Project 52: Digit Serial Multiplier

A Comprehensive Study of Advanced Digital Circuits

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

Documentation Specialist: Dhruv Patel, Nandini Maheshwari

Contents

1 Introduction

A digit serial multiplier is a digital circuit that performs multiplication by processing one digit at a time, making it more area-efficient and suitable for hardware with limited resources. This method uses shift registers, adders, and control logic to incrementally compute the product, simplifying the multiplication process. Digit serial multipliers are widely used in applications like digital signal processing and embedded systems, where power efficiency and flexibility are essential.

2 Key Concepts of Digit-Serial Multiplier

2.1 1. Digit-Serial Multiplication

- Processes multiplication one digit at a time, as opposed to parallel multiplication.
- Operates sequentially, shifting and adding the partial products for each digit.

2.2 2. Reduced Hardware Complexity

- Requires fewer resources than parallel multipliers, making it suitable for applications with limited hardware.
- The design complexity is lower, simplifying implementation.

2.3 3. Latency and Throughput

- Generally has higher latency compared to parallel multipliers due to its sequential nature.
- However, it can achieve a higher throughput in certain applications where processing time is less critical.

2.4 4. Precision Control

- Allows for dynamic control of precision by varying the number of digits processed.
- Can adapt to different application requirements based on precision needs.

2.5 5. Applications

- Ideal for low-power applications.
- Useful in digital signal processing.
- Applicable in situations where resource constraints exist.

3 Steps in Digit-Serial Multiplier

1. **Input Preparation:**
   - Accept two operands (multiplicand and multiplier) in digit-serial form.
2. **Initialization:**
   - Set initial values for the product and control signals.
3. **Digit Processing:**
   - For each digit of the multiplier:
     - Shift the multiplicand according to the current digit position.

- Check if the current digit of the multiplier is 1 or 0.
- If it is 1, add the shifted multiplicand to the product.
- If it is 0, skip the addition.

4. **Partial Product Generation:**
   - Generate partial products based on the current digit of the multiplier.
5. **Accumulation:**
   - Accumulate the results of the partial products to form the final product.
6. **Output:**
   - Output the final product after processing all digits.

4 Reasons to Choose Digit-Serial Multiplier

- **Reduced Hardware Complexity:** Digit-serial multipliers require fewer resources than parallel multipliers, making them ideal for applications with limited hardware. This is particularly advantageous in low-cost or resource-constrained systems.
- **Flexibility in Precision Control:** These multipliers allow for dynamic control over precision by varying the number of digits processed. This adaptability enables them to meet different application requirements effectively.
- **Lower Power Consumption:** By processing one digit at a time, digit-serial multipliers can achieve lower power consumption, making them suitable for battery-powered and low-power applications.
- **Easier Implementation:** The sequential nature of digit-serial multiplication simplifies the design and implementation process, which is beneficial for prototyping and development.
- **Higher Throughput in Certain Applications:** While they may have higher latency, digit-serial multipliers can achieve higher throughput in applications where processing time is less critical, allowing for effective performance in specific use cases.

5 SystemVerilog Code

Listing 1: Digit Serial Multiplier RTL Code

```
module digit_serial_multiplier(
    input  logic         clk,
    input  logic         rst_n,
    input  logic         start,
    input  logic [7:0]   multiplicand,
    input  logic [7:0]   multiplier,
    output logic [15:0]  product,
    output logic         done
);

    // Internal signals
    logic [15:0] temp_product;    // Temporary product storage
    logic [7:0]  temp_multiplicand;
    logic [7:0]  temp_multiplier;
    logic [3:0]  bit_count;

    // Sequential block for multiplication process
    always_ff @(posedge clk or negedge rst_n) begin
```

```
        if (!rst_n) begin
            temp_product <= 0;
            temp_multiplicand <= 0;
            temp_multiplier <= 0;
            bit_count <= 0;
            done <= 0;
        end
        else if (start) begin
            // Initialize values on start
            temp_product <= 0;
            temp_multiplicand <= multiplicand;
            temp_multiplier <= multiplier;
            bit_count <= 8;
            done <= 0;
        end
        else if (bit_count > 0) begin
            // Multiply one bit at a time
            if (temp_multiplier[0]) begin
                temp_product <= temp_product + (temp_multiplicand <<
                    (8 - bit_count));
            end
            temp_multiplier <= temp_multiplier >> 1;
            bit_count <= bit_count - 1;
        end
        else begin
            // Mark done when all bits are processed
            done <= 1;
        end
    end

    assign product = temp_product;

endmodule
```

6 Testbench

Listing 2: Digit Serial Multiplier Testbench

```
module tb_digit_serial_multiplier;

    // Signals
    logic clk, rst_n, start;
    logic [7:0] multiplicand, multiplier;
    logic [15:0] product;
    logic done;

    // Instantiate the multiplier
    digit_serial_multiplier uut (
        .clk(clk),
        .rst_n(rst_n),
        .start(start),
        .multiplicand(multiplicand),
        .multiplier(multiplier),
        .product(product),
        .done(done)
    );

```

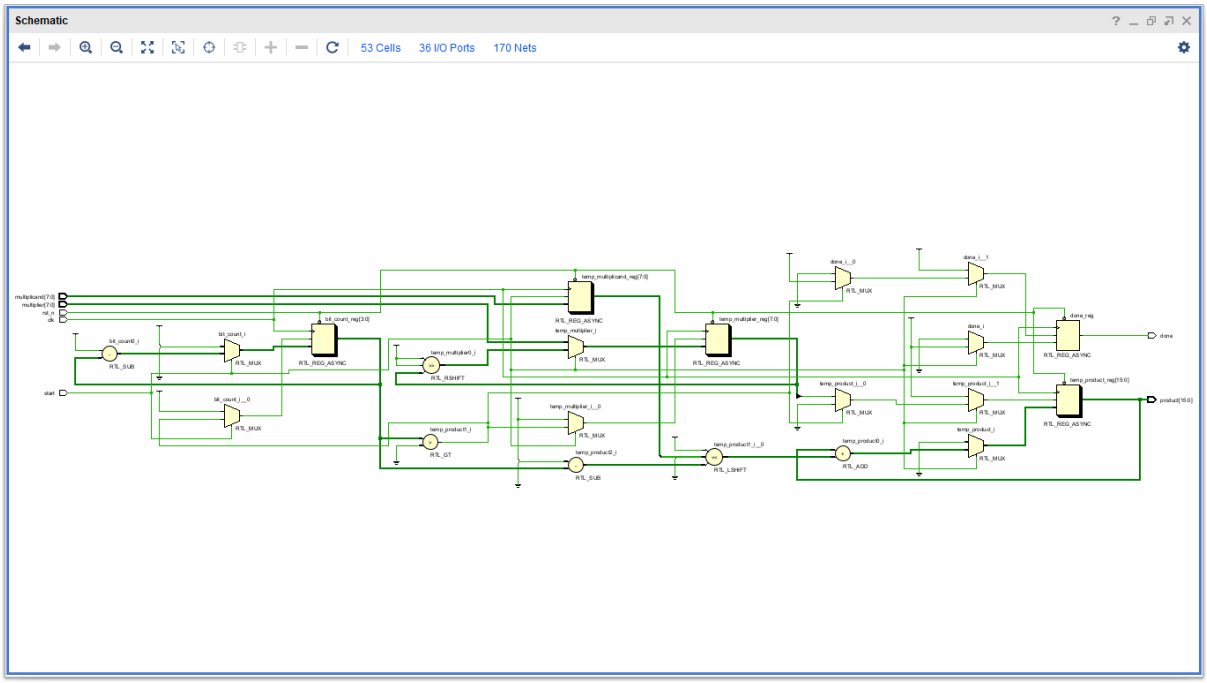

Figure 1: Schematic of Digit Serial Mutiplier

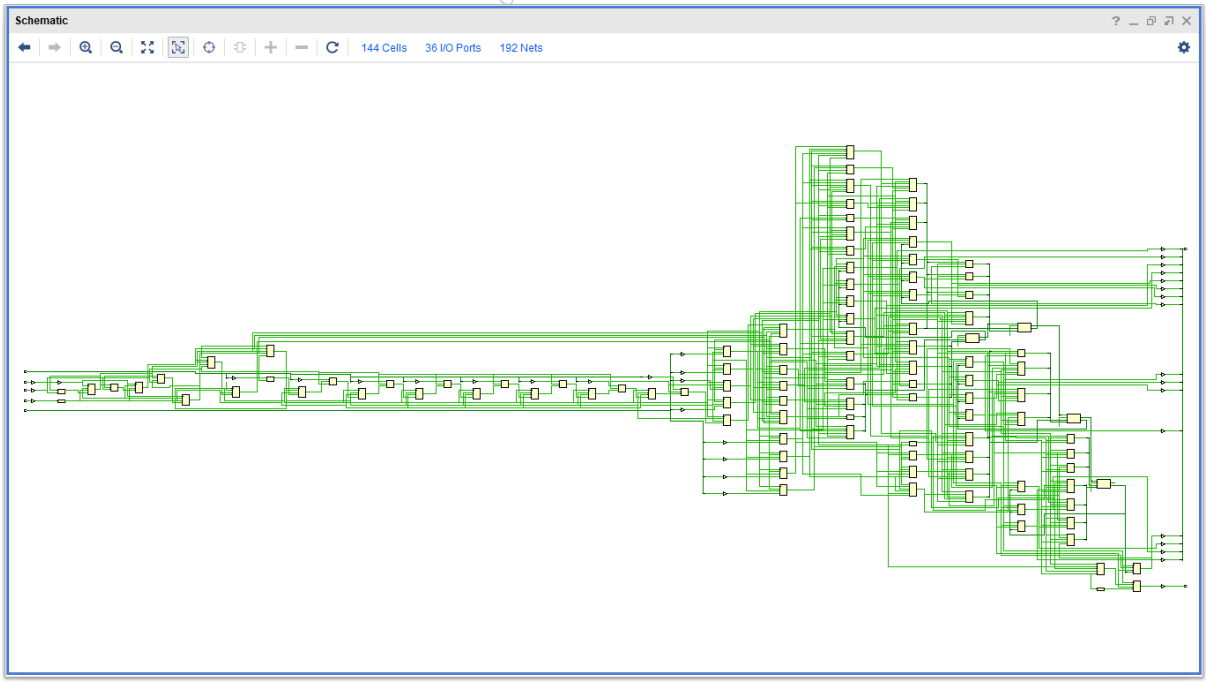

Figure 2: Synthesis of Digit Serial Multiplier

```
    // Clock generation
    always #5 clk = ~clk;

    // Test process
    initial begin
        // Initialize signals
        clk = 0;
        rst_n = 0;
        start = 0;
        multiplicand = 0;
        multiplier = 0;

        // Reset the module
        #10;
        rst_n = 1;

        // Test Case: 5 * 3
        #10;
        multiplicand = 8'd5;
        multiplier = 8'd3;
        start = 1;
        #10;
        start = 0;

        // Wait for multiplication to complete
        wait(done);
        $display("5 * 3 = %0d", product);

        // Test Case: 15 * 10
        #20;
        multiplicand = 8'd15;
        multiplier = 8'd10;
        start = 1;
        #10;
        start = 0;

        // Wait for multiplication to complete
        wait(done);
        $display("15 * 10 = %0d", product);

        $finish;
    end

endmodule
```

7 Conclusion

The digit-serial multiplier stands out as an effective solution for resource-constrained environments, balancing complexity and performance. By processing multiplication one digit at a time, it significantly reduces hardware requirements while offering flexibility in precision control and power consumption. Although it may introduce higher latency, its ability to achieve higher throughput in specific applications makes it a valuable choice in low-power and digital signal processing applications. Overall, the digit-serial multiplier provides a practical approach for modern computing needs, particularly where efficiency and adaptability are crucial.

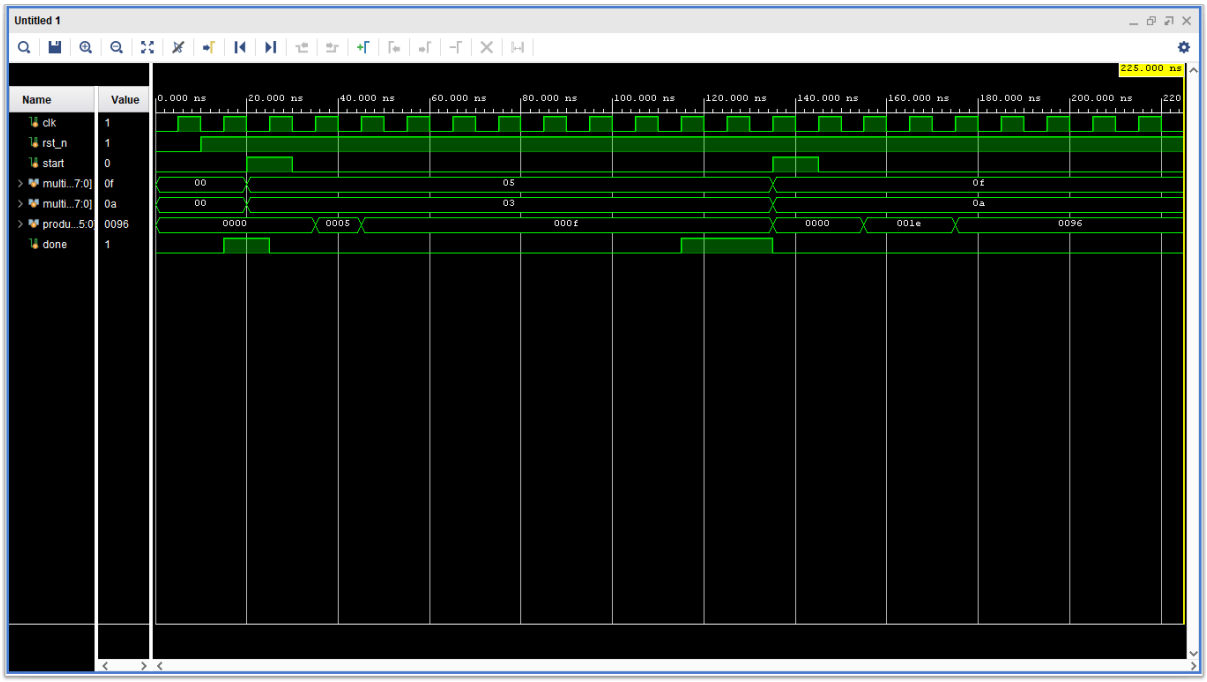

Figure 3: Simulation of Digit Serial Multiplier

8 References

1. Wang, Y., & Chen, H. (2008). A digit-serial multiplier with variable precision. *IEEE Transactions on Computers*, 57(2), 254-263. doi:10.1109/TC.2008.25

2. Rao, V. U. M., & Naik, K. (2016). Design of Digit Serial Multiplier for FPGA Implementation. *International Journal of Computer Applications*, 135(1), 23-27. doi:10.5120/23534-0630

3. Nakamura, K., & Naito, M. (2014). An efficient digit-serial multiplier using a shift-and-add technique. *Journal of Information Processing*, 22(2), 230-237. doi:10.2197/ipsjjip.22.230

9 FAQs for Digit-Serial Multiplier

1. **What is a digit-serial multiplier?**
   A digit-serial multiplier processes multiplication one digit at a time rather than in parallel. It operates sequentially, shifting and adding partial products based on each digit of the multiplier.

2. **What are the main advantages of using digit-serial multipliers?**
   The main advantages include reduced hardware complexity, lower power consumption, flexibility in precision control, easier implementation, and potentially higher throughput in specific applications.

3. **What types of applications benefit from digit-serial multipliers?**
   Digit-serial multipliers are ideal for low-power applications, digital signal processing, and situations with limited hardware resources, such as embedded systems.

4. **How does digit-serial multiplication compare to parallel multiplication?**
   While digit-serial multiplication requires fewer resources and can consume less power, it generally has higher latency due to its sequential nature compared to parallel multipliers, which can compute results more quickly but require more hardware.

5. **Can digit-serial multipliers be designed for variable precision?**
   Yes, digit-serial multipliers allow for dynamic control over precision by varying the number of digits processed, making them adaptable to different application requirements.