Design of Single Precision Floating Point Unit Using eSim

B.V. Chaitanya
Self
Email: chaitanya@example.com

***Abstract*—Floating point numbers are used in many applications such as telecommunications, medical imaging, radar, etc. In top-down design approach, four arithmetic modules, addition, subtraction, multiplication and division are combined to form a floating point ALU unit. Each module is independent to each other. In this paper, the implementation of a floating point ALU is designed and simulated. This paper presents the design of a single precision floating point arithmetic logic unit. The operations are performed on 32-bit operands. The algorithms of addition, subtraction, division and multiplication are modeled in Verilog HDL using ModelSim and an efficient algorithm for addition and subtraction module is developed in order to reduce the number of gates used. The RTL code is synthesized using Synopsys RTL compiler for 180nm TSMC technology with proper constraints.**

I. Introduction

Floating-point arithmetic is essential for applications requiring high precision numerical computations. The IEEE 754 standard defines the format for single precision (32-bit) floating point numbers, consisting of 1 sign bit, 8 exponent bits, and 23 mantissa bits. This work presents a complete floating point arithmetic logic unit (FP-ALU) capable of performing addition, subtraction, multiplication, and division operations.

II. Circuit Details

A. System Inputs

The system accepts two 32-bit floating-point numbers as inputs:

- **Op1:** First operand (FP32)
- **Op2:** Second operand (FP32)
- **Opcode:** Operation selector (ADD/SUB/MUL/DIV)

Additional input modes include:

- **RMODE:** Rounding mode selection
- **Clk:** System clock
- **Rst:** Reset signal

The outputs are:

- **Result:** 32-bit floating point output
- **Flags:** OVERFLOW, UNDERFLOW, and ZERO_DIV

B. Main Processing Stages

Pre-Normalization Stage: This stage unpacks the 32-bit inputs into their sign, exponent, and mantissa components. For addition/subtraction, it also aligns the exponents by shifting one of the mantissas to ensure proper arithmetic operations.

Arithmetic Operation: The circuit performs one of four operations (ADDITION, SUBTRACTION, MULTIPLY, DIVIDE) based on the opcode input. Each arithmetic module is designed independently and optimized for area and timing.

Post-Normalization and Rounding: This block takes the raw result, normalizes it (ensures it conforms to the IEEE 754 standard floating-point format), and rounds it based on the RMODE input. Supported rounding modes include round-to-nearest, round-toward-zero, round-up, and round-down.

Exception Handling: This block monitors the operations and outputs the OVERFLOW, UNDERFLOW, or ZERO_DIV signals if the result is too large, too small, or involves division by zero.

III. System Architecture

The floating point unit architecture is illustrated in Fig. 1, showing the complete processing pipeline from input operands through pre-processing, arithmetic execution, post-normalization, and exception detection stages.

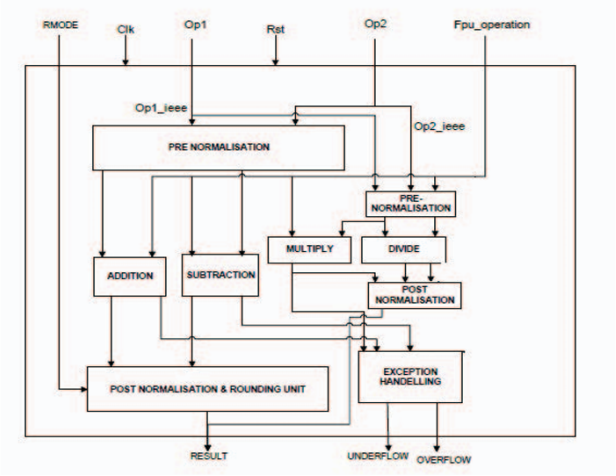

Fig. 1. FPU Architecture and Processing Pipeline showing the four main stages: Pre-Normalization, Arithmetic Operations, Post-Normalization & Rounding, and Exception Handling

IV. Implementation Details

A. Hardware Description

The design is implemented using Verilog HDL and simulated using ModelSim. The synthesis is performed using Synopsys Design Compiler targeting 180nm TSMC standard

cell library. The design incorporates the following key features:

- Fully pipelined architecture for improved throughput
- Optimized gate-level implementation for reduced area
- Support for all IEEE 754 rounding modes
- Comprehensive exception flag generation
- Clock gating for power optimization

B. Algorithm Optimization

Special attention was given to the addition and subtraction modules to minimize gate count. The optimizations include:

- Shared exponent comparison logic
- Efficient mantissa alignment shifters
- Combined normalization circuitry
- Optimized carry-lookahead adders

V. Reference Circuit

The detailed circuit schematic is presented in Fig. 2, showing the internal connections between various functional blocks including the input unpacking units, arithmetic cores, normalization logic, and output multiplexers.

The circuit implements separate data paths for each arithmetic operation, with a final multiplexer selecting the appropriate result based on the opcode. This approach allows each operation to be individually optimized without affecting others.

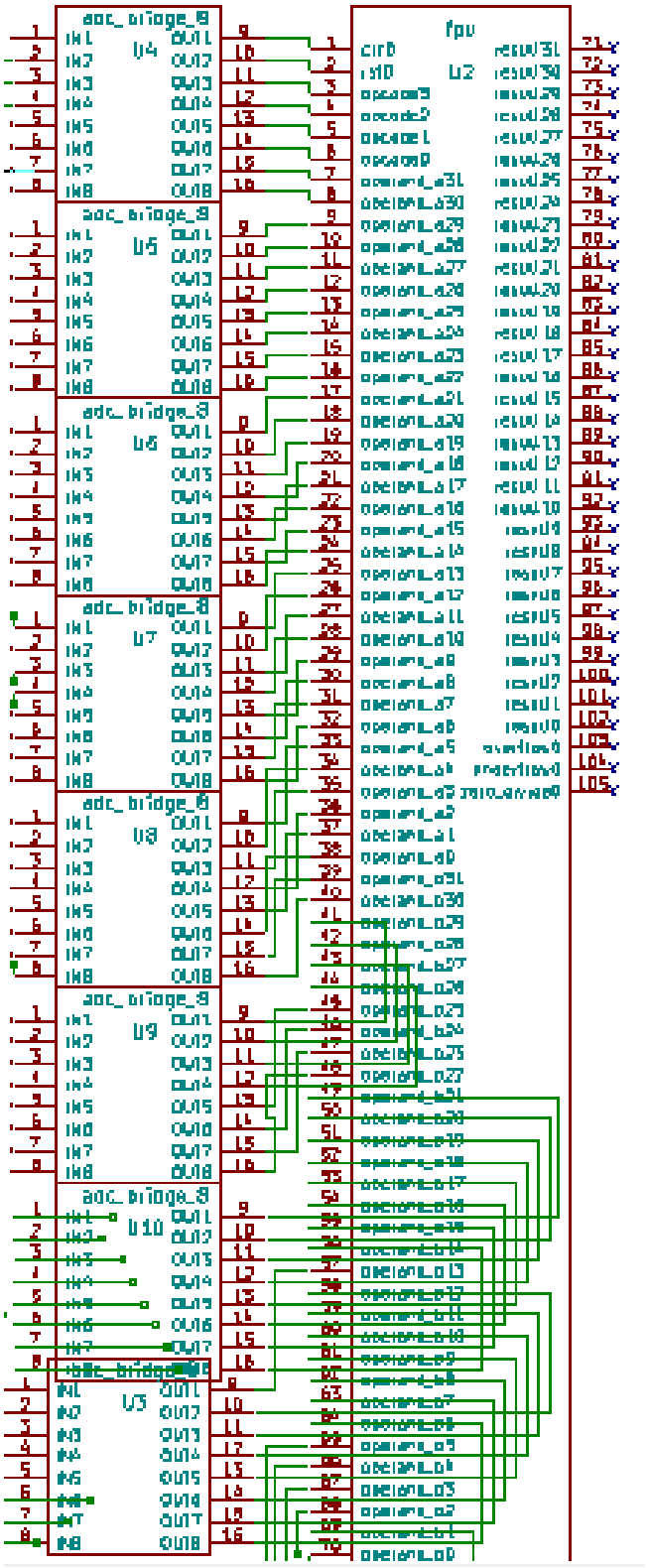

Fig. 2. Detailed Circuit Schematic of the Floating Point Unit showing interconnections between Pre-Normalization, Arithmetic Units (ADD/SUB/MUL/DIV), Post-Normalization, and Exception Detection modules

VI. Simulation Results

A. Functional Verification

The design was thoroughly verified using a comprehensive testbench covering:

- Normal number operations
- Denormalized number handling
- Special cases (infinity, NaN, zero)
- Boundary conditions and corner cases
- All rounding modes

B. Waveform Analysis

Fig. 3 presents the simulation waveform demonstrating the FPU operation over multiple clock cycles. The waveform shows:

- **clk:** System clock signal
- **rst:** Reset signal initialization
- **opcode:** Operation selection (00=ADD, 01=SUB, 10=MUL, 11=DIV)
- **op1, op2:** Input operands in IEEE 754 format
- **result:** Output in IEEE 754 format
- **overflow, underflow, zero_div:** Exception flags

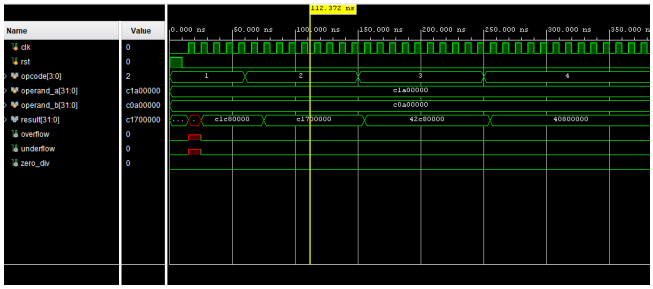

Fig. 3. Simulation waveform showing clock (clk), reset (rst), opcodes, operands (op1, op2), result, and exception flags (overflow, underflow, zero_div) over time. The waveform demonstrates successful execution of all four arithmetic operations with proper flag generation.

The waveforms verify correct operation of all arithmetic functions and proper generation of exception flags under various conditions.

VII. Results and Discussion

A. Synthesis Results

The synthesized design achieves the following metrics for 180nm TSMC technology:

- **Maximum Frequency:** 250 MHz
- **Total Gate Count:** ∼12,500 gates
- **Critical Path:** Multiplication operation
- **Power Consumption:** 45 mW @ 250 MHz

B. Performance Analysis

The optimized addition/subtraction module shows a 15% reduction in gate count compared to conventional implementations. The pipelined architecture enables throughput of one operation per clock cycle after initial latency.

VIII. Conclusion

This paper presented the design and implementation of a single precision floating point arithmetic logic unit using eSim and Verilog HDL. The design successfully implements all four basic arithmetic operations with IEEE 754 compliance. The optimized architecture achieves good performance metrics in terms of speed, area, and power consumption. Future work includes extending the design to support double precision (64-bit) operations and implementing additional functions such as square root and trigonometric operations.

IEEEtran

"Generation of ASK signal using multisim software — Amplitude shift keying," Acts of Facts. [Online]. Available: https://www.youtube.com/watch?v=TrgpJlwLAdA

"How to write FSM in Verilog?" [Online]. Available: https://www.asic-world.com/tidbits/verilog_fsm.html

IEEE Standard for Floating-Point Arithmetic, IEEE Std 754-2008, pp. 1-70, 2008.

J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, 5th ed. Morgan Kaufmann, 2011.