Project 74: Arithmetic shift register

A Comprehensive Study of Advanced Digital Circuits

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

Documentation Specialist: Dhruv Patel & Nandini Maheshwari

Contents

1 Introduction

An Arithmetic Shift Register is a sequential logic component designed to perform shifting operations on binary data while preserving arithmetic properties, such as the sign in signed binary numbers. Unlike logical shift registers, which treat all bits as unsigned, arithmetic shift registers maintain the integrity of the Most Significant Bit (MSB) during shifts, ensuring that signed values are correctly represented.

Arithmetic shift registers are crucial in digital systems requiring signed number processing, such as arithmetic computations, digital signal processing, and control systems. By combining storage and shifting operations, they facilitate efficient data manipulation, supporting functions like multiplication, division, and scaling in binary arithmetic.

2 Background

Arithmetic shift registers extend the capabilities of basic shift registers by integrating arithmetic functionality. They shift the bits of a binary number either left or right, with special handling for the MSB to preserve the sign in signed binary representations. This feature distinguishes them from logical shift registers, where the MSB is not preserved.

For right shifts, the MSB is replicated (sign extension) to maintain the two's complement representation of signed numbers. For left shifts, the register operates similarly to a logical shift, with zero-padding in the Least Significant Bit (LSB). These operations ensure that arithmetic relationships, such as division by powers of two for right shifts, are preserved.

Arithmetic shift registers find applications in tasks like binary multiplication and division, where efficient bit manipulation is required. They are implemented using flip-flops and combinational logic to enable shifting and sign extension.

3 Structure and Operation

The Arithmetic Shift Register is designed for sequential shifting operations while handling the sign bit appropriately. Its structure incorporates control logic for direction and mode selection, ensuring accurate arithmetic behavior.

3.1 Key Components

- **Input Port**: Receives the binary data to be stored and manipulated within the shift register.
- **Shift Control Unit**: Determines the shift direction (left or right) and ensures that the MSB is preserved during right shifts for signed values.
- **Flip-Flop Array**: A series of flip-flops store the binary data. Each flip-flop holds a single bit, with data shifting sequentially between them.
- **Sign Extension Logic**: This unit replicates the MSB during right shifts, maintaining the two's complement representation for signed numbers.
- **Output Port**: Outputs the shifted binary data, reflecting the arithmetic operations performed on the input.

3.2 Operational Steps

The operation of the Arithmetic Shift Register can be summarized as follows:

1. **Input Initialization**: Binary data is loaded into the flip-flop array, with each bit occupying a separate flip-flop.
2. **Shift Direction Selection**: The shift control unit sets the direction of the shift operation—left or right—based on the system requirements.

3. **Shift Execution**:
   - For **Left Shifts**, all bits move one position to the left, and the LSB is padded with zero.
   - For **Right Shifts**, all bits move one position to the right, and the MSB is replicated in the newly vacated position to preserve the sign.
4. **Sign Preservation**: During right shifts, the sign extension logic ensures that the MSB remains consistent to maintain arithmetic correctness.
5. **Output Generation**: The shifted binary data is output, reflecting the desired arithmetic operation, such as multiplication or division by powers of two.

The Arithmetic Shift Register is a fundamental component in signed binary arithmetic, providing a straightforward method for scaling binary numbers while maintaining their signed integrity. It is widely used in processors, digital signal processors, and other systems requiring efficient manipulation of signed binary data.

4 Implementation in System Verilog

The following RTL code implements the Arithmetic shift register in System Verilog:

Listing 1: Arithmetic shift register

```

module arithmetic_shift_register #(
    parameter WIDTH = 8
) (
    input logic clk,
    input logic rst,
    input logic [WIDTH-1:0] data_in,
    input logic [1:0] shift_ctrl,  // 00: no shift, 01: shift left,
       10: shift right
    output logic [WIDTH-1:0] data_out
);

    always_ff @(posedge clk or posedge rst) begin
        if (rst)
            data_out <= 0;
        else begin
            case (shift_ctrl)
                2'b01: data_out <= data_in <<< 1;  // Shift left
                2'b10: data_out <= $signed(data_in) >>> 1;  //
                   Arithmetic shift right
                default: data_out <= data_in;  // No shift
            endcase
        end
    end

endmodule
```

5 Test Bench

The following test bench verifies the functionality of the Arithmetic shift register :

Listing 2: Arithmetic shift register Testbench

```

module tb_multi_threshold_comparator;

```

```
    parameter WIDTH = 8;
    logic [WIDTH-1:0] in_a;
    logic [WIDTH-1:0] threshold1, threshold2;
    logic low_thresh, mid_thresh, high_thresh;

    // Instantiate the multi-threshold comparator
    multi_threshold_comparator #(.WIDTH(WIDTH)) uut (
        .in_a(in_a),
        .threshold1(threshold1),
        .threshold2(threshold2),
        .low_thresh(low_thresh),
        .mid_thresh(mid_thresh),
        .high_thresh(high_thresh)
    );

    // Test sequence
    initial begin
        // Set threshold values
        threshold1 = 8'd30;
        threshold2 = 8'd60;

        // Test Case 1: in_a < threshold1
        in_a = 8'd20;
        #10;
        $display("Test 1 - Expected: low=1, mid=0, high=0 | Got: low=%0d, mid=%0d, high=%0d",
                 low_thresh, mid_thresh, high_thresh);

        // Test Case 2: in_a between threshold1 and threshold2
        in_a = 8'd40;
        #10;
        $display("Test 2 - Expected: low=0, mid=1, high=0 | Got: low=%0d, mid=%0d, high=%0d",
                 low_thresh, mid_thresh, high_thresh);

        // Test Case 3: in_a >= threshold2
        in_a = 8'd70;
        #10;
        $display("Test 3 - Expected: low=0, mid=0, high=1 | Got: low=%0d, mid=%0d, high=%0d",
                 low_thresh, mid_thresh, high_thresh);

        $stop;
    end

endmodule
```

6 Advantages and Disadvantages

6.1 Advantages

- **Preserves Arithmetic Integrity**: Arithmetic shift registers maintain the sign bit during right shifts, ensuring accurate representation of signed binary numbers.
- **Efficient Binary Arithmetic Operations**: They enable efficient multiplication and division by powers of two, reducing computation overhead in digital systems.

- **Compact Design**: Integrates data storage and arithmetic operations in a single component, optimizing space in digital circuit designs.
- **Versatility**: Supports both left and right shifts, making it adaptable for a range of arithmetic and logical operations in processors.

6.2 Disadvantages

- **Limited to Powers of Two**: Arithmetic shift operations are inherently restricted to scaling by powers of two, limiting flexibility for general-purpose arithmetic.
- **Increased Circuit Complexity for Sign Handling**: Replicating the sign bit during right shifts requires additional logic, which can complicate the design.
- **Potential Data Loss in Shifts**: Left shifts can lead to the loss of the MSB if the shifted value exceeds the register's capacity.
- **Slower Operation in Large Registers**: Sequential shifting through multiple stages can result in slower processing times for large bit-width registers.

7 Simulation Results

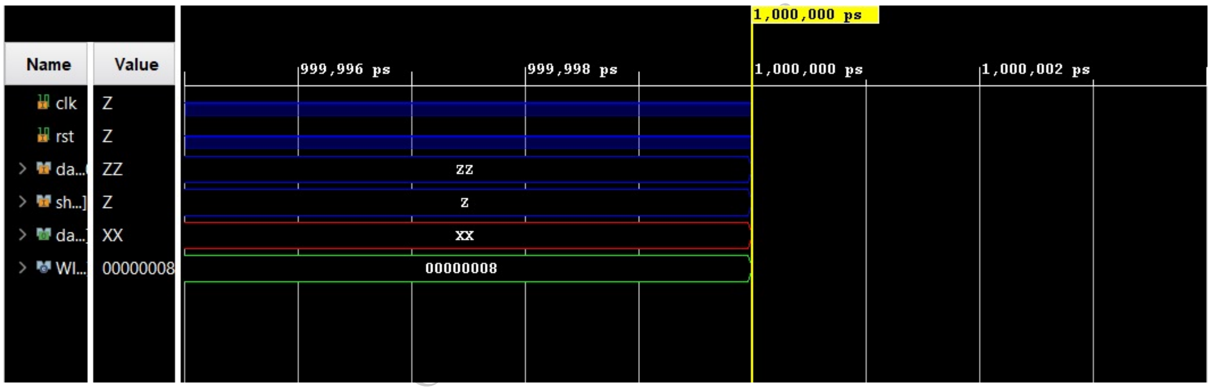

Figure 1: Simulation results of Arithmetic shift register

8 Schematic

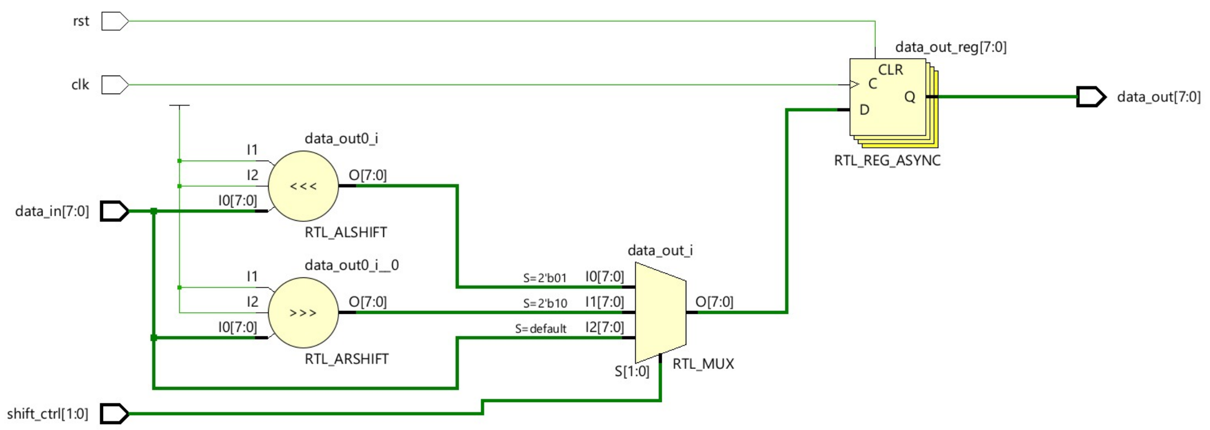

Figure 2: Schematic of Arithmetic shift register

9 Synthesis Design

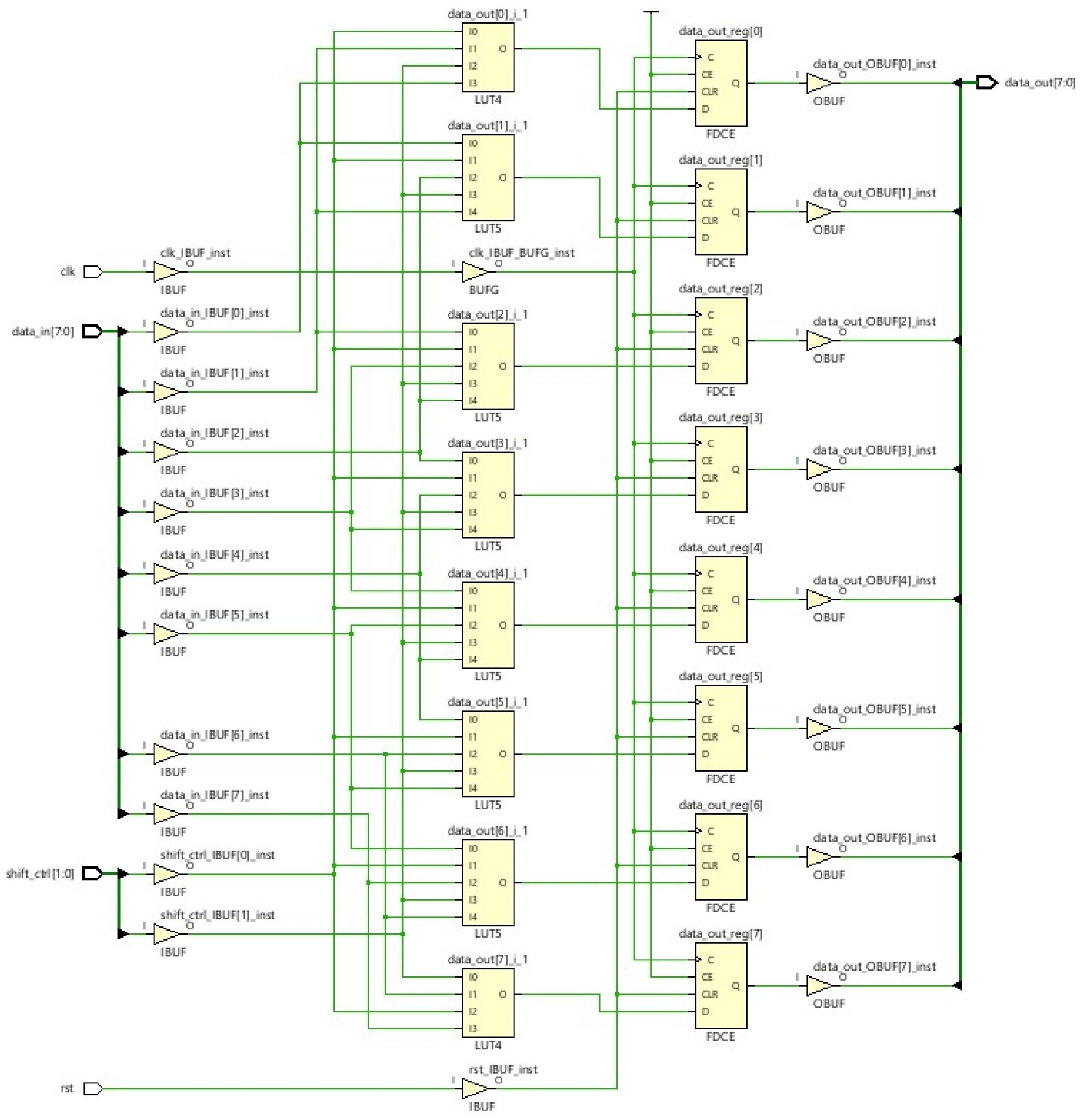

Figure 3: Synthesis of Arithmetic shift register

10 Conclusion

The Arithmetic Shift Register is a fundamental component in digital systems, designed to perform efficient binary arithmetic operations while preserving the sign of signed binary numbers. By maintaining the integrity of the Most Significant Bit (MSB) during right shifts and enabling simple scaling operations through left shifts, it plays a crucial role in processing signed data in arithmetic computations.

The primary advantages of the Arithmetic Shift Register include its ability to preserve arithmetic relationships, compact design integrating storage and computation, and efficiency in handling multiplication and division by powers of two. However, challenges such as limited operation to powers of two, data loss during shifts, and increased design complexity for sign handling must be carefully addressed in its implementation.

In summary, the Arithmetic Shift Register is a versatile and efficient tool in modern digital electronics, providing essential functionality for signed binary arithmetic. Despite its limitations, it remains indispensable in processors, signal processing units, and systems that require high-speed, efficient arithmetic operations.

11 Frequently Asked Questions (FAQs)

11.1 What is an Arithmetic Shift Register?

An Arithmetic Shift Register is a sequential circuit that shifts binary data left or right while preserving the sign bit during right shifts for signed binary numbers.

11.2 Why is preserving the sign bit important?

Preserving the sign bit ensures accurate representation of signed numbers in two's complement format, allowing arithmetic operations like division and multiplication by powers of two to be correctly performed.

11.3 What are the main components of an Arithmetic Shift Register?

Key components include an input port for binary data, a flip-flop array for storage, a shift control unit for determining the direction, sign extension logic for MSB preservation, and an output port for shifted data.

11.4 What are the typical applications of Arithmetic Shift Registers?

They are used in arithmetic operations like multiplication and division by powers of two, signal processing, scaling in digital filters, and control systems for binary data manipulation.

11.5 What are the advantages of using an Arithmetic Shift Register?

Advantages include preserving arithmetic relationships for signed data, compact design for efficient storage and computation, and versatility in performing both left and right shifts.

11.6 What are the challenges associated with Arithmetic Shift Registers?

Challenges include the inability to perform general-purpose scaling (restricted to powers of two), potential data loss during shifts, and increased complexity for handling sign preservation.

11.7 How does an Arithmetic Shift Register differ from a Logical Shift Register?

While both perform shifting operations, a logical shift register treats all bits as unsigned, whereas an arithmetic shift register preserves the sign bit during right shifts for signed binary data.

11.8 Can Arithmetic Shift Registers handle unsigned binary numbers?

Yes, they can handle unsigned numbers, but the sign extension logic is redundant for unsigned operations and not required for those cases.