Used for loading and shifting the dividend or remainder.
- **4-bit Adder/Subtractor (U3 - 74LS83)**:
  - Performs subtraction between shifted remainder and divisor.
- **4-bit Counter (U6 - 74LS93)**
  - Used to count how many times subtraction succeeded
- **Comparator Logic Gates (AND/OR/NOT)**
  - Used to check for underflow (negative result or borrow = 1 from subtractor) to control looping or stopping subtraction.
- **BCD to 7-Segment Decoders (U34, U37 - 74LS48)**
  - For display of quotient and remainder.
- **Full Adder (U36 - 74LS83)**
  - To aid BCD conversion or output addition logic.

4 Design Layout

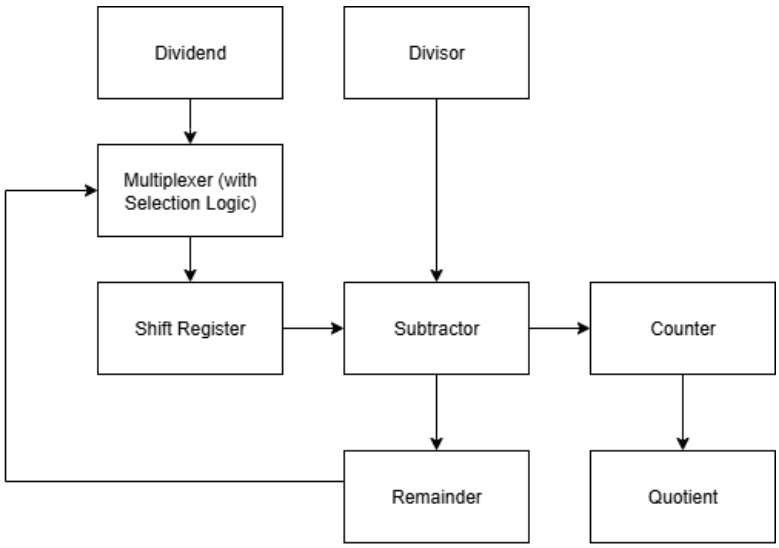
4.1 Top Level Block Diagram



Outlines the entire 4-bit divider system, the main components and how data flows through them from input, to processing, to output.

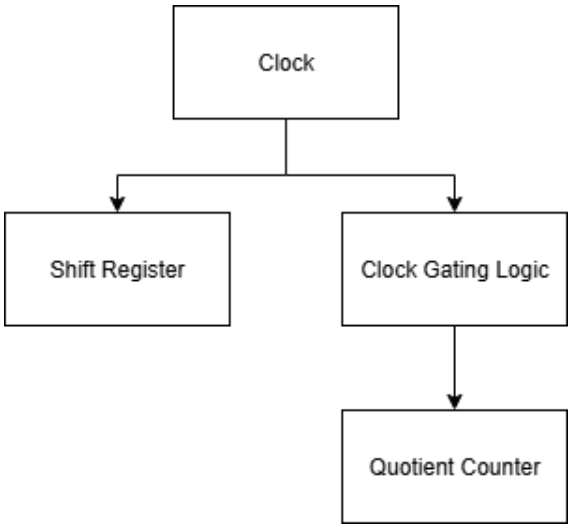
4.2 Modular Block Diagram

4.2.1 General Diagram



Breaks down the project into individual modules, each does a specific task like subtraction, shifting, or counting.

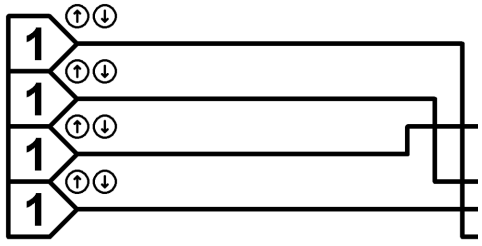
4.2.2 Clock Diagram

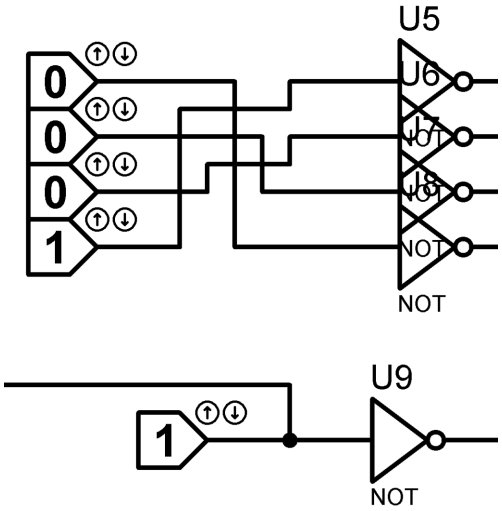


Illustrates how the clock signal controls the timing of operations across the system. Ensures that shifting and counting happens in the correct order and rhythm.

4.3 Logic/Circuit Diagrams

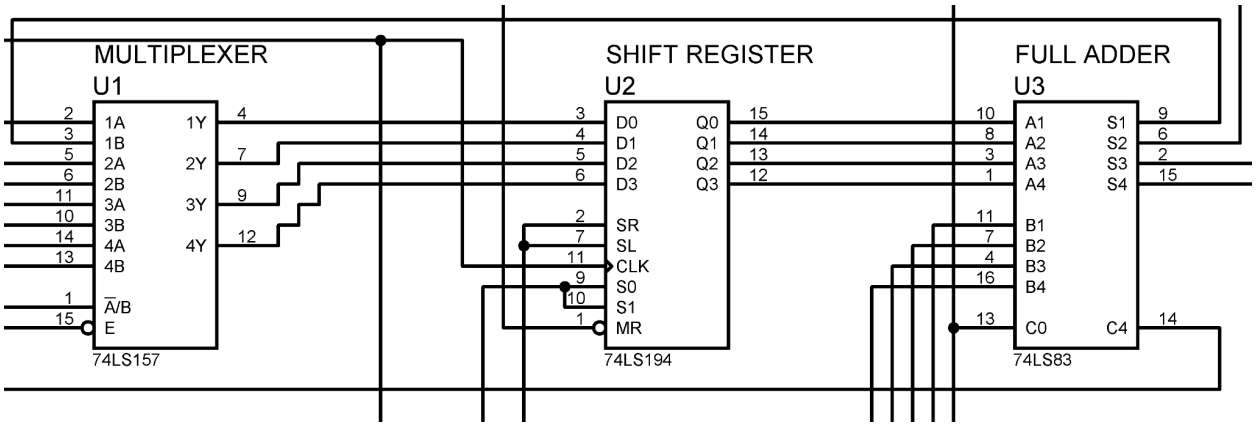
4.3.1 User Input Diagram





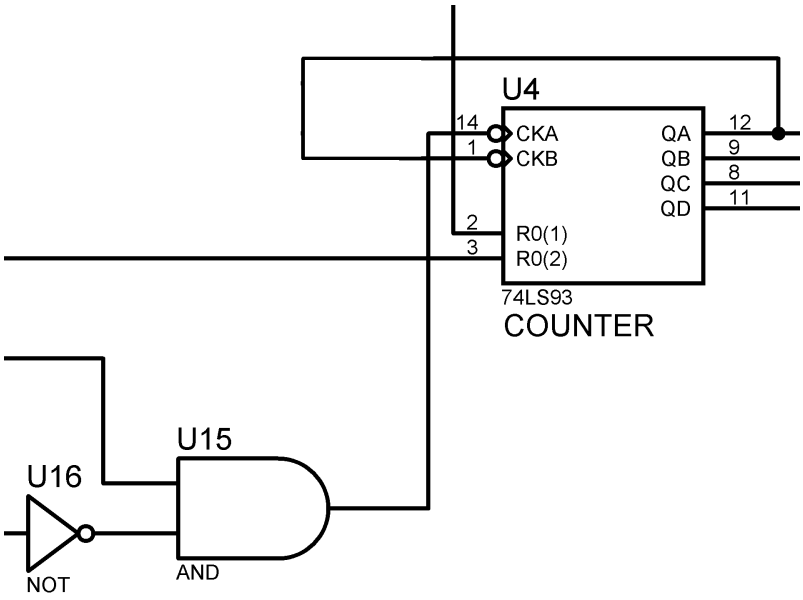
Allows the user to input 4-bit dividend, divisor and an enable/reset signal using switches. These binary values are then sent to the multiplexer and shift register for processing.

4.3.2 Repeated Subtraction Module Diagram



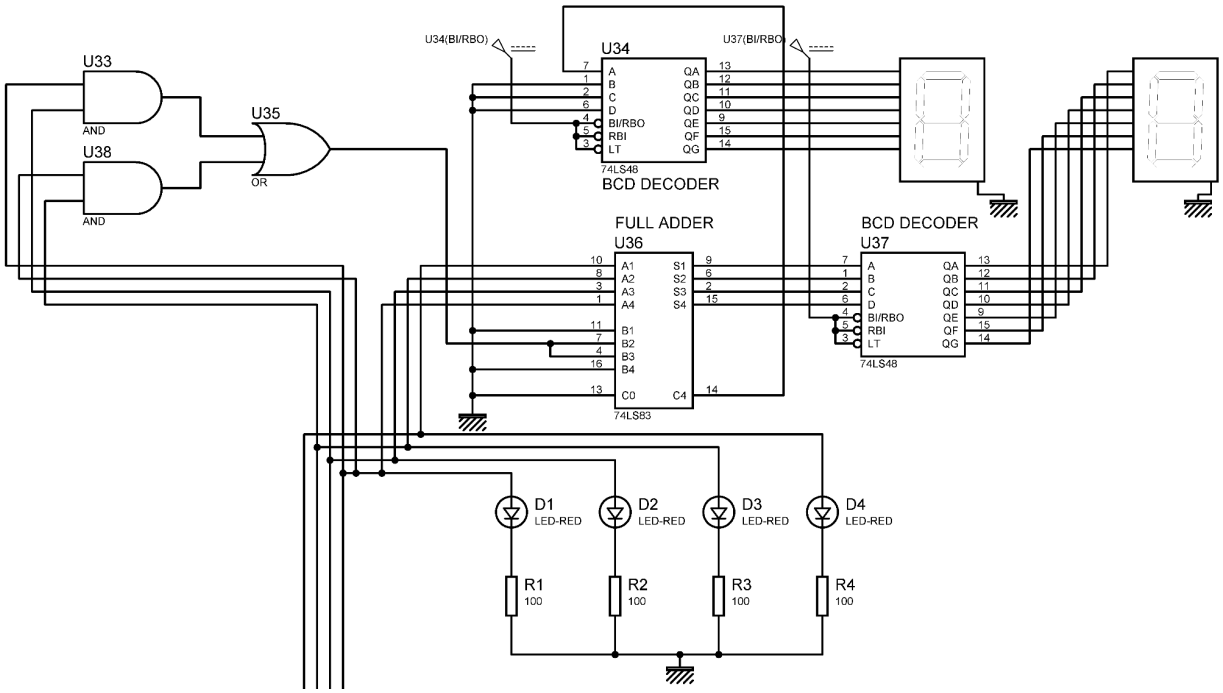
Focuses on the logic that does the main division process. Includes the subtractor, shift register and control logic. Does the core division by repeatedly subtracting the divisor from the remainder. Also determines whether to load the initial dividend or continue subtracting based on carry-out.

4.3.3 Counter Module Diagram



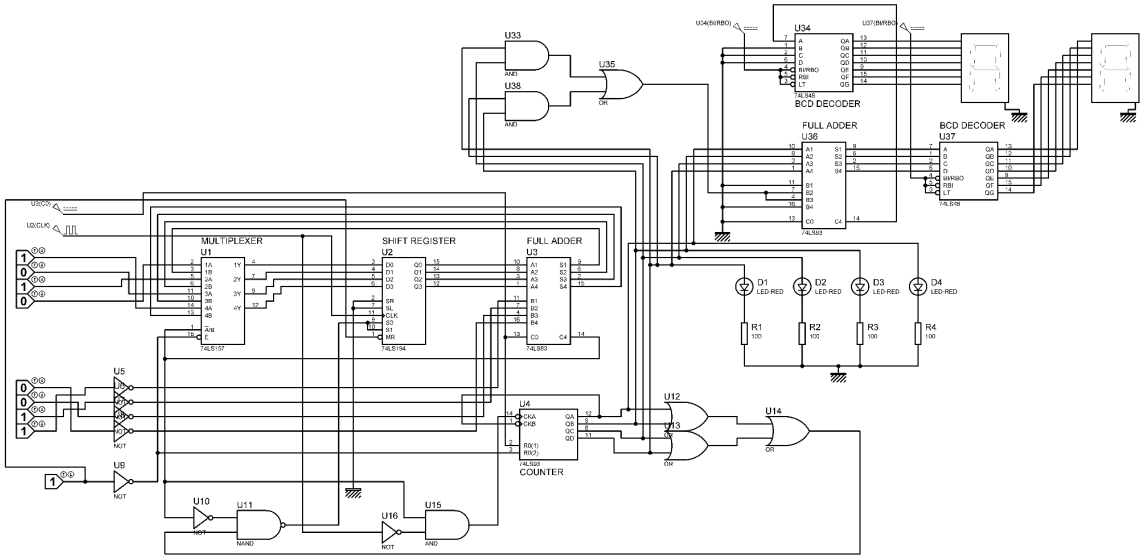
Counts the valid subtraction cycles to build the final quotient. Uses clock pulses and carry-out logic to control the count progression. This count is used to calculate the final quotient with the help of a 4-bit binary counter.

4.3.4 Display Module Diagram



Shows how the calculated quotient and remainder is displayed using 7-segment displays. Includes the conversion from binary to BCD and the use of decoders to make the output readable to the user.

4.3.5 Complete Circuit Diagram



The entire circuit does binary division by repeated subtraction of a 4-bit divisor from a 4-bit dividend. Outputs the 4-bit quotient through a counter and displays it on LEDs or 7-segment displays.

5 Test Cases and Outputs

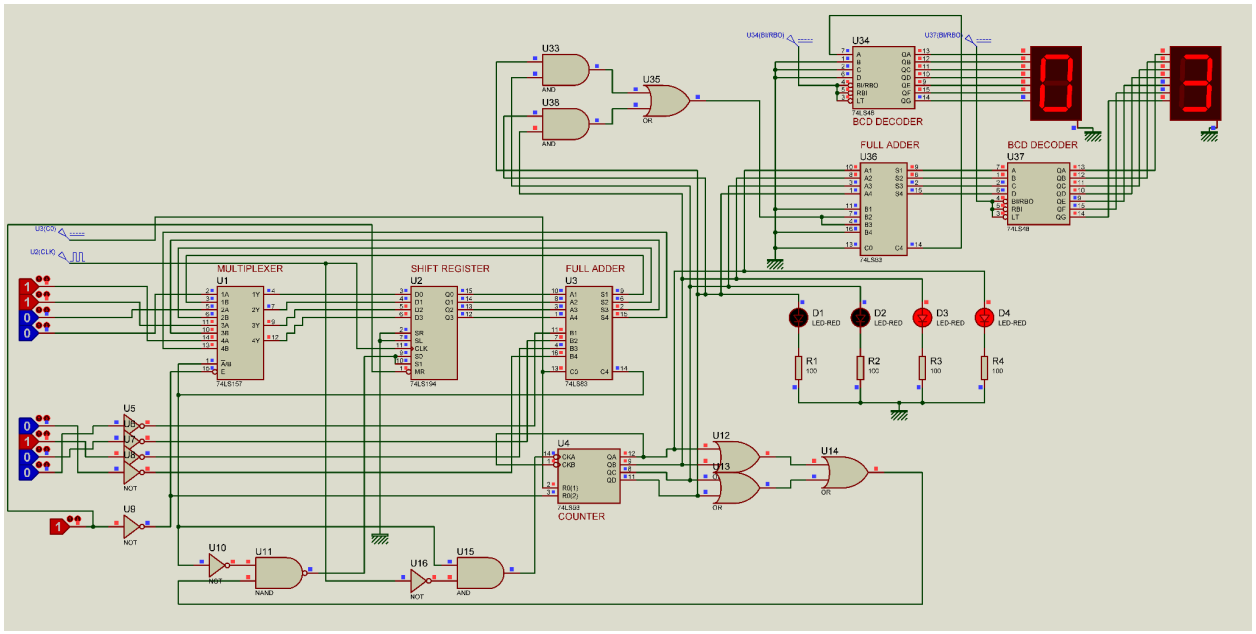
5.1 Test Cases and Outputs

5.1.1 Clean Division

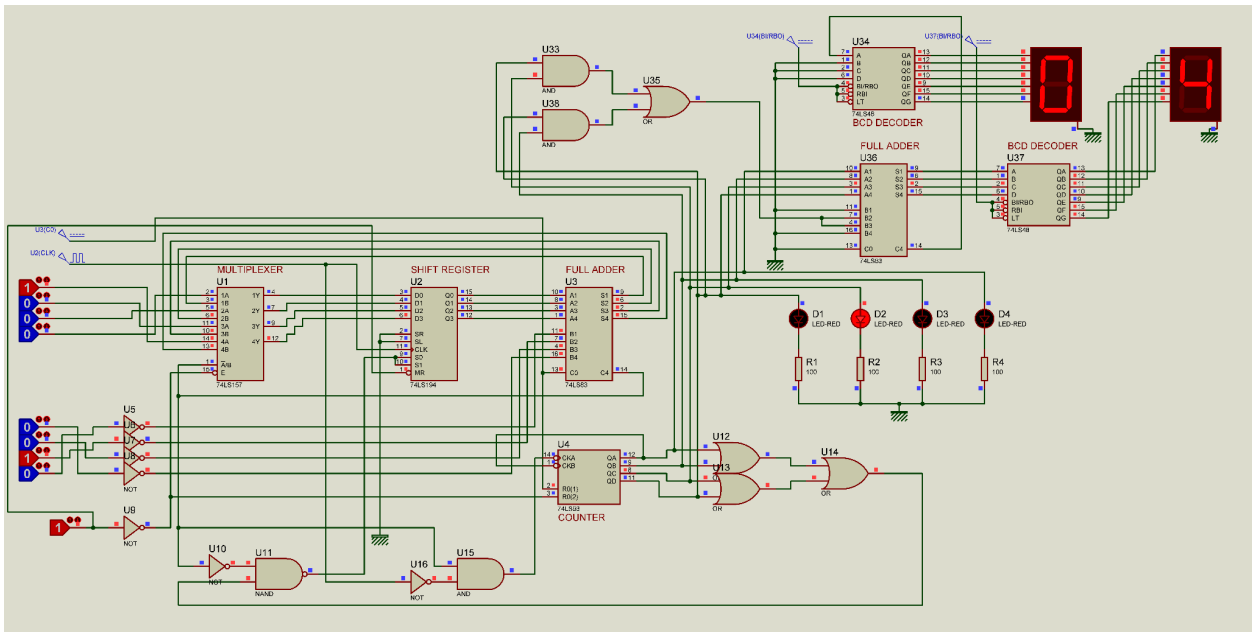
Dividend is divisible by divisor with no remainder.

Test #	Dividend	Divisor	Expected Quotient	Actual Quotient
1	1100 (12)	0100 (4)	0011 (3)	0011 (3)
2	1000 (8)	0010 (2)	0100 (4)	0100 (4)

Simulation Results:



Clean Division Test 1: 12 divided by 4



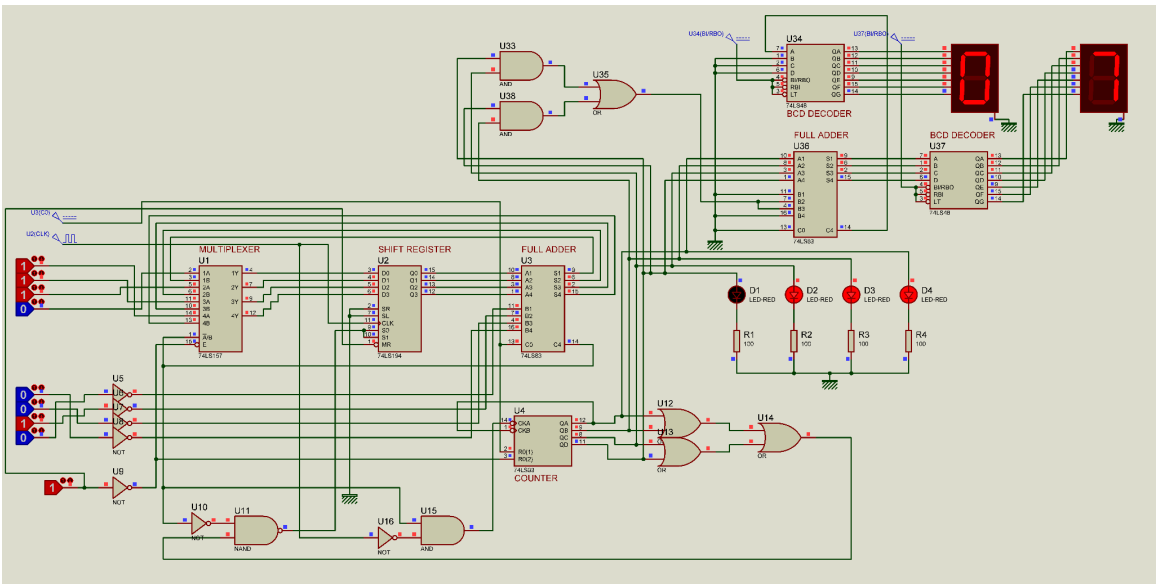
Clean Division Test 2: 8 divided by 2

5.1.2 Division with Remainder

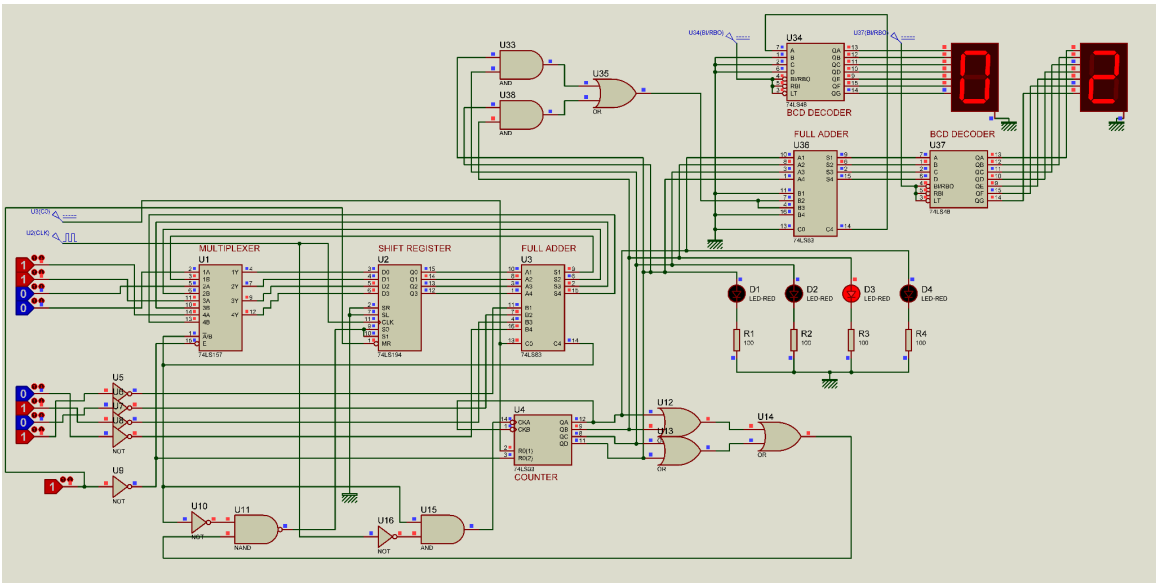
Division produces a remainder, but only quotient is shown. Correct quotient is expected

Test #	Dividend	Divisor	Expected Quotient	Actual Quotient
1	1110 (14)	0010 (2)	0111 (7)	0111 (7)
2	1100 (12)	0101 (5)	0010 (2)	0010 (2)

Simulation Results:



Division with Remainder Test 1: 14 divided by 2



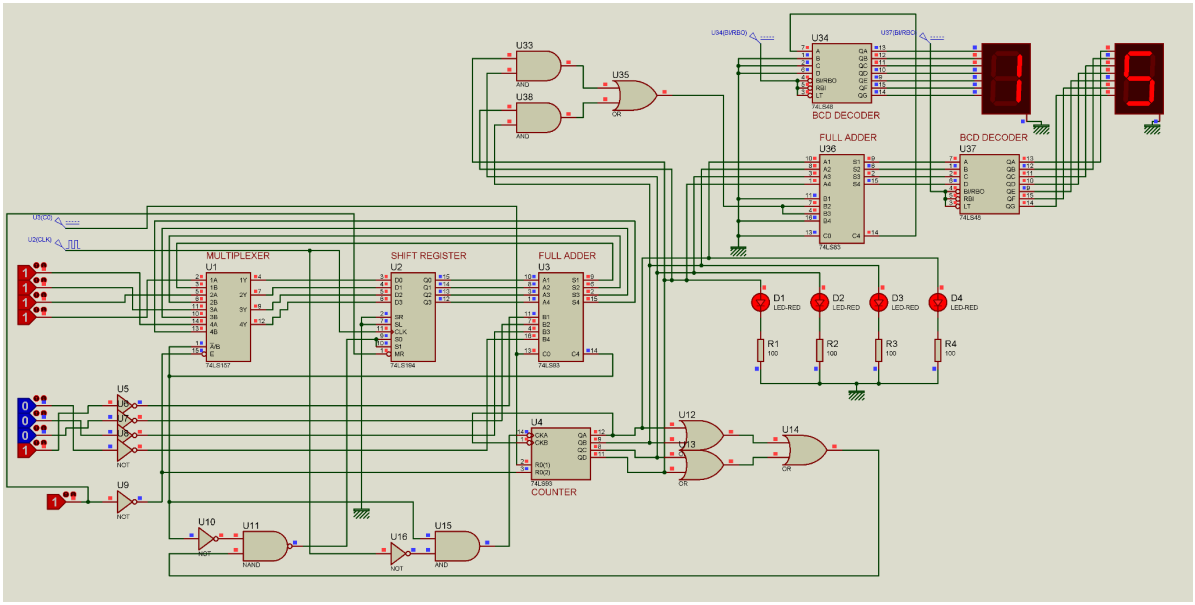
Division with Remainder Test 2: 12 divided by 5

5.1.3 Maximum Division

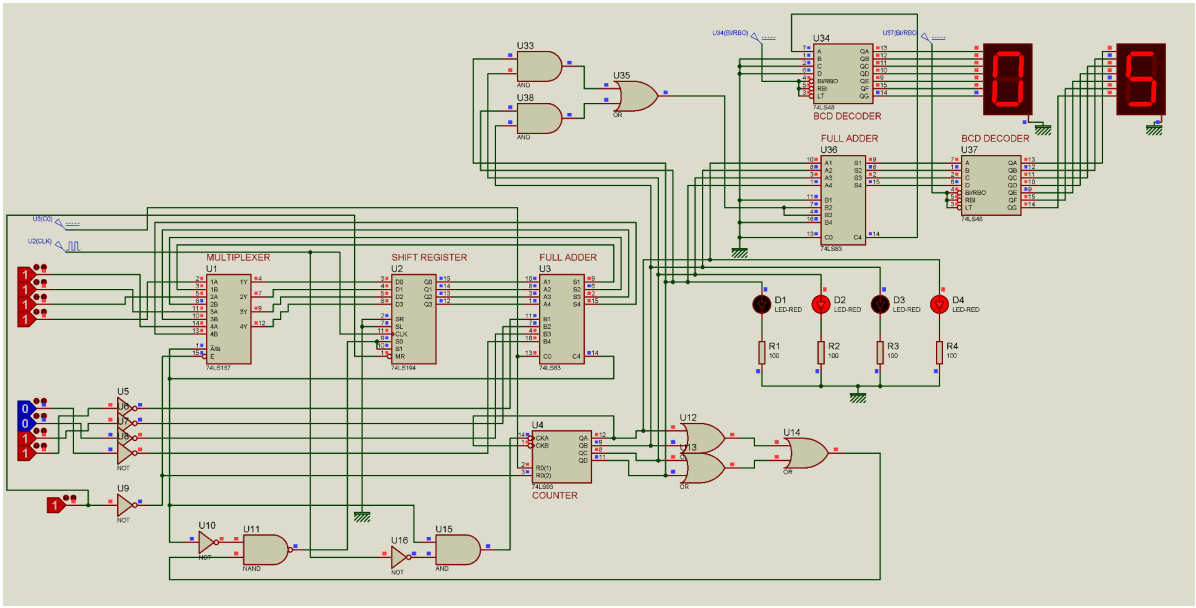
Divide the largest 4-bit number (15) by small values.

Test #	Dividend	Divisor	Expected Quotient	Actual Quotient
1	1111 (15)	0001 (1)	1111 (15)	1111 (15)
2	1111 (15)	0011 (3)	0101 (5)	0101 (5)

Simulation Results:



Maximum Division Test 1: 15 divided by 1



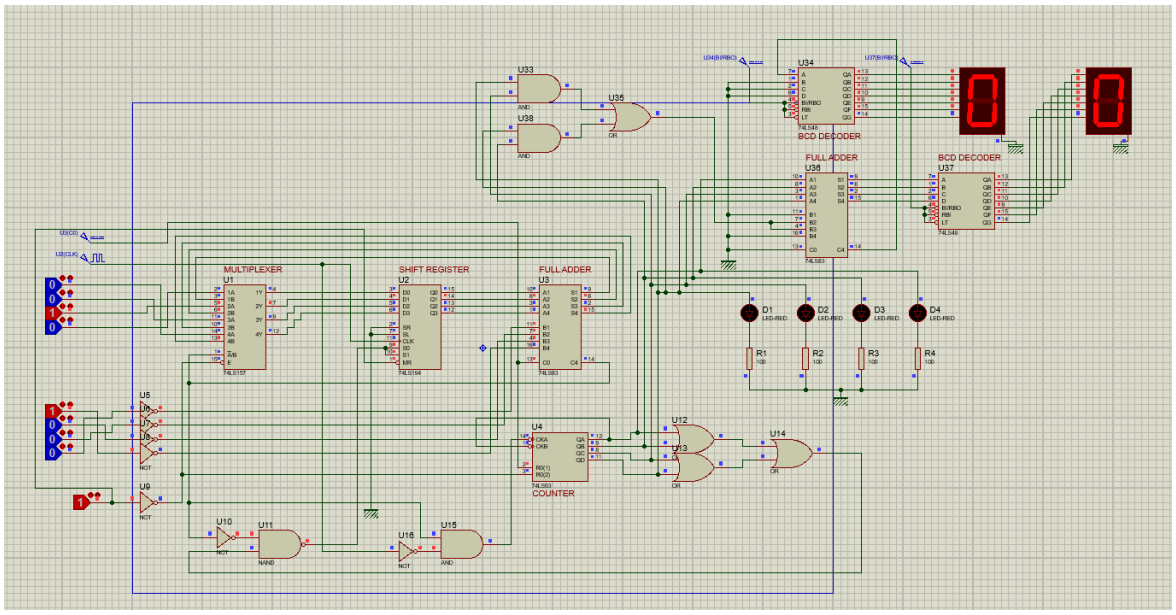
Maximum Division Test 2: 15 divided by 3

5.1.4 Divisor Larger Than Dividend

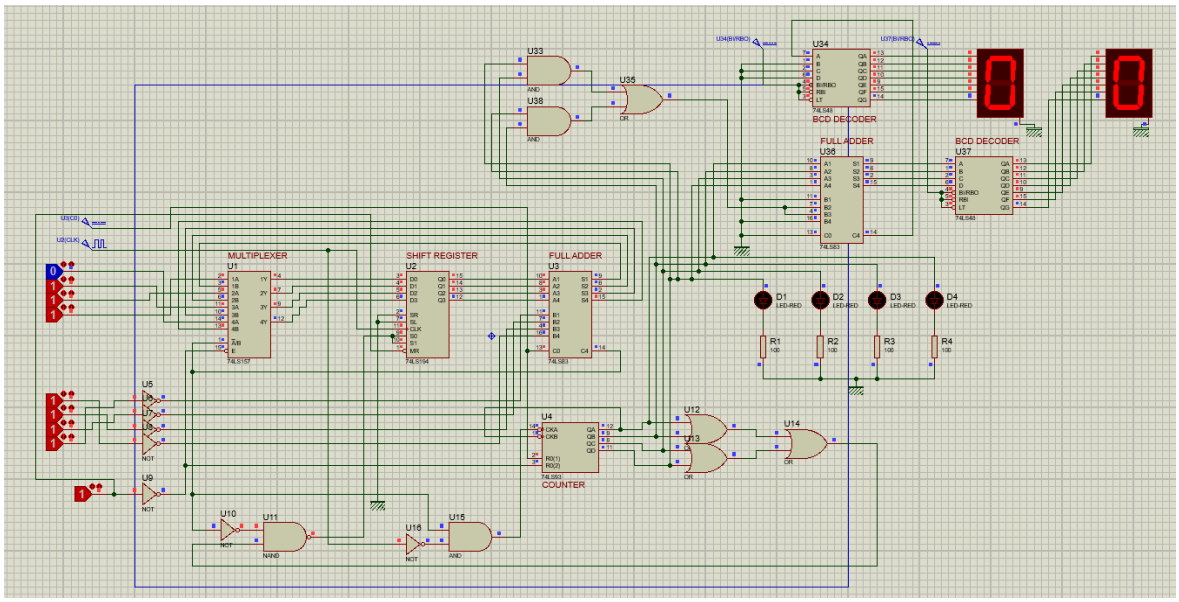
The quotient must be 0 because subtraction never happens.

Test #	Dividend	Divisor	Expected Quotient	Actual Quotient
1	0010 (2)	1000 (8)	0000 (0)	0000 (0)
2	0111 (7)	1111 (15)	0000 (0)	0000 (0)

Simulation Results:



Divisor Larger Than Dividend Test 1: 2 divided by 8



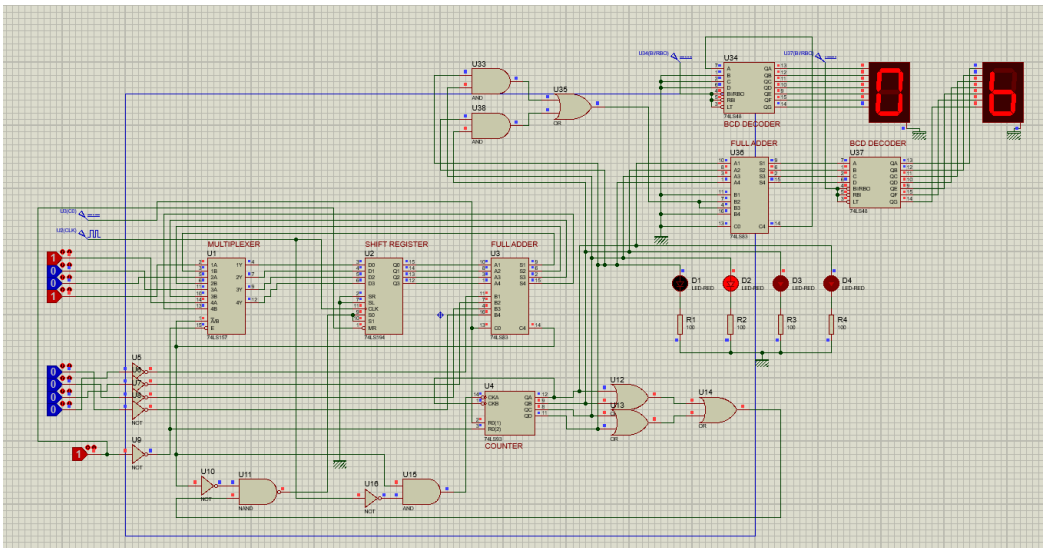
Divisor Larger Than Dividend Test 2: 7 divided by 15

5.1.5 Divide by Zero

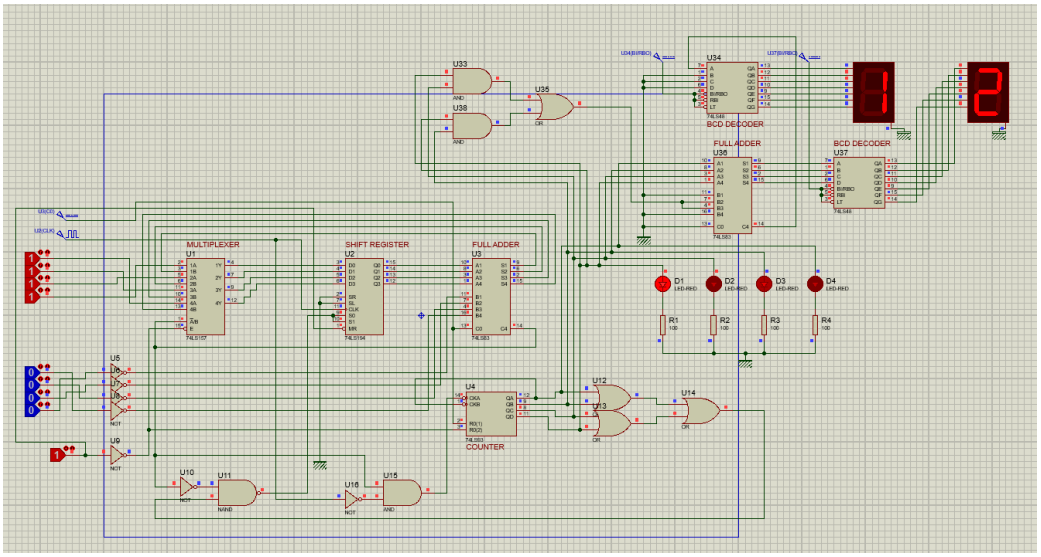
Divisor is 0, undefined; circuit may stall or give invalid output.

Test #	Dividend	Divisor	Expected Quotient	Actual Quotient
1	1001 (9)	0000 (0)	Undefined	Error
2	1111 (15)	0000 (0)	Undefined	Error

Simulation Results:



Divide by Zero Test 1: 9 divided by 0 (Note the non-correspondence of the LEDs with 7-Segment Display; Both are flashing/glitching in the simulation)



Divide by Zero Test 2: 15 divided by 0 (Note the non-correspondence of the LEDs with 7-Segment Display; Both are flashing/glitching in the simulation)

6 Conclusion and Recommendations

6.1 Conclusion

The 4-bit binary divider circuit designed and simulated in this project shows that complex arithmetic operations like division can be done using basic digital components like multiplexers, shift registers and subtractors. The system takes the dividend and remainder as inputs to a MUX, then feeds the result to a shift register that does a logical left shift on the remainder bits. After shifting the circuit subtracts the divisor from the shifted remainder. This cycle is repeated until the remainder becomes zero or negative or the 4 bits of output are filled. This algorithm allows the circuit to calculate both the quotient and the remainder of the binary division correctly.

Through this project we learned how to represent and execute binary division at hardware level. We also saw the importance of data flow control, bit level operations and timing in digital circuits. The project required careful planning of the logic sequence and connections between modules to ensure all outputs match the expected results. Overall the divider circuit simulated 4-bit division and gave us experience in digital logic design, modular circuit construction and systematic troubleshooting.

6.2 Recommendations

To improve upon this project, we recommend expanding the design to support 8-bit or higher numbers so the divider can handle larger and more practical values. This would also challenge us to scale the design and manage more complex control logic. We should also add a mechanism to handle division by zero, like a flag or error output so the user is notified when the divisor is set to 0. Currently the display glitches when the divisor is 0. A clear reset system would make repeated testing easier and avoid glitches from previous input states. A user friendly display for the quotient and remainder would make the system more interactive and easier to read.

7 Appendix

7.1 Datasheet of Materials

- 74LS157 Quad Multiplexer IC
- 74LS194 4 Bit Shift Register IC
- 74LS83 Binary Full Adder IC
- 74LS93 4 Bit Binary Counter IC
- 74LS48 BCD to 7 Segment Decoder IC
- 74LS04 NOT Gate
- 74LS08 AND Gate
- 74LS32 OR Gate

7.1 Member's Profile

Name: Arnold Najera jr
Program: BS - CpE
Student ID: 18020774
Age: 21
Address: Metropolis Subd, Pit-os, Cebu
Email: 18020774@usc.edu.ph
Phone #: 09454300031

Name: Ike Kian G. Abiera
Program: BS - CpE
Student ID: 19102013
Age: 25
Address: NMF Apartment Guizo, Mandaue
Email: 19102013@usc.edu.ph
Phone #: 09457238014

Name: Lev Altair Imetri S. Aguirre
Program: BS - CpE
Student ID: 22105327
Age: 20
Address: Albertos Drive, Talamban, Cebu
Email: 22105327@usc.edu.ph
Phone #: 09661892293