

Project 23 : Modular Adder

A Comprehensive Study of Advanced Digital Circuits

By: Abhishek Sharma, Nikunj Agrawal, Ayush Jain, Gati Goyal

Created By Team Alpha

Contents

1 Project Overview	3
2 Modular Adder	3
2.1 Description	3
2.2 Key Features of the Modular Adder:	3
2.3 RTL Code	3
2.4 Testbench	3
2.5 Simulation Results	4
3 How it works ?	4
3.1 Steps Involved in Modular Addition:	4
3.2 Basic Concept of Modular Addition:	5
3.3 Comparison with Other Adders:	5
3.4 Advantages of the Modular Adder:	5
3.5 Disadvantages of the Modular Adder:	6
3.6 Applications of the Modular Adder:	6
4 Results	7
4.1 Schematic	7
4.2 Synthesis Design	7

Created By Team Alpha

1 Project Overview

A modular adder is a computational device or algorithm that performs modular addition, where two input numbers are added together, and the result is reduced by a modulus to ensure it stays within a certain range. Formally, the modular adder computes the operation:

$$\text{Result} = (a + b) \bmod m$$

2 Modular Adder

2.1 Description

A modular adder is a device or circuit used to perform modular addition in mathematics and computing. Modular addition involves adding two numbers together and then taking the result modulo a certain value, typically referred to as the modulus. It is a fundamental operation in various fields, including cryptography, digital signal processing, and number theory.

2.2 Key Features of the Modular Adder:

- **Addition Operation:** Adds two given numbers.
- **Modulo Operation:** After addition, the sum is reduced by the modulus to ensure the result lies within a specific range, usually 0 to m-1.

2.3 RTL Code

Listing 1: Modular Adder

```
1
2 module modular_adder #(parameter WIDTH = 4, MOD = 15) (
3     input  [WIDTH-1:0] A,
4     input  [WIDTH-1:0] B,
5     output [WIDTH-1:0] Sum,
6     output CarryOut
7 );
8
9     wire [WIDTH:0] temp_sum;
10
11     // Perform addition
12     assign temp_sum = A + B;
13
14     // If the sum exceeds MOD, wrap around
15     assign Sum = (temp_sum >= MOD) ? (temp_sum - MOD) : temp_sum;
16     assign CarryOut = (temp_sum >= MOD);
17
18 endmodule
```

2.4 Testbench

Listing 2: Modular Adder

```
1 module tb_modular_adder;
2
3     reg [3:0] A;
4     reg [3:0] B;
5     wire [3:0] Sum;
```

```

6    wire CarryOut;
7
8    modular_adder #(4, 15) uut (
9        .A(A),
10       .B(B),
11       .Sum(Sum),
12       .CarryOut(CarryOut)
13   );
14
15   initial begin
16       // Test case 1
17       A = 4'b1010; // 10
18       B = 4'b0111; // 7
19       #10;
20       $display("A = %d, B = %d, Sum = %d, CarryOut = %b", A, B, Sum,
21               CarryOut);
22
23       // Test case 2 (modular overflow)
24       A = 4'b1101; // 13
25       B = 4'b0100; // 4
26       #10;
27       $display("A = %d, B = %d, Sum = %d, CarryOut = %b", A, B, Sum,
28               CarryOut);
29
30       $stop;
31   end
32 endmodule

```

2.5 Simulation Results

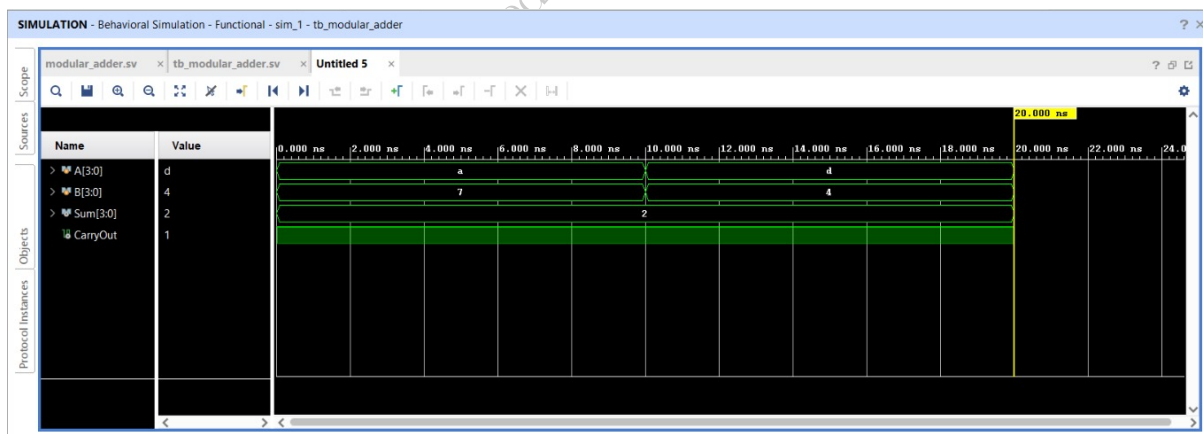


Figure 1: Simulation results of Modular Adder

3 How it works ?

3.1 Steps Involved in Modular Addition:

- **Addition:** Add the two numbers a and b normally, producing a sum $S=a+b$.
- **Modulo Operation:** Divide the sum S by the modulus m and take the remainder. This ensures the result stays within the range 0 to $m-1$.

If S is smaller than the modulus m , the result is simply S . If S is larger than or equal to m , the result is $S \bmod m$, which means subtracting multiples of m until the result is within the range.

Explanation

A Modular Adder is a device or algorithm used to perform modular addition, a fundamental operation in mathematics and computing. It involves adding two numbers and then applying a modulus operation to keep the result within a specific range. Modular adders are essential in systems where values are constrained within a defined range, preventing overflow and ensuring cyclical arithmetic.

3.2 Basic Concept of Modular Addition:

In Modular arithmetic, given two numbers a and b , and a modulus m , the modular addition of a and b is defined as: $(a+b) \bmod m$
where:

- a and b are the numbers to be added.
- m is the modulus (a predefined constant).
- The result is the remainder when $a+b$ is divided by m .

This operation ensures that the result always lies within the range $0 \leq \text{Result} < m$.

3.3 Comparison with Other Adders:

1. Modular Adder vs. Binary Adder:

Modular Adder: Performs addition followed by a modulus operation to constrain the result within a specific range, preventing overflow. Commonly used in cryptography and cyclic systems.

Binary Adder: Simply adds two binary numbers. Can overflow if the result exceeds the bit capacity. Used in general arithmetic.

2. Modular Adder vs. Ripple Carry Adder (RCA):

Modular Adder: Adds two numbers and applies modulus logic, ensuring bounded results. Slower due to modulus calculation.

RCA: Adds numbers by propagating the carry bit from one stage to the next. Simpler but slow for large numbers and prone to overflow.

3. Modular Adder vs. Carry-Lookahead Adder (CLA):

Modular Adder: Performs bounded addition, but slower because of the modulus check. Useful in scenarios needing wrap-around behavior.

CLA: Focuses on fast binary addition by minimizing carry propagation. Optimized for speed but lacks inherent modulus handling.

4. Modular Adder vs. Carry-Save Adder (CSA):

Modular Adder: Adds numbers and applies a modulus to limit the result. More complex due to modulus logic.

CSA: Optimized for adding multiple numbers simultaneously without propagating carry immediately. Fast for multi-operand operations.

5. Modular Adder vs. Half Adder / Full Adder:

Modular Adder: Adds larger numbers with modulus constraints, ensuring bounded results.

Half/Full Adder: Basic building blocks for binary addition of single or three bits (with carry). Simpler and used in basic arithmetic operations.

3.4 Advantages of the Modular Adder:

- **Prevents Overflow:**

Automatically bounds the result within a fixed range, ensuring that the output does not exceed the modulus. This makes it useful in systems where overflow needs to be managed.

- **Cyclical Nature:**

The result "wraps around" when it exceeds the modulus, making it ideal for cyclic or repetitive processes like clocks, counters, and circular buffers.

- **Efficiency in Specific Applications:**

Modular arithmetic is computationally efficient in certain cryptographic algorithms and number-theoretic applications, avoiding the need for large numbers to grow indefinitely.

- **Commutative and Associative:**

Modular addition is both commutative and associative, which simplifies parallel processing and mathematical proofs in algorithm design.

- **Wide Usage in Cryptography:**

Modular addition forms the core of cryptographic algorithms like RSA, Diffie-Hellman, and elliptic curve cryptography, which are widely used in secure communication and data encryption.

3.5 Disadvantages of the Modular Adder:

- **Additional Complexity:**

Requires extra circuitry or computational logic to handle the modulus operation, which can slow down operations compared to simple binary addition.

- **Limited Result Range:**

The result is constrained to a limited range $0 \leq \text{result} < m$ (less than equal to m), which may not be desirable in applications that need large numbers or precise arithmetic.

- **Slower Than Standard Adders:**

The additional modulus operation can add latency, making modular adders slower than high-speed adders like carry-lookahead or carry-save adders in applications not requiring modular constraints.

- **Not Suitable for All Arithmetic:**

In applications where continuous growth of numbers (without modulus) is needed, modular adders are inappropriate, as they truncate the result.

3.6 Applications of the Modular Adder:

- **Cryptography:**

Core to cryptographic algorithms such as RSA, Diffie-Hellman, and Elliptic Curve Cryptography (ECC). Modular addition ensures secure and bounded number operations within a predefined range.

- **Digital Clocks and Timers:**

Modular adders are used in digital clock circuits where time "wraps around" (e.g., a 12-hour clock uses modulo-12 arithmetic).

- **Cyclic Redundancy Checks (CRC):**

Used in error-detection codes and checksum algorithms to ensure data integrity, especially in network communications and data storage.

- **Digital Counters:**

Modular adders are used in systems where counters "reset" after reaching a certain value, such as modulo-256 counters in computer systems.

- **Hash Functions:**

Modular addition is used in hash algorithms to ensure that the output value of a hash function falls within a specific range, which is useful for hash tables and memory addressing.

- **Signal Processing:**

Used in algorithms and circuits that involve repetitive or cyclic processing, such as in filters and waveform generation.

- **Random Number Generators:**

Modular arithmetic is used in pseudorandom number generation to produce values within a controlled range, crucial in simulations and cryptographic applications.

4 Results

4.1 Schematic

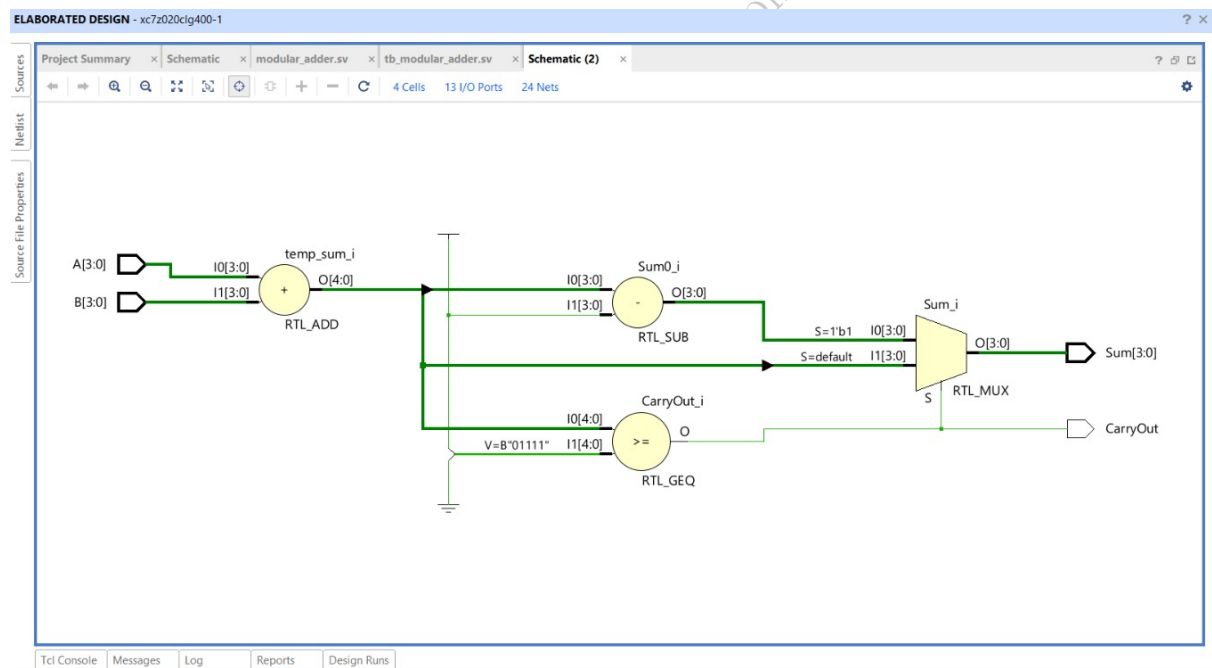


Figure 2: Schematic of Modular Adder

4.2 Synthesis Design

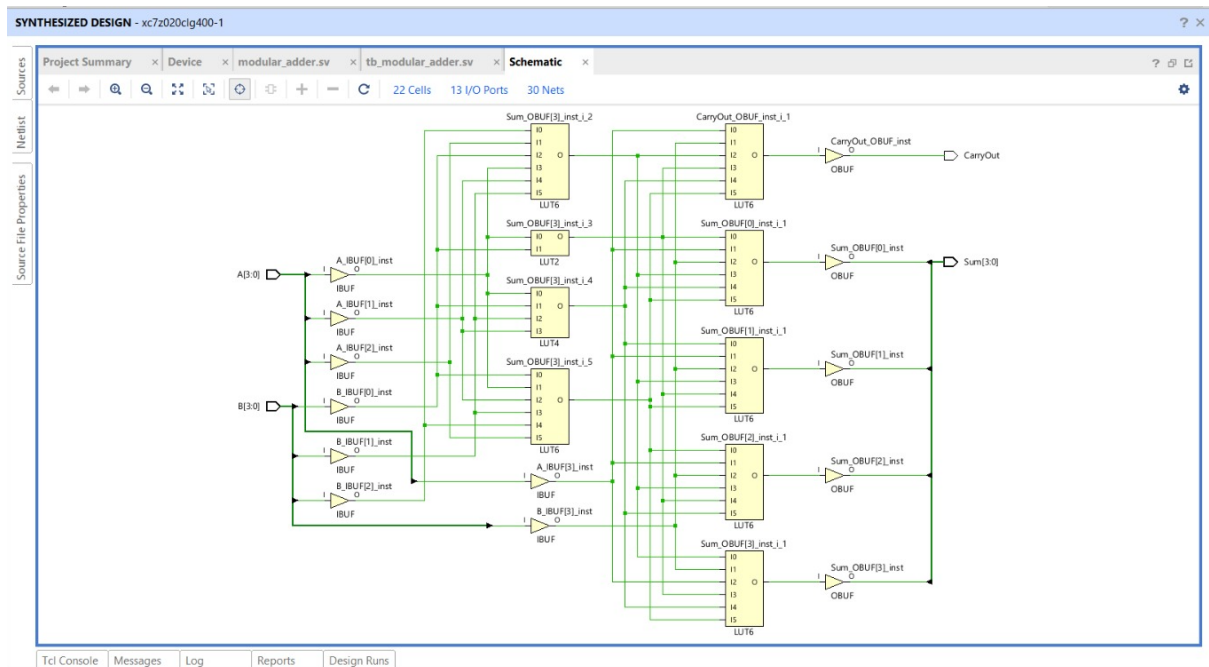


Figure 3: Synthesis Design of Modular Adderr