EE 224 Project Report

E5M3 Floating Point Multiplier Design

A 9-bit Floating Point Multiplier for AI Applications

Fall 2025

Team Members:

Rohit Perumal Chidhambara Babu
Madan Girish
Jade Nguyen

Abstract

This report presents the design and implementation of an E5M3 floating point multiplier intended for use in AI chips. The design utilizes a 9-bit format consisting of 1 sign bit, 5 exponent bits (in Excess-15 encoding), and 3 mantissa bits with an assumed hidden one. The multiplier is designed with a focus on high-speed operation targeting a 2GHz clock frequency, implementing saturation logic instead of error handling to maximize throughput. The design enforces a maximum fanout of 4 for all gates, with buffers strategically placed to meet timing constraints and avoid hold time violations.

1. Introduction

The E5M3 floating point format is a compact 9-bit representation designed for efficient computation in AI accelerators. This format provides a balance between precision and hardware efficiency, making it suitable for neural network inference workloads where reduced precision is acceptable.

1.1 Data Format

The 9-bit floating point format is structured as follows:

• **Sign (S):** 1 bit - Determines positive or negative value (sign-magnitude representation)

• **Exponent (E):** 5 bits - Encoded in Excess-15 (bias of 15)

• **Mantissa (M):** 3 bits - The fractional part with an assumed hidden 1 (actual mantissa is 4 bits)

1.2 Design Objectives

The primary design objectives include achieving a 2GHz clock frequency through careful pipelining, implementing saturation logic for overflow/underflow handling, and designing in complementary CMOS for optimal speed performance. A critical design rule enforced throughout is maintaining a maximum fanout of 4 for all gates to ensure signal integrity and timing closure.

2. Architecture Overview

The floating point multiplier is implemented as a 5-stage pipeline to achieve the target clock frequency. Each pipeline stage receives its inputs from the mux-based flip-flops positioned immediately before that stage. The flip-flops feature active-low reset capability and provide both true and complementary outputs. The stages perform the following operations:

Stage 1: Denormalization detection using 9-bit OR gates

Stage 2: Zero operand handling using multiplexer-based selection

Stage 3: Sign XOR, exponent carry-save addition, mantissa partial product generation

Stage 4: Exponent carry-skip addition, mantissa ripple-carry addition

Stage 5: Saturation logic and output multiplexing

2.1 Buffer Strategy

To meet timing requirements and enforce the maximum fanout rule of 4, buffers are strategically placed throughout the design:

• **Clock Buffers:** Distribute the clock signal to all flip-flops while maintaining fanout ≤ 4

• **Inverted Clock Buffers:** Provide the complementary clock signal required by the mux-based flip-flops

• **Inverted Reset Buffers:** Distribute the active-low reset signal to all pipeline registers

• **Signal Buffers:** Additional buffers are inserted in data paths to address hold time violations and to ensure no gate drives more than 4 loads

3. Basic Logic Gate Implementations

The design utilizes complementary static CMOS logic for all basic gates to optimize speed and power consumption. The following subsections detail the transistor-level implementations of the fundamental gates used throughout the design.

3.1 OR Gate

The 2-input OR gate is implemented as a NAND gate with inverted inputs, exploiting the De Morgan equivalence: (A'·B')' = A + B. This approach allows the OR function to be realized efficiently by feeding inverted signals to a standard NAND gate structure, avoiding the need for an output inverter and reducing propagation delay.

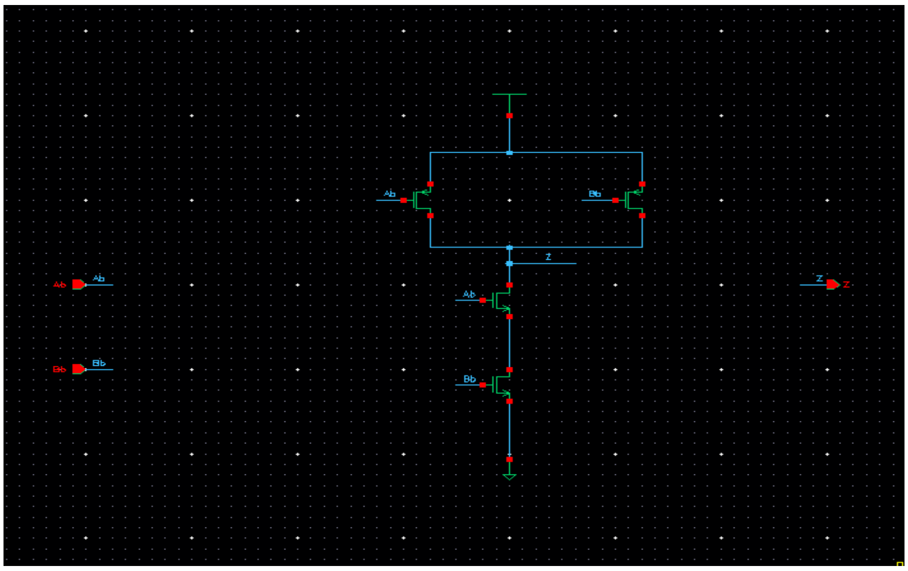


Figure 3.1: 2-Input OR Gate (NAND with Inverted Inputs)

3.2 NOR Gate

The 2-input NOR gate features parallel PMOS transistors connected to VDD and series NMOS transistors connected to ground. This configuration produces a low output when any input is high. The NOR gate is a fundamental building block used in constructing larger logic functions throughout the design.

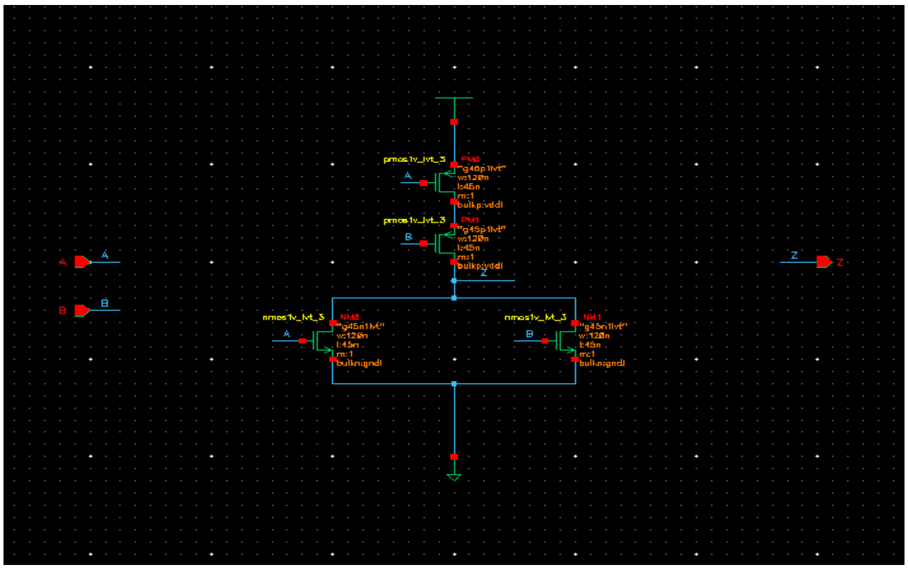


Figure 3.2: 2-Input NOR Gate Schematic

3.3 AND Gate

The 2-input AND gate is implemented as a NOR gate with inverted inputs, utilizing the De Morgan equivalence: (A' + B')' = A·B. This implementation is extensively used in the mantissa multiplication stage where partial products Am·Bn are generated. By accepting pre-inverted signals from the flip-flop complementary outputs, the AND function is achieved without additional inverter delay.

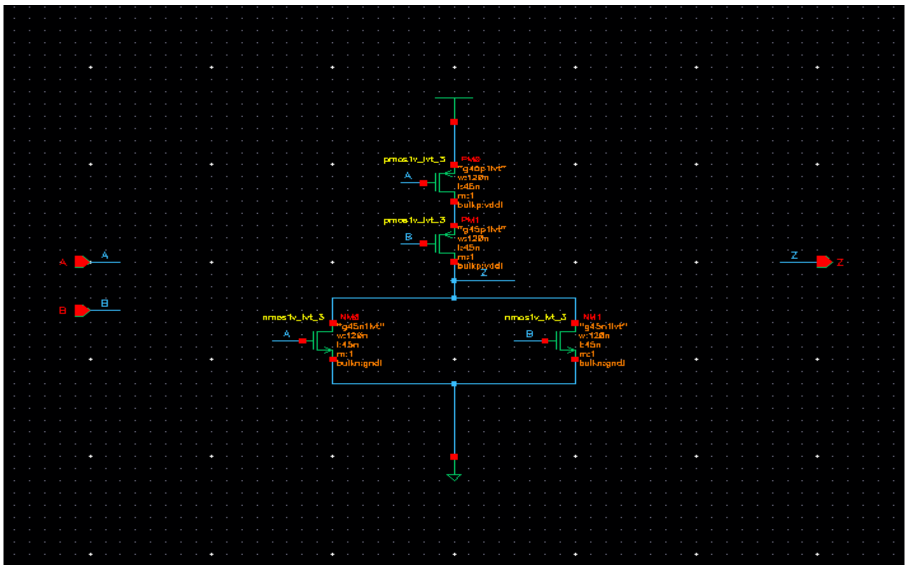


Figure 3.3: 2-Input AND Gate (NOR with Inverted Inputs)

3.4 NAND Gate

The 2-input NAND gate is implemented in standard complementary CMOS topology with series PMOS transistors in the pull-up network and parallel NMOS transistors in the pull-down network. The NAND gate produces a low output only when both inputs are high, and is used as a building block for complex logic functions.

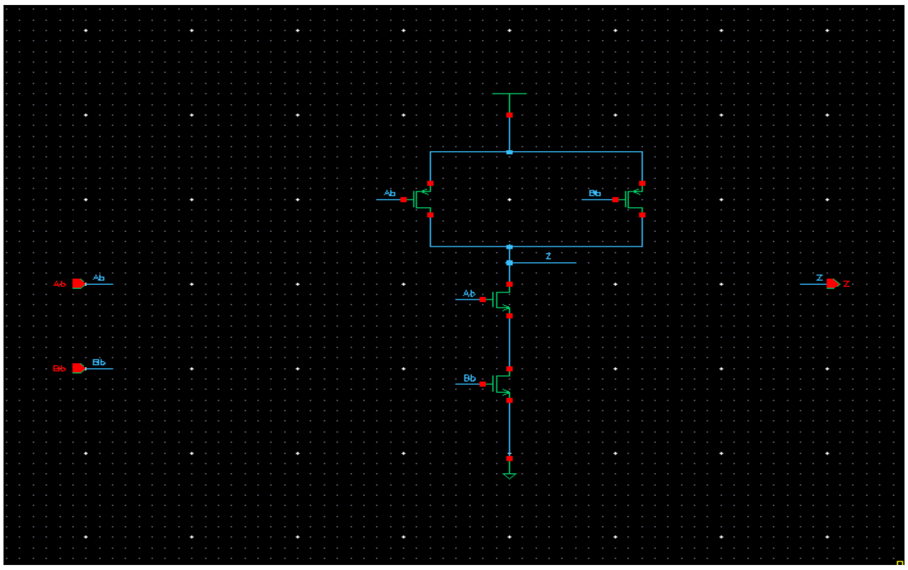


Figure 3.4: 2-Input NAND Gate Schematic

3.5 Pass Gate (Transmission Gate)

The pass gate, also known as a transmission gate, consists of a parallel combination of NMOS and PMOS transistors. When the control signal is high (and its complement is low), both transistors conduct, allowing the input signal to pass to the output with full voltage swing. Pass gates are essential components in the mux-based flip-flops and multiplexer implementations, providing low-delay signal routing with rail-to-rail output capability.

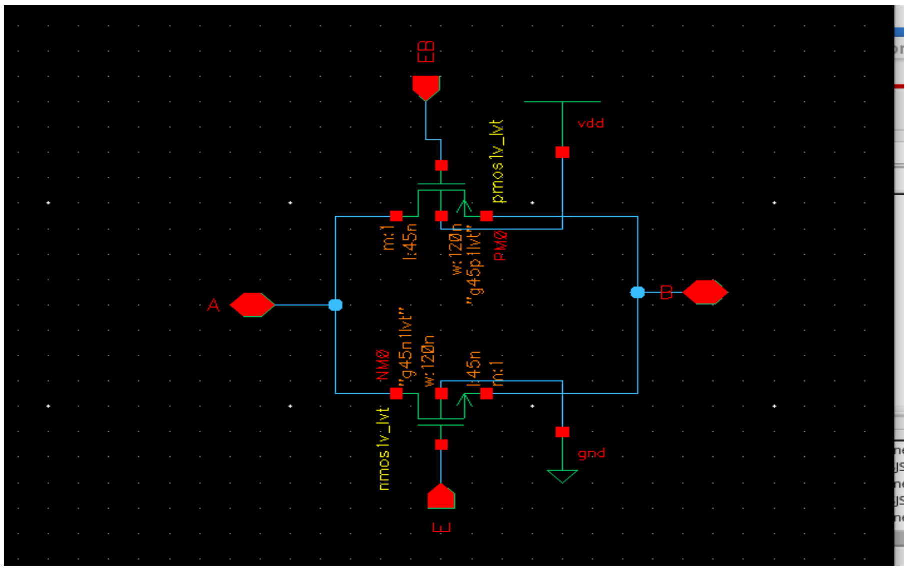


Figure 3.5: Pass Gate (Transmission Gate) Schematic

3.6 XOR Gate

The XOR gate is used for the sign bit computation in floating point multiplication. The sign of the product is the XOR of the two operand signs: Sign_result = Sign_A ⊕ Sign_B. The XOR gate is implemented using pass-gate logic for optimal speed.

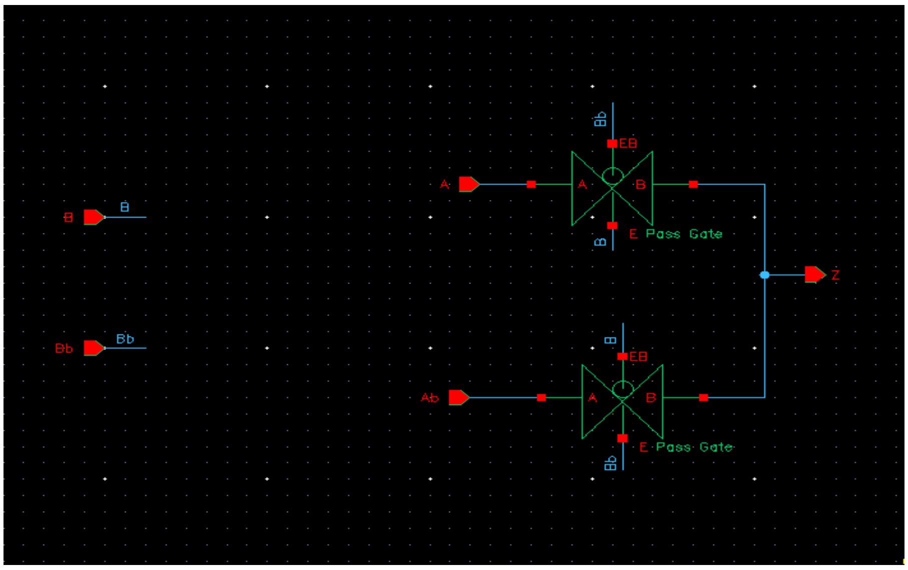


Figure 3.6: XOR Gate Schematic

4. Multi-Input Logic Gates

4.1 3-Input NOR Gate

The 3-input NOR gate extends the basic NOR topology with three parallel PMOS transistors and three series NMOS transistors. This gate is used as a building block for the 9-bit OR gate implementation, where three 3-input NOR gates feed into a 3-input NAND gate.

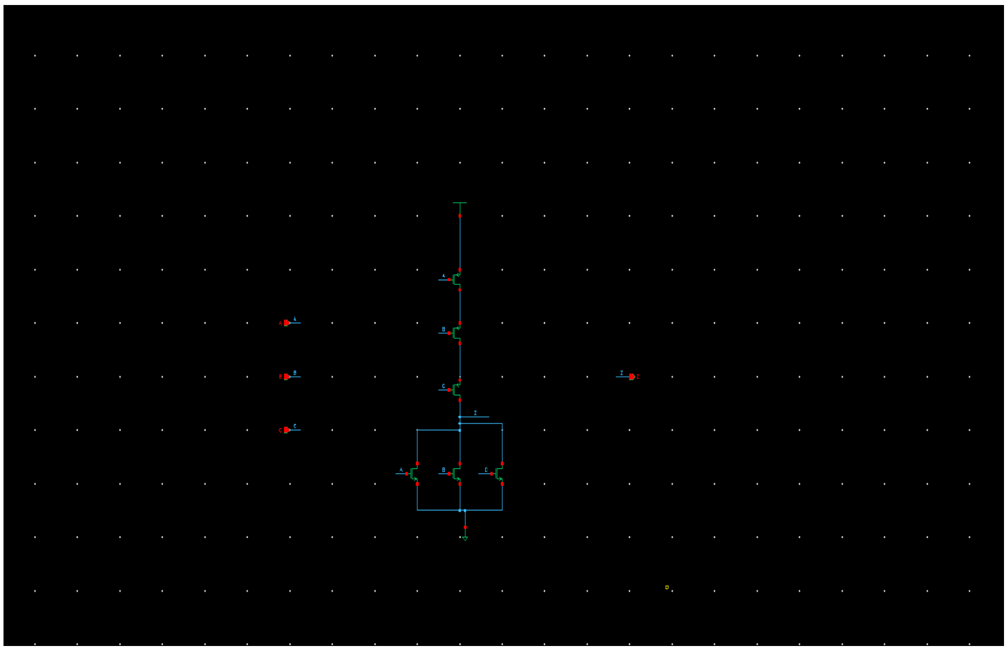


Figure 4.1: 3-Input NOR Gate Schematic

4.2 3-Input NAND Gate

The 3-input NAND gate is implemented with three series PMOS transistors and three parallel NMOS transistors. The outputs of three 3-input NOR gates are fed to this NAND gate to create the 9-bit OR function using the identity: ((A+B+C)' · (D+E+F)' · (G+H+I)')' = A+B+C+D+E+F+G+H+I.

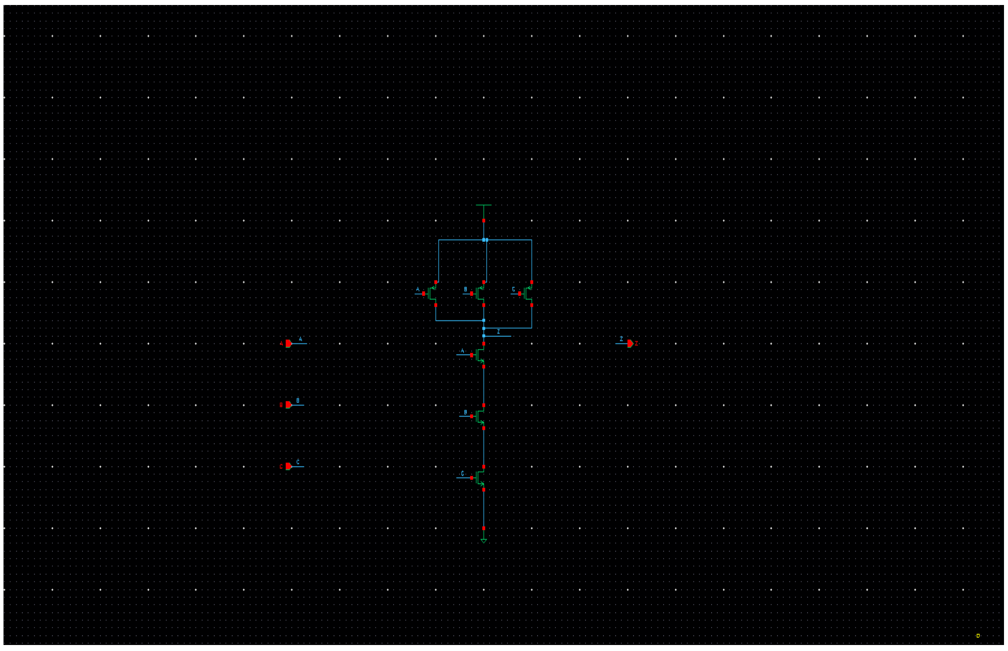


Figure 4.2: 3-Input NAND Gate Schematic

4.3 4-Input OR Gate

The 4-input OR gate is constructed using a hierarchical approach: two 2-input NOR gates process pairs of inputs, and their outputs feed into a 2-input NAND gate. This implements the logic: ((A+B)' · (C+D)')' = (A+B) + (C+D) = A+B+C+D. This structure minimizes the critical path delay while maintaining balanced loading.

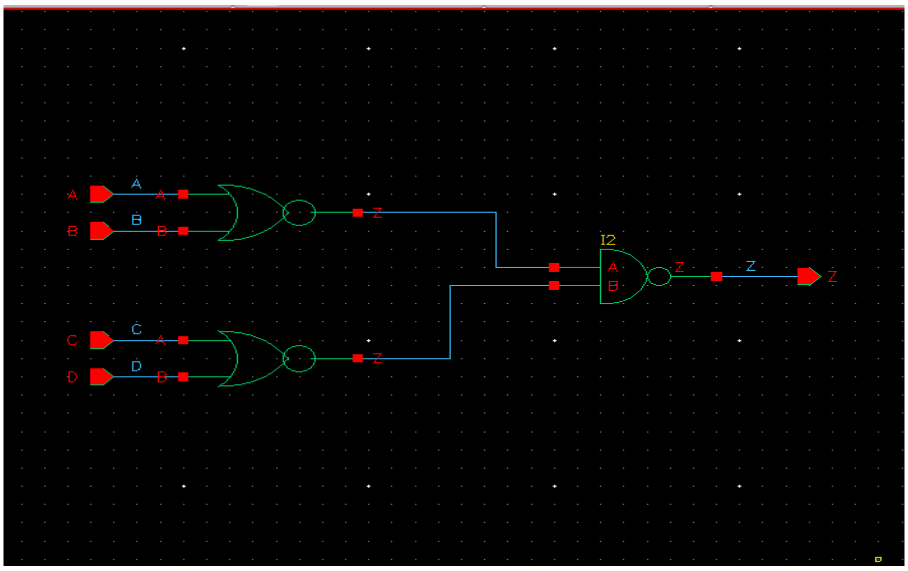


Figure 4.3: 4-Input OR Gate (Two 2-Input NORs feeding a 2-Input NAND)

4.4 4-Input AND Gate

The 4-input AND gate is implemented using complementary CMOS with four series NMOS transistors in the pull-down network and four parallel PMOS transistors in the pull-up network. This gate is critical for the saturation detection logic in the final pipeline stage.

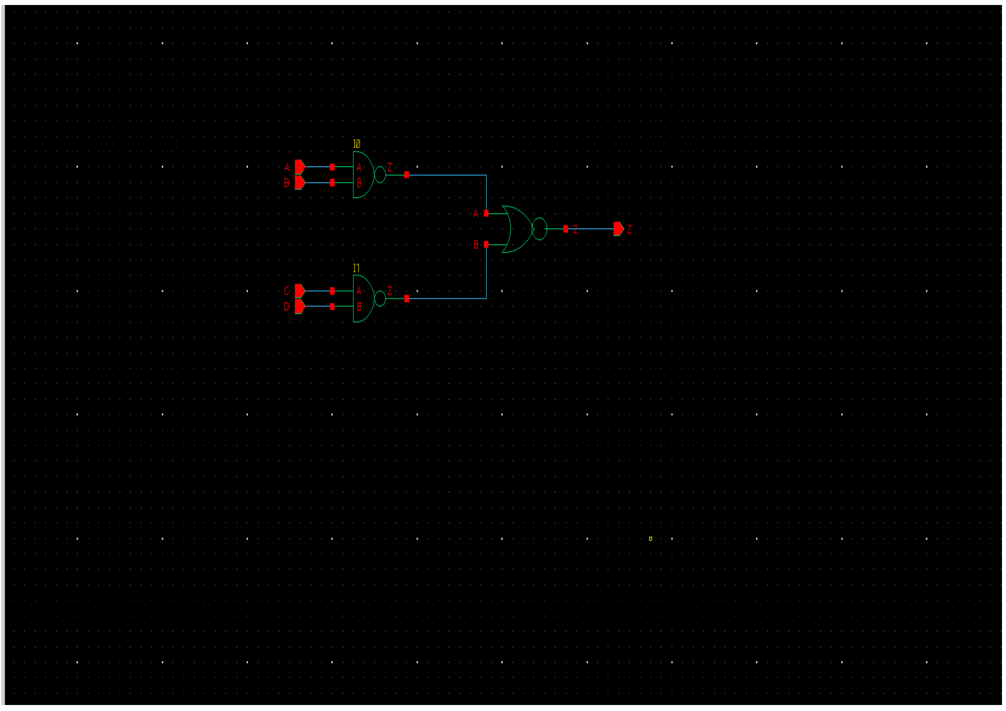
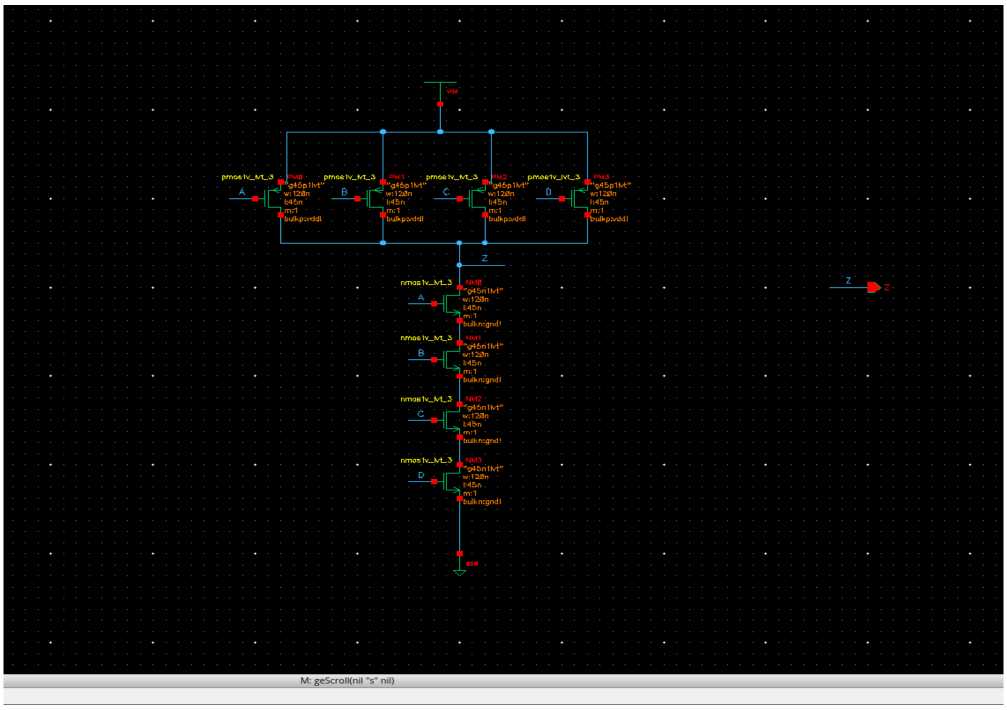


Figure 4.4: 4-Input AND Gate Transistor-Level Schematic

4.5 4-Input NAND Gate

The 4-input NAND gate provides the complemented AND function and is implemented with four parallel PMOS transistors and four series NMOS transistors. This gate is used extensively in the control logic and carry generation circuits.



M: geScroll(nil "s" nil)

Figure 4.5: 4-Input NAND Gate Transistor-Level Schematic

4.6 6-Input AND Gate

The 6-input AND gate is used for the saturation upper limit detection. It determines the S0 signal when all exponent bits indicate an overflow condition (Sign 11111 100). This gate detects when the result should saturate to the maximum representable value.

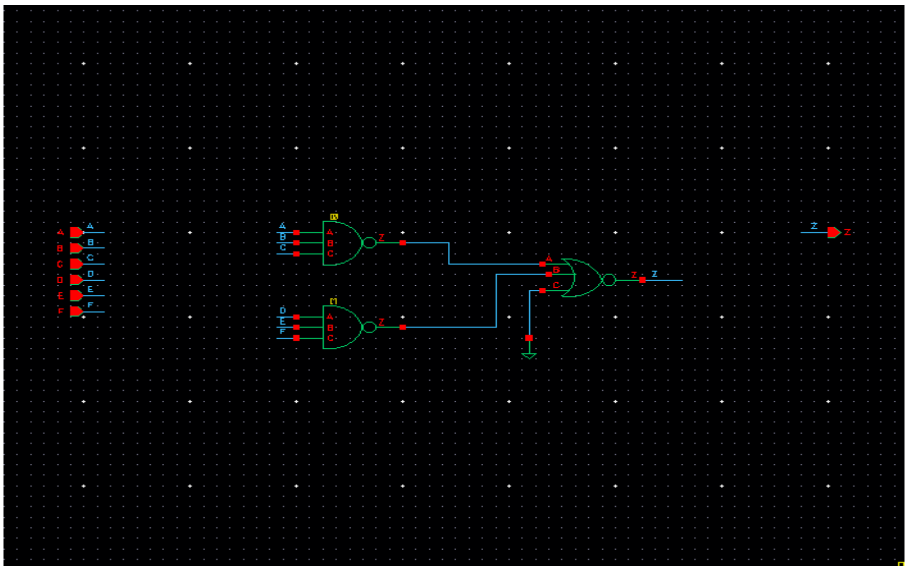


Figure 4.6: 6-Input AND Gate Schematic

5. 9-Bit OR Gate for Hidden Bit Detection

The 9-bit OR gate serves a critical function in detecting whether a floating point operand is normalized or denormalized. It is constructed by connecting the outputs of three 3-input NOR gates to a 3-input NAND gate, implementing the logic:

Z = ((A+B+C)' · (D+E+F)' · (G+H+I)')' = A+B+C+D+E+F+G+H+I

Two instances of this 9-bit OR gate are used in the first pipeline stage. Each OR gate takes the 9 input bits of one floating point operand from the flip-flops immediately preceding Stage 1, and its output determines the hidden one in the mantissa. When the output is 0 (all bits zero), the number is considered denormalized or zero.

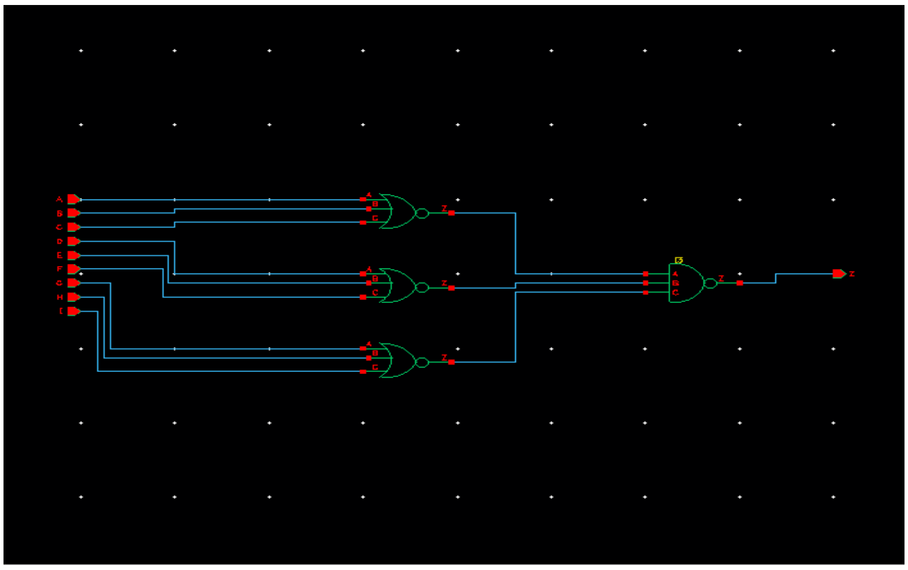


Figure 5.1: 9-Bit OR Gate Implementation (Three 3-Input NORs feeding a 3-Input NAND)

6. 3-Input Multiplexer

The 3-input multiplexer is a key component used throughout the design for data selection and saturation logic. It is implemented using pass gates (transmission gates) for efficient signal routing. The multiplexer selects one of three inputs based on two select signals (S0 and S1). The selection logic operates as follows:

S1=0, S0=0: Output = A (Normal computed result)

S1=0, S0=1: Output = Us (Upper saturation limit)

S1=1, S0=X: Output = SF (Saturation Floor - zero)

In Stage 5, this multiplexer receives inputs from the flip-flops at the stage boundary. S0 determines the upper limit saturation (when the result exceeds the maximum representable value), implemented using a 6-bit AND gate. S1 is the MSB of the exponent, indicating when the result should be saturated to zero.

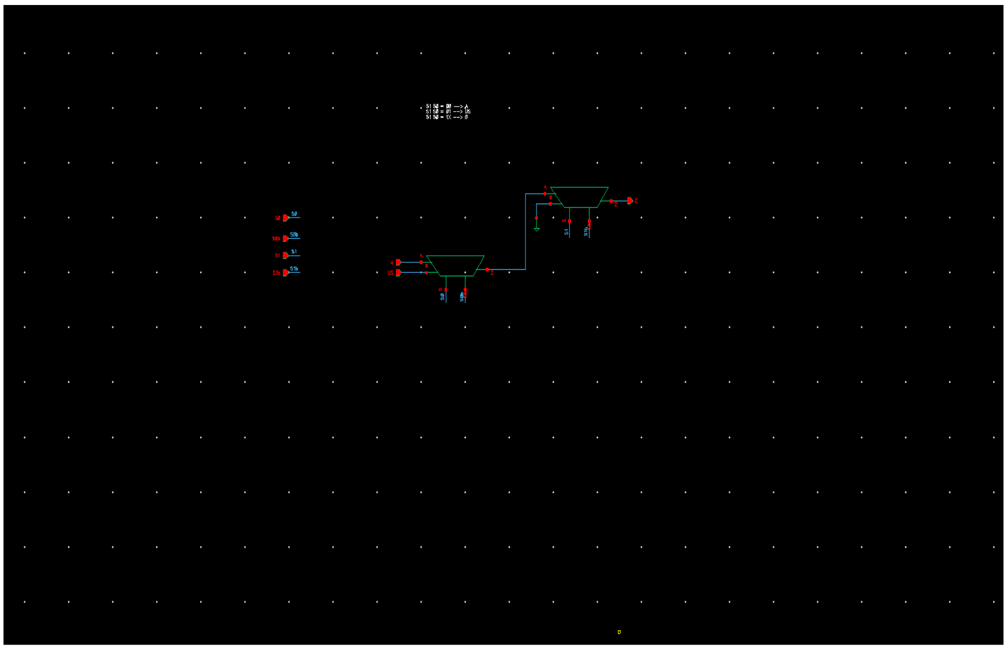


Figure 6.1: 3-Input Multiplexer Schematic using Pass Gates

7. 1-Bit Full Adder

The 1-bit full adder is the fundamental building block for both the exponent addition and mantissa multiplication operations. The design uses an optimized complementary static CMOS implementation where the logic has been simplified through factorization to minimize transistor count and propagation delay.

Unlike traditional full adder implementations that use XOR gates, this design employs factorized Boolean expressions that are directly mapped to complementary CMOS pull-up and pull-down networks. This approach reduces the critical path delay while maintaining full voltage swing at the output. The factorization exploits common sub-expressions between the Sum and Carry outputs to share transistors and reduce area.

The full adder computes Sum = A ⊕ B ⊕ Cin and Cout = AB + BCin + ACin, but the implementation achieves these functions through optimized CMOS structures rather than discrete XOR gates.

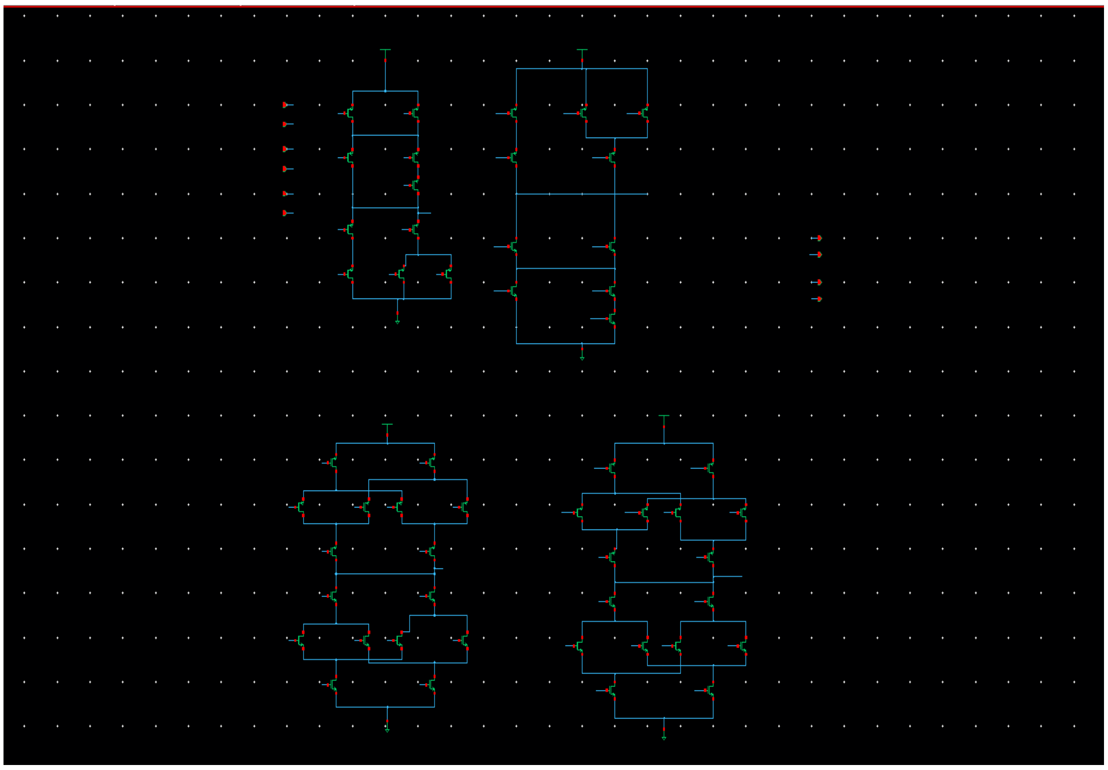


Figure 7.1: Optimized 1-Bit Full Adder (Factorized CMOS Implementation)

8. 4-Bit Ripple Carry Adder

The 4-bit ripple carry adder is constructed by cascading four 1-bit full adders, where each carry output connects to the next adder's carry input. This adder is used in Stage 4 for both the exponent carry-skip addition and the mantissa final addition. The inputs to Stage 4 come from the flip-flops positioned at the boundary between Stage 3 and Stage 4.

For the exponent path, two 4-bit ripple carry adders form a carry-skip configuration. The full adder handling the MSB receives its carry from the generate-propagate block, which calculates carry values for positions 2^1 through 2^4 using optimized carry logic.

For the mantissa path, the 4-bit ripple carry adder processes the output of the carry-save tree, producing the final 8-bit mantissa product. The bits corresponding to 2^5, 2^4, and 2^3 are then selected for the normalized result.

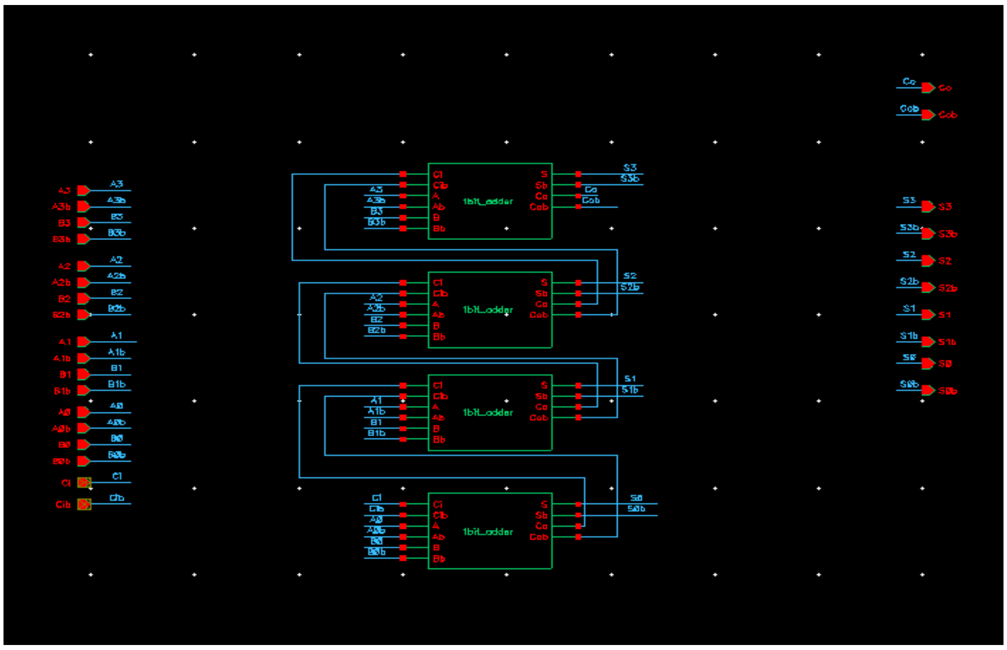


Figure 8.1: 4-Bit Ripple Carry Adder Implementation

9. Mux-Based Flip-Flop with Reset

The pipeline registers are implemented using mux-based flip-flops with an active-low reset signal. These flip-flops are positioned at the boundary of each pipeline stage, and their outputs directly feed the combinational logic of the subsequent stage. This design choice provides several advantages for the high-speed multiplier:

1. Complementary Outputs: The flip-flop provides both Q and Q' outputs, which is exploited throughout the design. The inverted outputs directly feed gates that use inverted inputs (such as the AND gate implemented as NOR with inverted inputs), eliminating additional inverter delays.

2. Low Latency: The transmission-gate-based mux structure minimizes clock-to-Q delay.

3. Active-Low Reset: Provides clean initialization of pipeline stages on power-up or system reset.

The flip-flop operates on both clock edges, with the master latch transparent when clock is low and the slave latch transparent when clock is high.

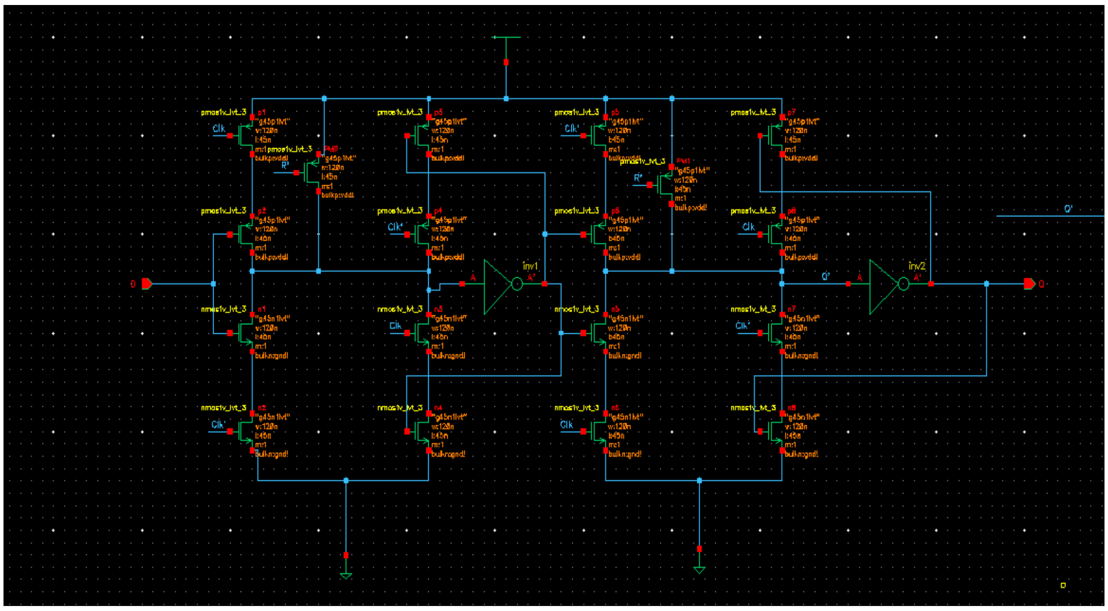


Figure 9.1: Mux-Based Flip-Flop with Active-Low Reset and Complementary Outputs

10. Pipeline Stage Implementation

Each pipeline stage receives its inputs from the mux-based flip-flops positioned immediately before that stage. The flip-flops provide both true (Q) and complementary (Q') outputs, enabling efficient implementation of gates that require inverted inputs. Clock, inverted clock, and inverted reset signals are distributed through dedicated buffer trees to maintain the maximum fanout rule of 4.

10.1 Stage 1: Denormalization Detection

Stage 1 receives the 9-bit floating point operands A and B from the input flip-flops. Two 9-bit OR gates process the input bits to determine the hidden one bit for each operand. The OR gates are constructed from three 3-input NOR gates feeding a 3-input NAND gate.

When the OR gate output is 1, the operand is normalized and has an implicit leading 1 in its mantissa. When the output is 0, all bits are zero, indicating a denormalized number or true zero, where the hidden bit should be 0. Buffers are placed on high-fanout signals to maintain the fanout ≤ 4 constraint.

10.2 Stage 2: Zero Operand Handling

Stage 2 receives the hidden bit values and operand data from the flip-flops at the Stage 1/Stage 2 boundary. The hidden one (Am<3>) serves as a select signal for a multiplexer, implementing the critical identity X × 0 = 0. If Am<3> (the hidden bit of operand A) is low, operand B is forced to zero, and vice versa.

This early zero detection and handling simplifies the downstream multiplication logic and ensures correct results for special cases without additional exception handling circuitry.

10.3 Stage 3: Initial Multiplication Operations

Stage 3 receives its inputs from the flip-flops at the Stage 2/Stage 3 boundary. The actual multiplication operations begin in three parallel paths:

Sign Computation: The sign bits are XORed to determine the result sign.

Exponent Addition: A 7-bit carry-save adder computes Exp_A + Exp_B - 15 + 1. Two extra bits are added to the MSB side as guard bits for overflow detection.

Mantissa Multiplication: A carry-save tree generates partial products. The AND gates for partial product generation (Am·Bn) are implemented as NOR gates with inverted inputs: (A'+B')' = A·B. The inverted inputs come directly from the Q' outputs of the flip-flops.

In the carry-save tree, the full adder at position (m,n) receives: Cin from Cout of adder (m-1, n-1), input A from the AND gate output (Am·Bn), and input B from the Sum output of adder (m, n-1). Buffers are inserted where needed to meet hold time requirements and maintain fanout ≤ 4.

10.4 Stage 4: Final Addition

Stage 4 receives the carry-save outputs from the flip-flops at the Stage 3/Stage 4 boundary. The addition operations are completed:

Exponent Path: The carry-save adder outputs corresponding to positions 2^1, 2^2, 2^3, and 2^4 (specifically the nth Sum and (n-1)th Carry) are fed to the carry calculator using generate and propagate logic. A carry-skip adder using two 4-bit ripple carry adders processes the outputs for positions 2^1 through 2^7. The MSB full adder receives its carry from the generate-propagate block.

Mantissa Path: A 4-bit ripple carry adder processes the final row of the carry-save tree, producing the complete mantissa product.

10.5 Stage 5: Saturation and Output Selection

Stage 5 receives the computed exponent and mantissa values from the flip-flops at the Stage 4/Stage 5 boundary. Saturation logic is implemented using 3-input multiplexers:

Mantissa Output: The mantissa bits corresponding to 2^5, 2^4, and 2^3 are fed to a 3-input multiplexer.

Exponent/Sign Output: The sign bit and 5 less significant bits of the exponent are fed to a 3-input multiplexer.

S0 (Upper Saturation): Implemented using a 6-bit AND gate, triggers when result exceeds maximum (outputs Sign=0, Exp=11111, Mantissa=100).

S1 (Zero Saturation): The MSB of the computed exponent indicates underflow, saturating to true zero (Exp=00000, Mantissa=000).

10.6 Buffer Implementation

Throughout all pipeline stages, buffers are strategically placed to ensure proper timing and signal integrity:

• **Clock Buffer Tree:** Hierarchical buffer structure distributes the clock signal to all flip-flops, ensuring each buffer drives at most 4 loads.

• **Inverted Clock Buffer Tree:** Parallel buffer structure for the complementary clock signal required by the master-slave flip-flop architecture.

• **Inverted Reset Buffer Tree:** Distributes the active-low reset signal to initialize all pipeline registers.

• **Hold Time Buffers:** Additional delay buffers inserted in short paths to prevent hold time violations, ensuring data stability at flip-flop inputs.

11. Complete Integrated Design

The figure below shows the complete integrated E5M3 floating point multiplier design as implemented in Cadence Virtuoso. This schematic integrates all the individual components described in the previous sections, including the 5-stage pipeline with mux-based flip-flops, the denormalization detection logic, zero operand handling, exponent and mantissa computation paths, and the final saturation logic.

The design shows the hierarchical organization with input flip-flops on the left, followed by the computational stages progressing to the right. The buffer trees for clock, inverted clock, and reset signals are visible throughout the design, ensuring proper signal distribution while maintaining the maximum fanout constraint of 4.

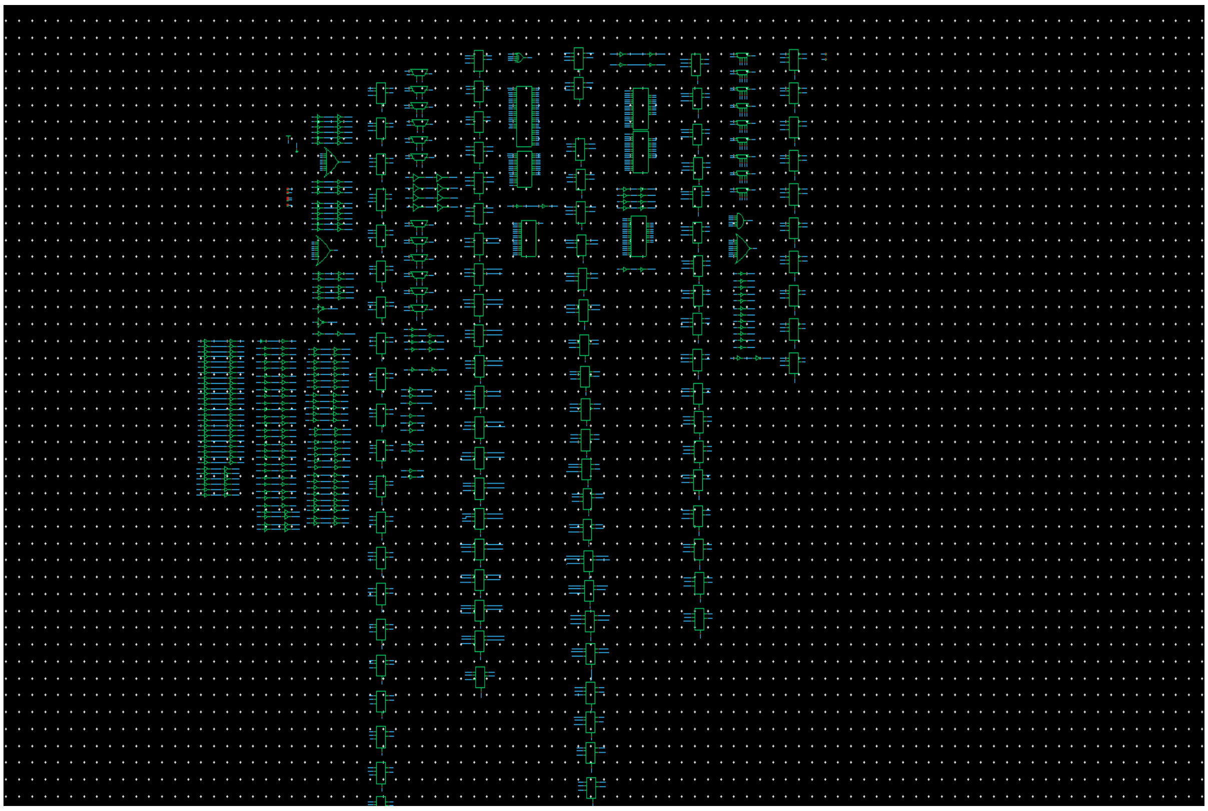


Figure 11.1: Complete Integrated E5M3 Floating Point Multiplier Design

12. Conclusion

This report presented the design of an E5M3 floating point multiplier implemented in complementary CMOS technology. The 5-stage pipeline architecture enables high-frequency operation while maintaining correct functionality for all input combinations including special cases such as zero operands and saturation conditions.

Key design decisions include the use of factorized full adders (avoiding XOR gates) in carry-save configurations, implementation of OR gates as NAND with inverted inputs and AND gates as NOR with inverted inputs to exploit complementary flip-flop outputs, mux-based flip-flops providing both Q and Q' outputs for enhanced speed, pass-gate-based multiplexers, and hierarchical gate implementations to minimize propagation delay.

The design strictly enforces a maximum fanout of 4 for all gates, with dedicated buffer trees for clock, inverted clock, and inverted reset distribution. Additional buffers are placed throughout the data paths to address hold time violations and maintain signal integrity. The saturation logic ensures graceful handling of overflow and underflow without generating exceptions, which is particularly suited for AI inference workloads where throughput is prioritized over exact IEEE compliance.

The design demonstrates the principles of floating point arithmetic implementation at the transistor level, providing insight into the trade-offs between speed, area, and functionality in modern AI accelerator designs.