Orange
Digital Center

Orange Digital Center

Digital Design and FPGA Flow

Assignment 2
Week 2

Created By: Ahmed Mostafa Gomaa
Submitted to :
Dr. Tamer Saleh
Eng. Asmaa El-sayed
Eng. Nourhan

Table of Contents

1. Introduction

In this task, we will design and implement a 4-bit Arithmetic Logic Unit (ALU) using Verilog. An ALU is a fundamental component of a computer's central processing unit (CPU) that performs arithmetic and logical operations on binary data. The ALU takes two 4-bit inputs, A[3:0] and B[3:0], and produces a 4-bit output F[3:0] based on the control signals S[3:0]. Additionally, the ALU generates a carry-out signal for arithmetic operations.

The ALU supports a variety of operations, including addition, subtraction, logical AND, logical OR, logical XOR, 1's complement, 2's complement, and comparison operations (greater than, less than, and equal to). The control signals S[3:0] determine which operation is performed, as specified in the provided truth table.

The design will be modular, with separate units for arithmetic operations, logic operations, and comparison operations. The output of the ALU will be stored in a 4-bit register, and the entire system will be synchronized using a clock signal (CLK).

This task will involve writing Verilog code for the ALU, creating a testbench to verify its functionality, and simulating the design to ensure it meets the specified requirements. The goal is to create a robust and efficient ALU that can be used as a building block for more complex digital systems.

2. 2 × 1 Multiplexer

2.1. Introduction

A 2-to-1 multiplexer (2×1 MUX) is a basic combinational circuit used in digital electronics to select one of two input signals and pass it to the output based on a control signal. It acts like an electronic switch, directing data from one of the inputs to the output depending on the value of the select line.

2.2. Working Principle

- A **2×1 MUX** has **two inputs (A and B)**, **one select line (S)**, and **one output (Y)**.
- When **S = 0**, the output **Y = A** (first input is selected).
- When **S = 1**, the output **Y = B** (second input is selected).

2.3. Circuit Implementation

A **2×1 MUX** can be implemented using basic logic gates like AND, OR, and NOT. It is widely used in data routing, signal selection, and digital system design.

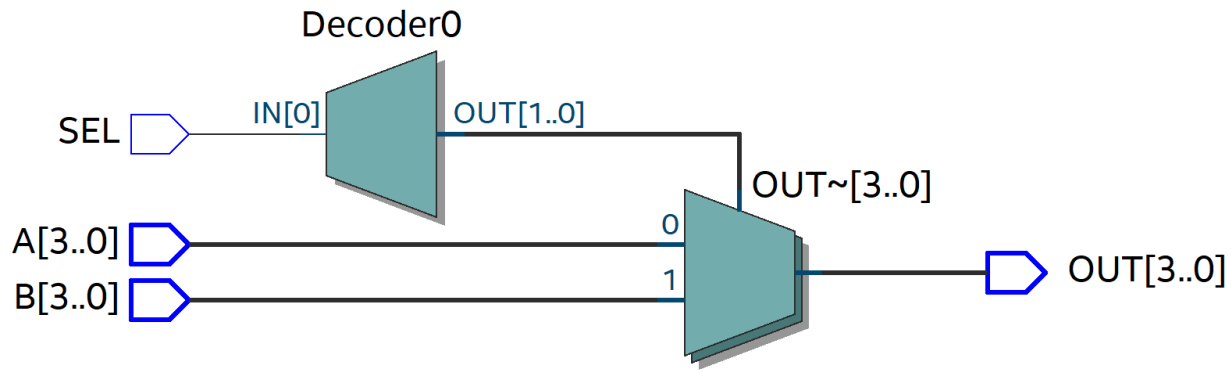
2.4. Logic Expression

he Boolean equation for a 2×1 MUX is:

$$\mathbf{Y = (A \,.\, \sim S) + (B \,.\, S)}$$

where $\overline{\mathbf{S}}$ is the complement of S.

2.5. RTL Viewer



3. 4 bit Adder

3.1. Introduction

A **4-bit adder** is a combinational digital circuit used to perform binary addition on two 4-bit numbers. It is built using four **1-bit full adders** connected in cascade, allowing it to add two 4-bit binary inputs along with a carry-in bit, producing a 4-bit sum and a carry-out.

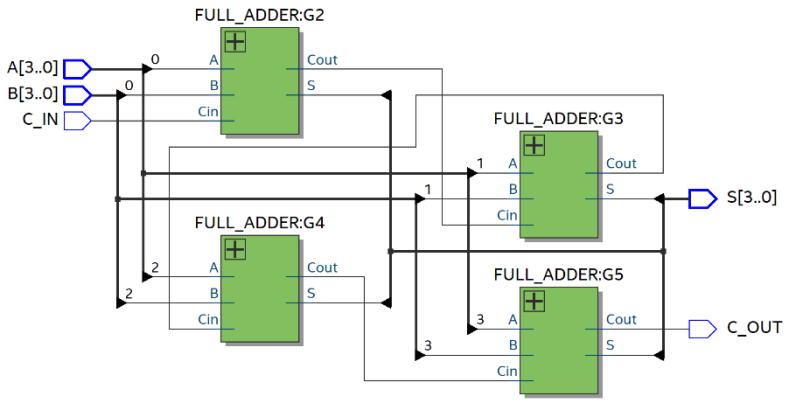
3.2. Working Principle

- It takes **two 4-bit binary numbers (A3 A2 A1 A0 and B3 B2 B1 B0)** as inputs.
- A **carry-in (Cin)** can be provided to support multi-bit addition.
- The adder produces a **4-bit sum (S3 S2 S1 S0)** and a **carry-out (Cout)** if there is an overflow.
- Each bit addition follows the rule:

$$S = A \oplus B \oplus C_{in}$$

$$C_{out} = (A \,.\, B) + C_{in} \,.\, (A \oplus B)$$

3.3. RTL Viewer



4. Arithmetic Units

4.1. Introduction

Arithmetic units are essential components of digital systems, responsible for performing fundamental mathematical operations like addition, subtraction, and two's complement conversion. These operations are commonly implemented in **Arithmetic Logic Units (ALUs)** within microprocessors and digital circuits.

4.2. Addition

The addition operation is performed using **binary adders**, such as:

- **Half Adder** (adds two single-bit numbers without carry-in).
- **Full Adder** (adds two bits plus a carry-in, used in multi-bit adders).
- **Ripple Carry Adder** or **Carry Lookahead Adder** (used for multi-bit addition).

4.3. Subtraction

Subtraction in digital systems is typically performed using the **two's complement method** instead of direct subtraction. A binary subtractor can be:

- **Half Subtractor** (handles single-bit subtraction).
- **Full Subtractor** (handles multi-bit subtraction with borrow handling).
- **Subtraction using Adders** (by adding the two's complement of the subtrahend).

4.4. Two's Complement Representation

Two's complement is the most common method for representing signed numbers in binary. It is used for subtraction and negative number representation. The process involves:

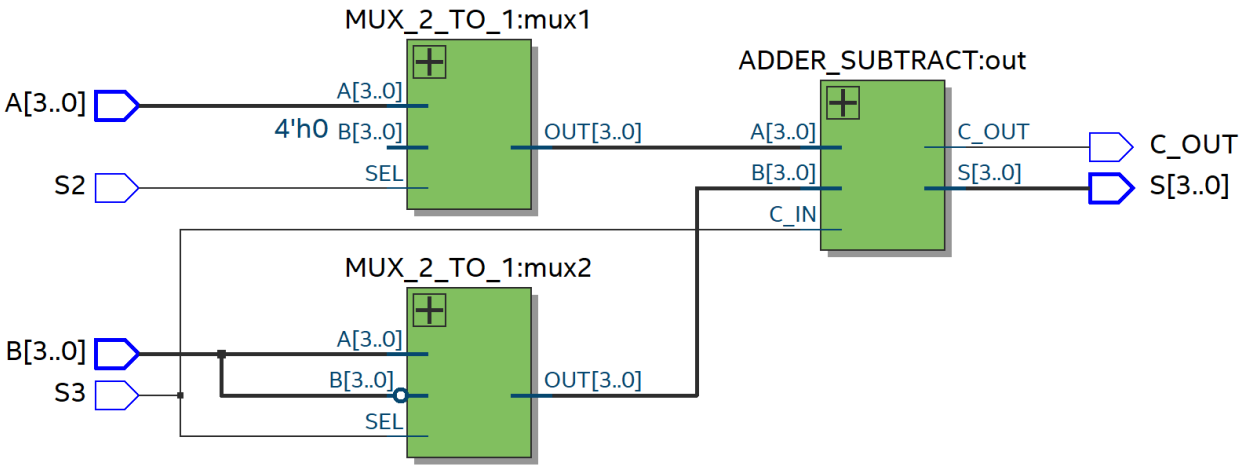
1. **Inverting all bits** (one's complement).
2. **Adding 1 to the least significant bit (LSB)**.

This technique allows subtraction to be handled using addition, simplifying arithmetic unit design.

4.5. Applications

- Used in **ALUs, microprocessors, and digital computing systems**.
- Performs essential operations in **data processing, signal processing, and embedded systems**.

4.6. RTL Viewer



5. Logic Units

5.1. Introduction

A **logic unit** is a fundamental part of a digital system that performs basic **bitwise logical operations** such as AND, OR, XOR, and NOT. It is a key component of the **Arithmetic Logic Unit (ALU)** in microprocessors and computing devices, enabling logical decision-making and data manipulation.

5.2. AND Gate

- Performs a **bitwise AND operation** on two binary inputs.
- The output is **1** only if **both inputs** are 1; otherwise, it is 0.

5.3. OR Gate

- **Performs a bitwise OR operation on two binary inputs.**
- **The output is 1 if at least one input is 1.**

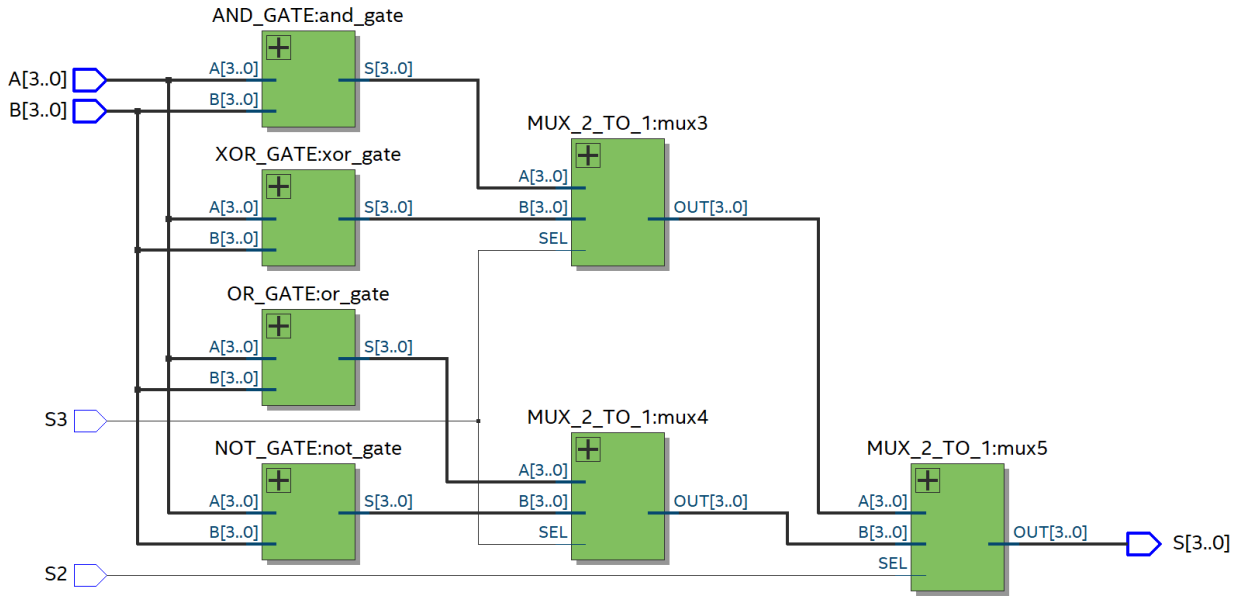
5.4. XOR Gate

- Performs a **bitwise exclusive OR operation**.
- The output is **1** if the inputs are **different** (one is 1, the other is 0).

5.5. NOT Gate

- Performs a **bitwise negation** (inverts the input).
- A **0 becomes 1**, and a **1 becomes 0**.

5.6. RTL Viewer



6. Compare Units

6.1. Introduction

A **comparator unit** is a digital circuit used to compare two binary numbers and determine their relationship. It evaluates whether one number is **less than (<), greater than (>), or equal to (=) another**. Comparator units are widely used in **processors, control systems, and digital circuits** for decision-making operations.

6.2. Equality Comparator (=)

- Checks if two binary numbers are equal.
- Outputs **1** if both inputs are identical; otherwise, it outputs **0**.
- Implemented using **XNOR gates** since XNOR produces 1 when inputs match.

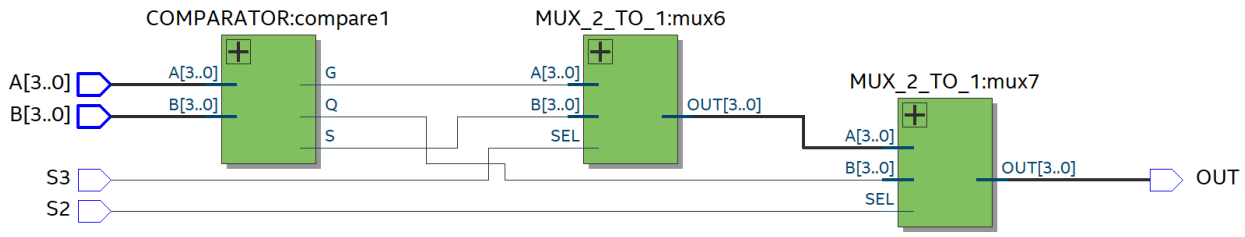
6.3. Greater Than Comparator (>)

- Compares two numbers and outputs **1** if the first number is greater than the second.
- Works by checking the most significant bit (MSB) first.

6.4. Less Than Comparator (<)

- Outputs **1** if the first number is smaller than the second.
- Similar to the greater-than comparator but checks for the opposite condition.

6.5. RTL Viewer



7. 4 bit Register

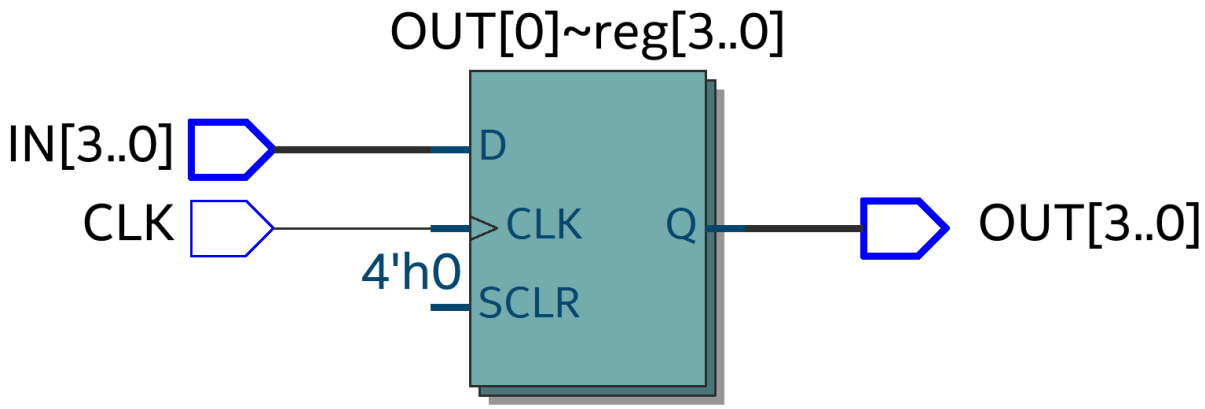
7.1. Introduction

A **4-bit register** is a sequential circuit used to store and manipulate 4-bit binary data. It consists of **four flip-flops**, each capable of holding one bit of information. Registers are essential components in digital systems, serving as temporary storage for data in microprocessors, ALUs, and memory units.

7.2. Working Principle

- The register stores a **4-bit binary value** (D3 D2 D1 D0).
- Data is **loaded on a clock pulse (CLK)**, which synchronizes storage operations.

7.3. RTL Viewer

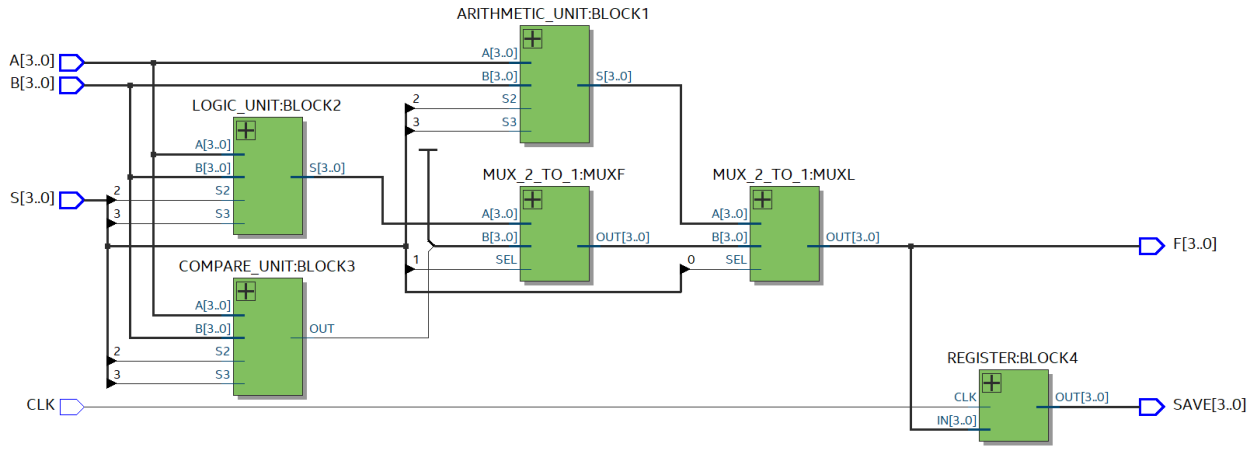


8. ALU “ Top Module “

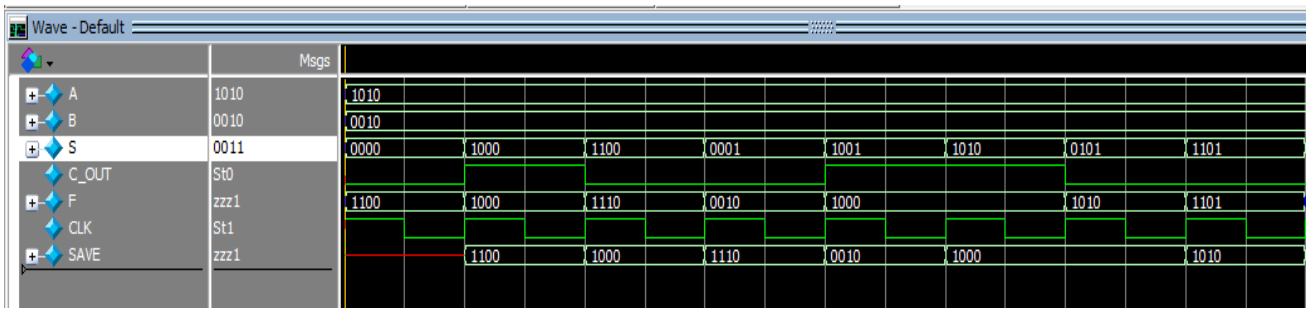
8.1. Introduction

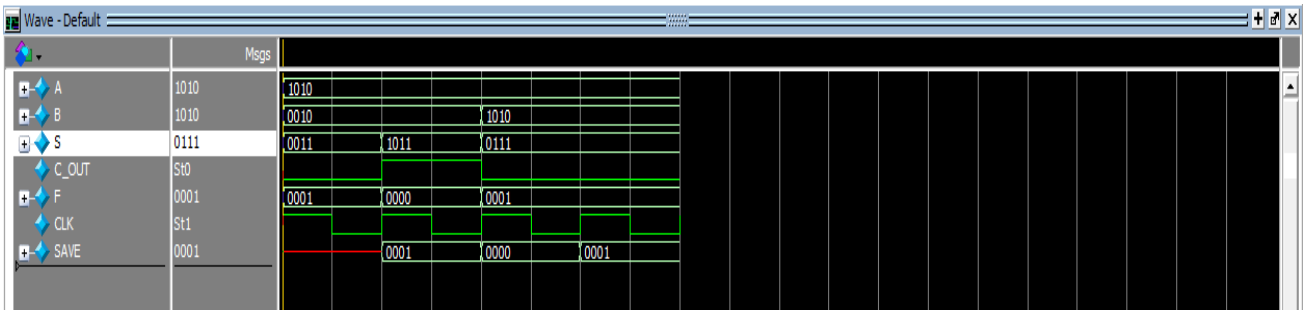
This **ALU (Arithmetic Logic Unit)** module performs arithmetic, logic, and comparison operations on 4-bit inputs. It consists of several sub-modules that execute different functions and a **register** to store results.

8.2. RTL Viewer



8.3. Wave Form



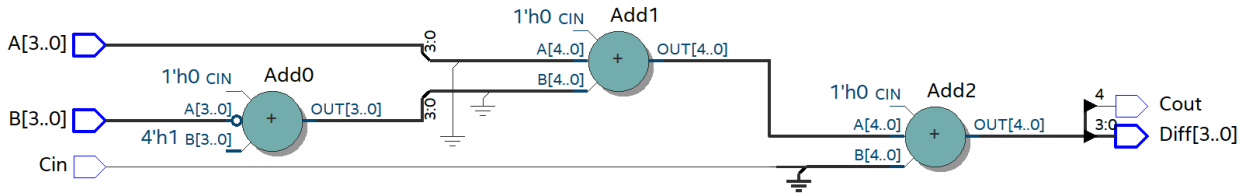


9. Subtractor

9.1. Description

In this task, we will design and implement a **subtractor** circuit using Verilog. A subtractor is a fundamental digital circuit that performs subtraction on binary numbers. It is an essential component in arithmetic logic units (ALUs) and is widely used in processors, calculators, and other digital systems.
The subtractor circuit will take two binary inputs, A and B, and produce two outputs: the difference (Diff) and a borrow-out (Bout). The borrow-out signal indicates whether a borrow is required when subtracting B from A. Depending on the design, the subtractor can be a **half subtractor** (which handles single-bit subtraction) or a **full subtractor** (which handles single-bit subtraction with a borrow-in).
The subtractor can be implemented using basic logic gates such as XOR, AND, and NOT gates. The design will be modular, allowing for easy integration into larger systems. We will also create a testbench to verify the functionality of the subtractor by testing all possible input combinations.
This task will involve writing Verilog code for the subtractor, simulating the design to ensure it meets the specified requirements, and analyzing the results. The goal is to create an efficient and reliable subtractor circuit that can be used as a building block for more complex digital systems.

9.2. RTL Viewer



10. Adder

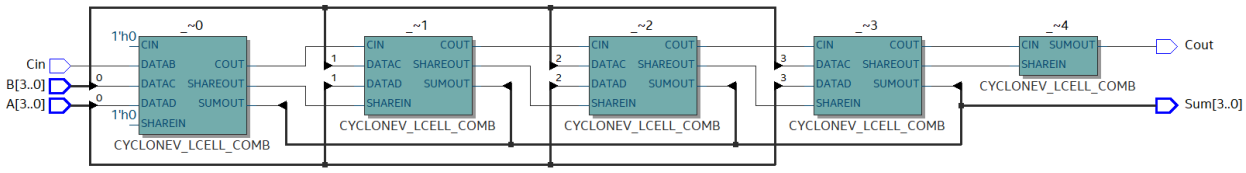
10.1. Description

An **adder** is a fundamental digital circuit used to perform binary addition. It is widely used in **arithmetic logic units (ALUs), microprocessors, and digital signal processors**. The two main types of adders are:

1. **Half Adder** – Adds two **single-bit** binary numbers but does not handle carry-in.
2. **Full Adder** – Adds two binary numbers **with carry-in**, enabling multi-bit addition.

For multi-bit operations, multiple full adders are connected in a **Ripple Carry Adder** or a **Carry Lookahead Adder** to improve speed.

10.2. RTL Viewer



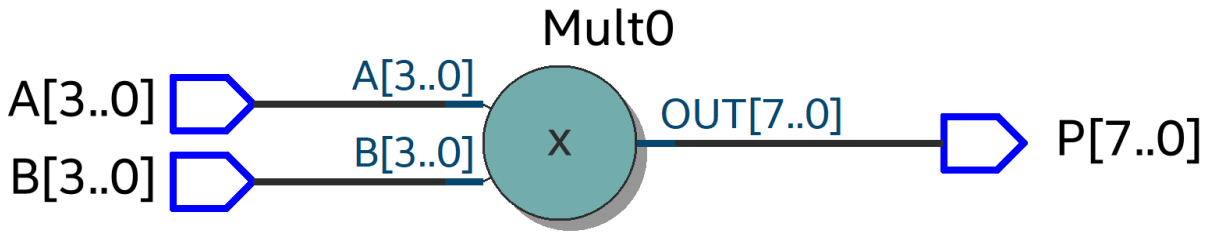
11. Multipliers

11.1. Description

A **multiplier** is a digital circuit used to perform binary multiplication. It is an essential component in **ALUs, microprocessors, and digital signal processors (DSPs)**. Multipliers can be implemented using various techniques, such as:

1. **Shift-and-Add Multiplier** – Based on iterative addition and shifting.
2. **Array Multiplier** – Uses a combinational circuit for faster multiplication.
3. **Booth Multiplier** – Efficient for signed number multiplication.

11.2. RTL Viewer



12. Decoders

12.1. Description

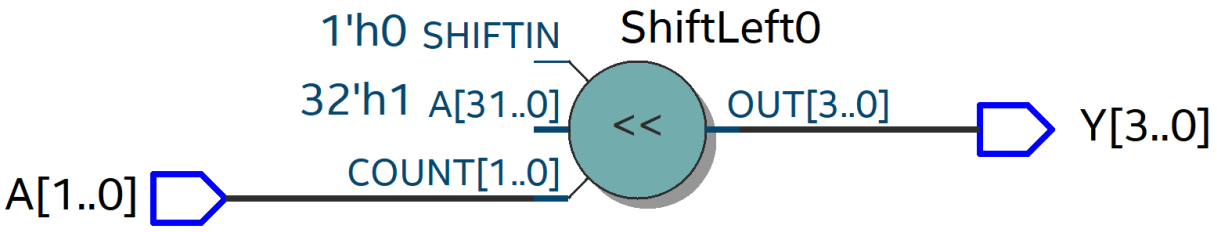
A **decoder** is a combinational circuit used to convert binary data from one format to another. It takes an **n-bit input** and generates a corresponding **2^n output**. Decoders are commonly used in applications like **memory address decoding, data multiplexing, and instruction decoding** in microprocessors.

Decoders can be categorized into:

1. **2-to-4 Decoder** – Converts 2 input bits to 4 output lines.
2. **3-to-8 Decoder** – Converts 3 input bits to 8 output lines.
3. **n-to-2^n Decoder** – Converts n input bits to 2^n output lines.

The decoder essentially activates one of the output lines based on the input value.

12.2. RTL Viewer



13. Multiplexer

13.1. Description

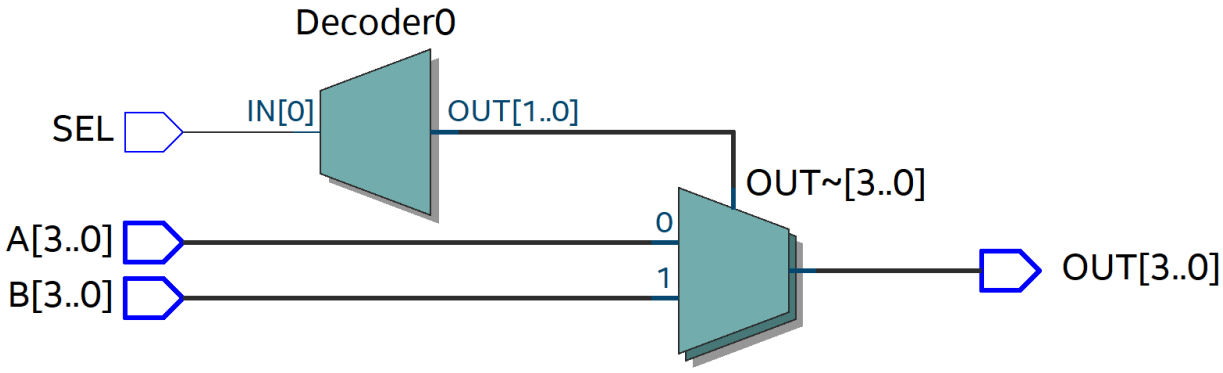
A **multiplexer (MUX)** is a digital switching device used to select one of several input signals and forward it to a single output line. It is a key component in **data routing, signal selection, and communication systems**. The MUX operates based on a set of **selection lines**, where each combination of selection lines corresponds to one of the input signals.

Common types of multiplexers:

1. **2-to-1 MUX** – Selects 1 out of 2 inputs.
2. **4-to-1 MUX** – Selects 1 out of 4 inputs.
3. **8-to-1 MUX** – Selects 1 out of 8 inputs.

The MUX allows multiple data lines to share a single resource, optimizing the use of available lines in a digital system.

13.2. RTL Viewer



14. SR Latch

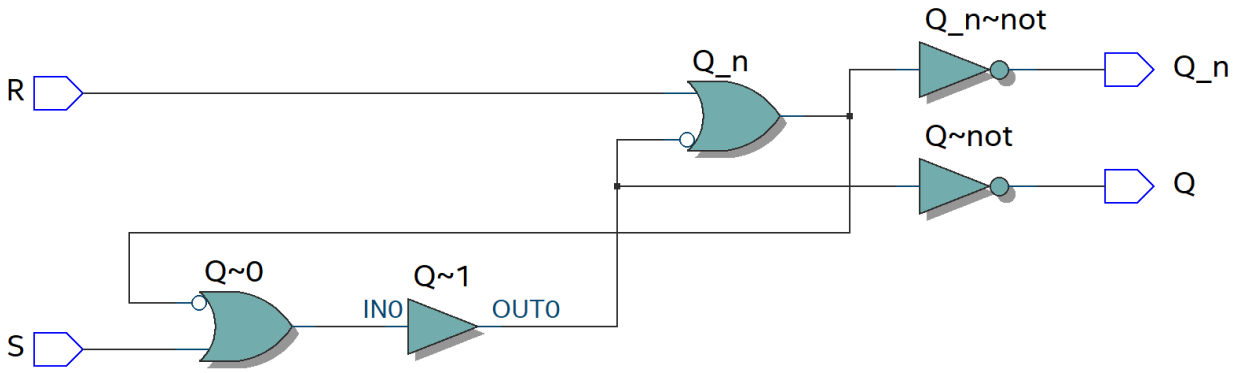
14.1. Description

An **SR latch** (Set-Reset latch) is a basic **bistable multivibrator** that has two stable states. It is a simple memory element in digital circuits that stores a single bit of data. The SR latch has two inputs: **Set (S)** and **Reset (R)**, and two outputs: **Q** and **Q'** (the complementary output). The state of the latch changes based on the inputs:

- **S = 1, R = 0**: Set the latch, **Q = 1**, **Q' = 0**.
- **S = 0, R = 1**: Reset the latch, **Q = 0**, **Q' = 1**.
- **S = 0, R = 0**: The latch **holds its previous state** (memory).
- **S = 1, R = 1**: This is an **invalid condition** because it forces both outputs to be the same, which is not allowed.

SR latches can be implemented using **NAND gates** or **NOR gates**, and they are commonly used for **data storage** and **control circuits**.

14.2. RTL Viewer



15. Flip Flop

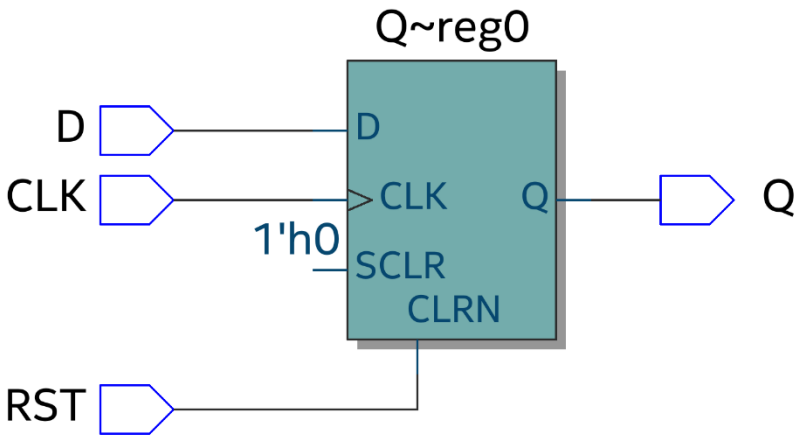
15.1. Description

A **flip-flop** is a digital memory element used to store a single bit of data. Unlike latches, flip-flops are **edge-triggered**, meaning they change states on specific transitions of the clock signal (rising or falling edge). This makes flip-flops more reliable in sequential circuits like registers, counters, and memory storage systems.

There are several types of flip-flops, but the most common ones are:

1. **SR Flip-Flop** – A set-reset flip-flop (edge-triggered version of the SR latch).
2. **D Flip-Flop** – A data or delay flip-flop, where the output follows the input.
3. **T Flip-Flop** – A toggle flip-flop, which changes state with each clock pulse.
4. **JK Flip-Flop** – A more versatile flip-flop, where both set and reset inputs can be activated.

15.2. RTL Viewer



16. Ring Counter

16.1. Description

A **ring counter** is a type of **sequential counter** where the output rotates in a loop, meaning the last output stage connects back to the first. It is often used for **sequential state cycling** or **rotation-based operations**. Ring counters are typically implemented using **flip-flops** connected in a series, where the output from the last flip-flop is fed back into the first one.
In a **n-bit ring counter**, there are **n flip-flops**, and only one flip-flop is "set" (high) at any given time. The active "1" moves from one flip-flop to the next with each clock cycle, creating a rotating pattern.
Key Features:

- The counter has **only one "1"** at a time, with all other outputs being "0".
- After **n cycles**, the pattern repeats, creating a **circular behavior**.

Ring counters are used in applications such as **control systems**, **sequential operations**, and **state machines**.

16.2. RTL Viewer