Project 49 : Systolic Multiplier

A Comprehensive Study of Advanced Digital Circuits

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

Documentation Specialist: Dhruv Patel & Nandini Maheshwari

Contents

1 Project Overview

A Systolic Multiplier is a type of hardware architecture based on a systolic array, which is a specialized computational model designed for performing repetitive, regular tasks efficiently. The systolic array architecture involves multiple processing elements (PEs) arranged in a grid or pipeline, where data flows rhythmically between the PEs in a synchronized manner. This architecture is particularly well-suited for tasks involving matrix multiplication, convolution operations, and signal processing.

2 Systolic Multiplier

2.1 Basic Concept of Systolic Multiplier

- **Systolic Array:**

A systolic multiplier is part of a systolic array, which is a grid or line of simple processing elements. The elements perform small tasks (like multiplication and addition) and pass intermediate results to neighboring elements.

- **Data Flow:**

In a systolic array, data flows through the processing elements in a pipelined fashion. Each element processes part of the data and forwards the result to the next element. This continuous flow of data allows multiple operations to happen in parallel, greatly increasing the speed of the overall computation.

- **Multiplication in Parts:**

Instead of computing a large multiplication in one step, the systolic multiplier divides it into smaller, more manageable pieces. For example, if you want to multiply two matrices, the systolic array computes the multiplication of individual elements step-by-step, accumulating partial results as the data moves through the array.

- **Processing Element (PE):**

Each PE in the array typically does a simple operation like multiplying two numbers and adding the result to an ongoing sum. For example, a PE might take two numbers (one from each input), multiply them, and add the result to the current sum from the previous PE.

- **Pipelined Operation:**

Data is fed into the array, and as it moves through, each PE works on part of the operation. Since the data is constantly flowing, the systolic array acts like a pipeline—allowing for high throughput of operations.

- **Parallelism:**

The major advantage is parallelism. Multiple parts of the computation are done at the same time by different PEs, which makes the systolic multiplier much faster than sequential methods for tasks like matrix multiplication.

2.2 Explanation of Systolic Multiplier

A Systolic Multiplier is a parallel hardware architecture used for efficient multiplication, especially in matrix operations. It consists of a grid of Processing Elements (PEs) that work in a synchronized manner, performing simple arithmetic operations like multiplication and addition. Data (such as matrix elements) flows through the PEs in a pipelined fashion, allowing for continuous and simultaneous processing of inputs.

Each PE multiplies corresponding input values and accumulates partial sums, passing the results to the next PE. The final result, such as an element of the product matrix, is produced after all PEs have processed the data. This architecture provides high throughput and efficient parallelism, making it ideal for tasks like matrix multiplication, convolution, and digital signal processing.

2.3 Working of Systolic Multiplier

- **Initialization:**

Matrix elements or operands are loaded into the systolic array. For matrix multiplication, rows from matrix A and columns from matrix B are fed into the array.

- **Processing Elements (PEs) Operation:**

Each PE receives two input values (one from matrix A and one from matrix B). The PE multiplies these two values and adds the product to the partial sum it has accumulated so far.

- **Data Propagation:**

After each PE processes its inputs, it passes the partial sum to the next PE in the array. The next PE continues accumulating the partial products.

- **Pipelined Data Flow:**

New data is continuously fed into the array while previous inputs are still being processed. This ensures parallel processing and increases throughput.

- **Partial Result Accumulation:**

Each PE adds its contribution (product of its inputs) to the cumulative sum, gradually building up the final result.

- **Final Output:**

Once all necessary data has been processed by the PEs, the final result (e.g., an element of the product matrix) emerges from the last PE in the array.

- **Synchronization with Clock:**

All operations within the systolic array are synchronized by a global clock, ensuring that the PEs perform operations and pass results in lockstep.

- **Continuous Processing:**

The systolic array operates continuously, allowing new computations to be processed in a pipelined manner without waiting for previous results to fully complete.

2.4 RTL Code

Listing 1: Systolic Multiplier

```


module systolic_multiplier(
    input logic clk,
    input logic reset,
    input logic [1:0] A,
    input logic [1:0] B,
    output logic [3:0] P
);
    logic [1:0] partial[1:0];

    // Systolic array for multiplication
    always_ff @(posedge clk or posedge reset) begin
        if (reset) begin
            P <= 4'b0;
            partial[0] <= 2'b0;
            partial[1] <= 2'b0;
        end else begin
            // Systolic operation
            partial[0] <= A[0] ? B : 2'b0; // First row
            partial[1] <= A[1] ? B : 2'b0; // Second row

            // Final product calculation
            P <= (partial[0] << 0) + (partial[1] << 1);
        end
    end
endmodule
```

2.5 Testbench

Listing 2: Systolic Multiplier

```


module tb_systolic_multiplier;
    logic clk;
    logic reset;
    logic [1:0] A;
    logic [1:0] B;
    logic [3:0] P;

    // Instantiate the Systolic Multiplier
    systolic_multiplier uut (
        .clk(clk),
        .reset(reset),
        .A(A),
        .B(B),
        .P(P)
    );

    // Clock generation
    initial begin
        clk = 0;
        forever #5 clk = ~clk; // 10 time units clock period
    end

    // Test sequence
    initial begin
        reset = 1;
        #10;
        reset = 0;

        A = 2'b01; // 1
        B = 2'b10; // 2
        #10; // Wait for one clock cycle
        $display("P = %b, Expected = %b", P, A * B); // P should be 2

        A = 2'b10; // 2
        B = 2'b11; // 3
        #10; // Wait for one clock cycle
        $display("P = %b, Expected = %b", P, A * B); // P should be 6

        A = 2'b11; // 3
        B = 2'b01; // 1
        #10; // Wait for one clock cycle
        $display("P = %b, Expected = %b", P, A * B); // P should be 3

        // Add more test cases as needed

        $finish; // End the simulation
    end
endmodule
```

3 Results

3.1 Simulation

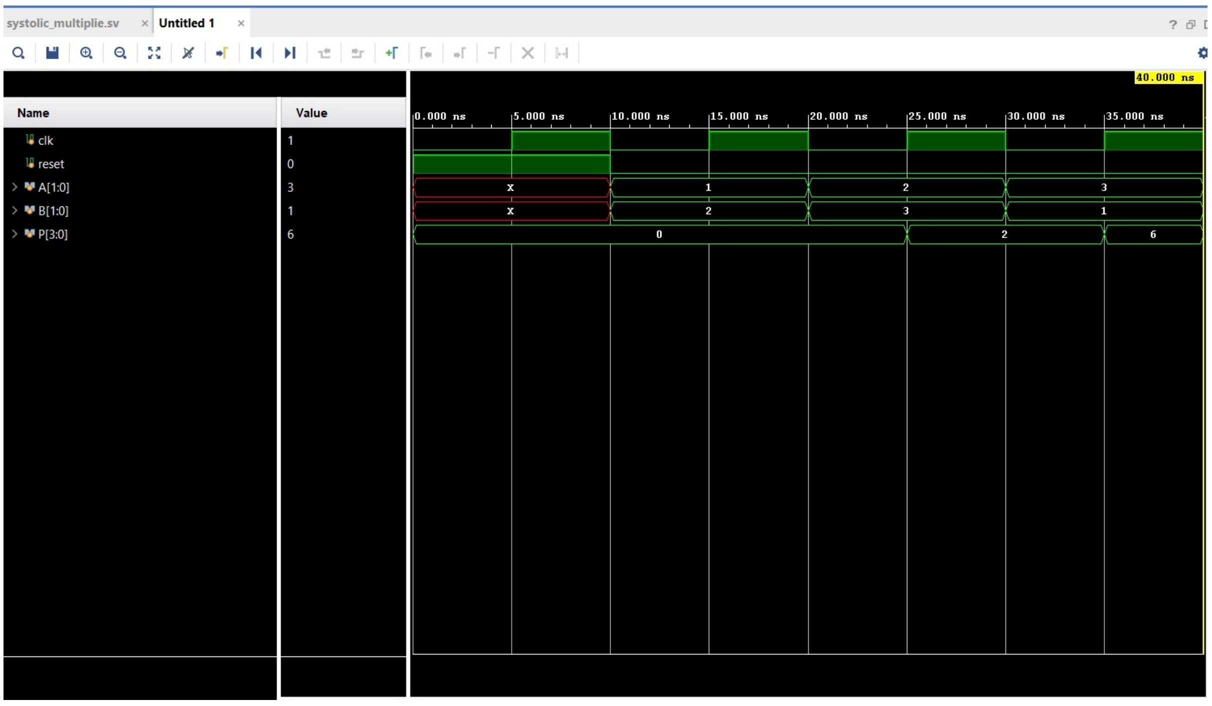

Figure 1: Simulation of Systolic Multiplier

3.2 Schematic

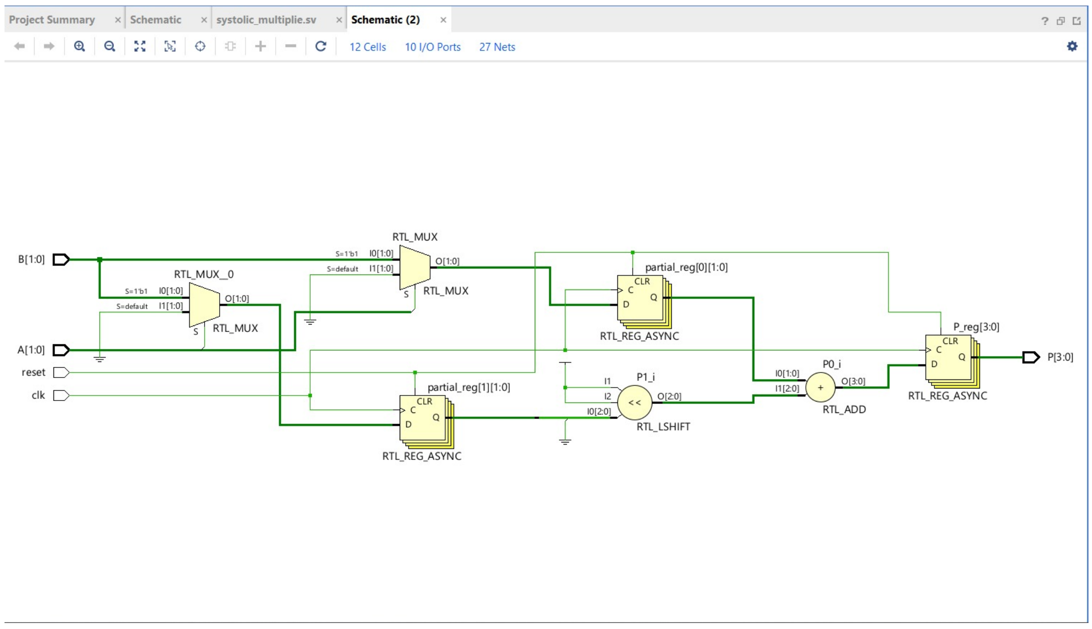

Figure 2: Schematic of Systolic Multiplier

3.3 Synthesis Design

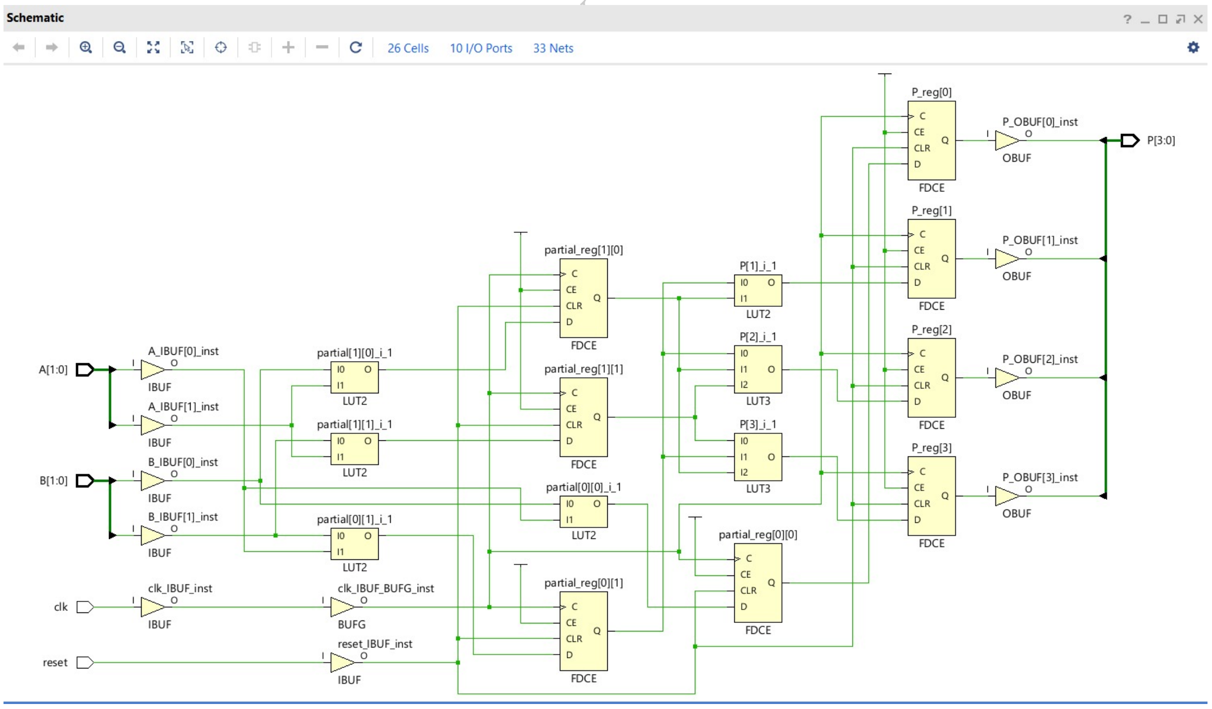

Figure 3: Synthesis Design of Systolic Multiplier

4 Advantages of Systolic Multiplier

1. High Throughput:
Systolic multipliers use pipelining and parallel processing, allowing for continuous data processing. This results in high throughput for large computations, especially matrix multiplications.

2. Efficient Data Flow:
Data is passed locally between neighboring processing elements (PEs), reducing memory access and the need for complex global data transfers. This lowers the communication overhead.

3. Scalability:
The systolic array architecture is highly scalable. More PEs can be added to handle larger matrices or more complex operations, making it flexible for various sizes of problems.

4. Modularity:
The repetitive, regular structure of the systolic array makes it easy to implement in hardware and simplifies design, testing, and scaling.

5. Low Latency:
Due to its pipelined structure, once the systolic array is filled with data, each clock cycle produces a new output, resulting in low latency for large-scale operations.

6. Reduced Power Consumption:
As data movement is localized to neighboring PEs, systolic multipliers can consume less power compared to architectures that require frequent global memory access.

5 Disadvantages of Systolic Multiplier

1. Fixed Design for Specific Tasks:
The systolic array is designed for specific repetitive tasks like matrix multiplication. It lacks flexibility and is not as efficient for general-purpose computation or tasks that require irregular data access patterns.

2. Complex Control Logic:
The synchronization of the PEs requires precise control, and designing the control logic for timing and data flow can be complex.

3. Initial Latency:
There is an initial latency before the systolic array begins outputting results, as the data needs to propagate through the array. This can affect performance in applications where the input size is small.

4. Area and Hardware Cost:
A systolic array requires multiple PEs, leading to increased area and cost in hardware implementations. The number of PEs can significantly affect the hardware resources needed.

6 Applications of Systolic Multiplier

1. Matrix Multiplication:
Widely used in applications where large-scale matrix multiplication is required, such as in scientific computing, 3D graphics, and machine learning (e.g., neural network operations).

2. Digital Signal Processing (DSP):
Systolic multipliers are used for tasks such as filtering, Fourier transforms, and convolution, all of which involve repetitive multiplications and summations.

3. Image Processing:

Applications such as image compression, feature extraction, and transformations (e.g., convolution operations in image filters) benefit from the high parallelism of systolic arrays.

4. Artificial Intelligence (AI) and Machine Learning (ML):
Commonly used in AI hardware accelerators for efficient execution of deep learning algorithms, particularly in convolutional layers of neural networks.

5. Cryptography:
Used in cryptographic computations, where matrix and vector operations play a crucial role in algorithms like RSA and elliptic curve cryptography.

6. Real-time Systems:
Systolic multipliers are used in real-time embedded systems, such as those for signal processing in telecommunications and radar systems, where speed and parallelism are crucial.

7 Conclusion

A systolic multiplier is a parallel, pipelined architecture where small, synchronized processing elements work together to quickly perform large computations like matrix multiplication by breaking them into smaller, manageable parts.

8 FAQs

1. What is a systolic multiplier?

Answer: A systolic multiplier is a parallel, pipelined hardware architecture designed to perform repetitive arithmetic operations, such as matrix multiplication, using a grid of processing elements (PEs). Data flows synchronously through the PEs, which multiply and accumulate partial results.

2. How does a systolic multiplier work?

Answer: Data, such as matrix rows and columns, is fed into the systolic array, where each processing element multiplies inputs, accumulates partial sums, and passes results to neighboring PEs. The result is computed in a pipelined, parallel manner.

3. What are the advantages of a systolic multiplier?

Answer: Advantages include high throughput due to parallel processing, efficient data flow between PEs (reducing memory access), scalability, low latency after pipeline fill, and reduced power consumption through localized data transfer.

4. What are the limitations of systolic arrays?

Answer: Systolic arrays are task-specific (e.g., matrix operations), have high initial latency due to data propagation, require complex control logic, and involve higher hardware area and cost.

5. What is the key difference between a systolic array and a traditional multiplier?

Answer: A traditional multiplier performs multiplication sequentially, while a systolic array uses parallelism and pipelining to perform multiple operations simultaneously, making it faster for large-scale computations like matrix multiplication.

6. Explain the role of Processing Elements (PEs) in a systolic array.

Answer: PEs are basic units in the systolic array that perform multiplication and accumulation. They process data in parallel and forward partial results to the next PE, enabling continuous data flow and computation.

7. What kind of applications benefit from systolic multipliers?

Answer: Applications include matrix multiplication in scientific computing, digital signal processing, image processing, AI/machine learning, cryptography, and real-time systems like telecommunications and radar processing.

8. Why is pipelining important in a systolic multiplier?

Answer: Pipelining allows the systolic multiplier to process new data inputs continuously while earlier inputs are still being processed. This maximizes throughput and ensures high-speed performance for large computations.

9. What is the significance of the clock in a systolic array?

Answer: The clock synchronizes the operations of all PEs, ensuring that data flows rhythmically and computations are completed in lockstep, maintaining synchronization across the entire array.

10. What are the different types of systolic arrays?

Answer: Common types include 1D systolic arrays (used for simpler tasks like vector multiplication) and 2D systolic arrays (used for complex operations like matrix multiplication). The arrangement depends on the application requirements.