University of Tripoli Computer and Electronics Engineering department



Design and Implementation of Single Precision Floating-point Arithmetic Logic Unit for RISC Processor on FPGA

FPGA Design for Embedded Systems EC 582

By: Noor Alhuda Adela (022152760) Dania K. Alamen (022160629)

> Supvr: **Dr. Mohamed M. Eljhani**

> > March 2022

Summary

In this project report, we discuss the design of an arithmetic logic unit (ALU) and a floating-point unit (FPU) architecture that together performs all arithmetic and logical operations of computer processors and gives the flexibility of scalability up to 32-bits. The ALU and FPU were implemented in Verilog, simulated, and tested in ModelSim Altera.

Table of Contents

1. In	troduction	3
1.1.	Project Timeline	. 4
2. Aı	rithmetic Logic Unit Overview	4
2.1.	Design Block	.5
2.1.1.	Opcode Decoder	.5
2.1.2.	Arithmetic Block	. 7
2.1.3.	Comparator block.	.8
2.1.4.	Logical block.	. 9
2.1.5.	Shifter/Rotate block	. 9
3. Fl	oating-Point Unit Overview	10
3.1.	The IEEE 754 Standard	10
3.2.	FPU Instructions.	11
3.2.1.	Floating-Point Adder/Subtractor.	11
3.2.2.	Floating-Point Multiplication	12
4. Al	LU & FPU Synthesis Results	13
5. Co	onclusion	22

1. Introduction

This paper focuses on the design of the arithmetic logic unit (ALU) and floating-point unit (FPU), which perform integer mathematical and logic operations, and decimal mathematical operations, respectively. The end goal of this thesis is the design, Verilog implementation, and synthesis of the arithmetic logic unit and floating-point unit of a 32-bit reduced instruction set computer (RISC) processor.

The input consists of an instruction that contains an operation code (opcode), one or more operands. The operation code tells the ALU what operation to perform and the operands are used in the operation. (For example, two operands might be added together or compared logically.) The format may be combined with the opcode and tells, for example, whether this is a fixed-point or a floating-point instruction. The output consists of a result that is placed in a storage register and settings indicate whether the operation was performed successfully.

The remainder of the paper is divided into two parts - one each for the ALU and FPU. The first part describes the implementation of the functions of the arithmetic logic unit. Integer adders, multipliers, and dividers are discussed in detail, with the basic logic operations - AND, OR, XOR, etc. The second part of the paper describes the floating-point unit. Implementations of single-precision adder/subtractors, multipliers, and dividers that conform to the IEEE 754 standard are presented.



Figure 1. The general architecture of hybrid ALU



Figure 2: Flowchart

1.1 Project Timeline

We planned to follow a Bottom to up approach while designing this ALU. Firstly, we implemented simple logic blocks such as Comparator, Adder, Shifter, etc. in Verilog. Then after verifying their performance in the ModelSim simulator, with respect to generated signal patterns as inputs, we implemented the whole design.

Part I

2. Arithmetic Logic Unit Overview

An arithmetic logic unit (ALU) is an integral essential combinational circuit of any central processing unit (CPU). The processor uses the ALU to perform arithmetic operations such as addition, subtraction, multiplication, and division - as well as logical operations - OR, AND, XOR, inversion, and transformation operations. Additional operations such as data block comparisons. The ability to perform all these results of operations in the ALU is one of the most complex circuits in the CPU.

2.1. Design Block

Our design comprises 5 basic sections. We describe each one below

- Opcode Decoder
- Arithmetic Block
- Logical Block
- Comparator Block
- Shifter Block



Figure 3 . ALU Block Diagram

2.1.1. Opcode Decoder:

The proposed design consists of a primary opcode decoder unit that activates the respective function block based on the instruction opcode performed by the processor and forwarded to the ALU block. The respective output lines will feed the enable lines of respective function blocks.

The decoder is a 6:43 unit i.e., it takes 6 bits opcode as input and in turn activates the respective function block to perform the desired operation.

Selected Instructions

The designed ALU takes a clock signal, two 32-bit operands, and a 6-bit opcode as inputs. A complete list of the implemented instructions can be seen in Table 1.

S	ELE	СТ				ACTIVE-HIGH DA	ATA	
0	PCC	DDE		ARITH	IMETIC	COMPARRATOR	LOGIC	SHIFTING
				S4 S	5 =00	S4 S5 = 01	S4 S5 =10	S4 S5 =11
S0	S1	S2	S3	Cin=0	Cin=1			
0	0	0	0	OUT=A	OUT=A+1	OUT=A>B	OUT=~A	OUT= shl A
0	0	0	1	OUT=A+B	OUT=A+B+1	OUT=A < B	OUT=~(A^B)	OUT=shl B
0	0	1	0	OUT=A+`B	OUT=A+`B+1	OUT=A≠B	OUT=~A^B	OUT=shr A
0	0	1	1	OUT=-1	OUT=0		OUT=0	OUT=shr B
0	1	0	0	OUT=B	OUT=B+1		OUT=~(A∨B)	OUT=rol A
0	1	0	1	OUT=`B	OUT=`B+1		OUT=~B	OUT=rol B
0	1	1	0	OUT=A-B	OUT=A-B+1		OUT=A	OUT=ror A
0	1	1	1	OUT=B-A	OUT=B-A+1		OUT=A^~B	OUT=ror B
1	0	0	0	OUT=1	OUT=2		OUT=A ^V B	
1	0	0	1	OUT=0	OUT=1		OUT=~A^B	
1	0	1	0	OUT=A-1	OUT=A		OUT=B	
1	0	1	1	OUT=A+A	OUT=A+A+1		OUT=A^B	
1	1	0	0	OUT=B-1	OUT=B		OUT=1	
1	1	0	1				OUT=A ^{∨~} B	

Table 1. ALU Operations

Using a Combination of These operations, any logic operation can be implemented.

2.1.2. Arithmetic Block:

This block is used to implement arithmetic operations such as Addition, Subtraction, Multiplication, and division. The Accumulator and the auxiliary b register feed as inputs to this block.



Figure 4. Arithmetic Operation

Integer Addition/Subtraction

Fast addition is extremely important in many digital systems. As an elementary school child knows, addition and subtraction are the same operations. Subtraction merely inverts the sign of the second operand. Using this knowledge, it becomes simple to implement integer subtraction using any type of integer adder.

To alter the adder to be able to handle both integer addition and subtraction, logic must be established to perform two's complement conversion on the B operand if subtraction is selected, and to leave B untouched if addition is selected. This can be done in one of two ways: using an inverter and a multiplexer on each bit, or an XOR gate with one bit tied to control, and the other tied to the corresponding bit of B.

Integer Multiplication

Multiplication, like addition, is a heavily used operation that figures prominently in many types of operations. Among many other uses, multiplication is used in signal processing and scientific applications. It is also a common basis for division.

Integer multipliers can be implemented in a variety of ways. Typical implementations are a shift and add. The multiplication operation produces a 64-bit output that utilizes both registers.



Figure 5. Shift and Add Multiplier Circuit



Figure 6: Unsigned Binary Multiplication Flowchart

Integer Division

The division is the least used of the four basic arithmetic operations. As such, it has been the least researched of the four operations and remains the most difficult operation to implement efficiently.



Figure 7: Unsigned Binary Division Flowchart

2.1.3. Comparator block:

This block consists of combinational circuit HDL code which performs bit matching and comparisons. Respective outputs may be used to branch instructions based on the comparison. Similarly, numerically larger, or smaller indications on respective operands may be used.

2.1.4. Logical block:

This block comprises basic logic gate units such as AND, OR, NOT, XOR, etc. Such operations on data operands are served by this block. Outputs are stored in respective latches. The logical operations implement a series of standard logic operations on the operands at the bit level. The AND operation produces a 1 at output bit *i* only if A_i and B_i are both equal to 1. The OR operation produces a 1 at output bit *i* if A_i or B_i is B_i is equal to 1, but not both. The NOR operation is the opposite of the OR operation. A 0 is inserted at the bit position if the operating conditions are not met. The ENV operation inverts each bit of both A and B.

2.1.5. Shifter/Rotate block:

This block consists of basic shifters such as Barrel Shifters and mechanisms for bit rotation. The right shift is implemented both arithmetically - where the operation is sign-extended as it is shifted – and logically - where 0's are inserted into the bit positions that are shifted out.



Figure 8. An example for a 32 bit scaled shifter block

Part II

3. Floating-Point Unit Overview

Floating-Point Units (FPU) are the hardware components that handle decimal mathematical operations in the CPU. Like the ALU, the FPU implements the four basic mathematical operations - addition, subtraction, multiplication, and division - the difference being the number representation scheme utilized. An ALU handles integer values, represented in binary numbers. This means that the entire 32-bits of the bit vector represents the portion of a number to the left of the decimal point. An FPU deals with both the integer and fraction portions of numbers. As there is no way to slide a decimal point into the bit vectors to tell the computer what is the integer and what is the fraction, the operands must be divided into sections representing the sign, exponent, and mantissa of the number.

3.1. The IEEE 754 Standard

The standard supports single-precision 32-bit numbers, and doubleprecision 64-bit numbers. As would be expected, double-precision offers a larger range (11 exponent bits compared to 8) and greater accuracy (52 fraction bits compared to 23) than single-precision. As they operate in the same manner and the focus of the work presented in this paper is on 32-bit inputs, only the single-precision format will be explored.



Figure 9: Single-Precision IEEE 754 Format

Number	Sign	Exponent (e)	Fraction (f)
0	Х	0000000	000000000000000000000000000000000000000
∞	0	11111111	000000000000000000000000000000000000000
-∞	1	11111111	000000000000000000000000000000000000000
NAN	Х	11111111	nonzero

3.2. FPU Instructions

The instructions implemented for the floating-point unit are added, subtract, multiply and divide.

Operation	OP Code
Addition	00
Subtraction	01
Multiplication	10

Table 3.	FPU	Operations
----------	-----	------------

3.2.1.Floating-Point Adder/Subtractor

Just as integer addition/subtraction is the most used ALU operation, FP addition/subtraction is the most utilized floating-point operation.



Figure 10. Block Diagram of Floating-Point Adder/Subtractor

The difference between the exponents is used to determine the amount of right shifting necessary to align the smaller operand with the larger operand.



Figure 11: FP Addition & Subtraction Flowchart

3.2.2.Floating-Point Multiplication

Floating-point multiplication has nearly as many far-reaching applications as floating-point addition/subtraction. As a result, it is important to implement an efficient multiplier design.



Figure 12. Generic FP Multiplier Block Diagram



Figure 13: Floating Point Multiplication Flowchart

4. ALU & FPU Synthesis Results

Each of the arithmetic blocks and floating-point described previously were implemented in the Verilog hardware descriptive language. Test benches were developed in ModelSim, and executed in order to ensure that the functions performed as expected.



Figure 14: A – 1



Figure 15: A



Figure 16: A - B



Figure 17: A - B + 1

🔷 👻		Msgs																				
Opera	ands —																					
📻 — 🤣 /m	ain_tb/A	00110000011111001011101110101010	01000	001000	10111110	1001100	001100000	111110010	11101110	01010											1011011101	1
😐 🔶 /m	ain_tb/B		00111	1111000	01000011	1111101	101101110	111111000	100001010	11010												
— Cin —																						
- 🔶	_tb/Cin	0																				
— Орсос																						
• • • • • • • • • • • • • • • • • • •	opcode	0010000	10100	000	1100000		0000110				0001010				0000101		001 <mark>000</mark>	0	0010001		0010010	
Resul																						
🔶 🖭	њ/олт	000000000000000000000000000000000000000	01000	001000	11000011	001011	011110001	1111111	01111000	1111111	00110000)1111110	001100000	111110	01001000	000000	000000	00000000	10000000	0000000	0000000000	0
Statu																						
- 🔶 …	ut/ALU	St1																				
- 🧇 …	ut/FPU	St0																				
- 🔶	/ARITH	St0																				
- 🔶 …	ut/CMP	St1																				
- 🥎 ····	ut/LOG	St0																				
- 🥎 ···	/SHIFT	St0																				
- 🥎 ···	ut/Add	StO																				
- 🥎 ····	ut/Sub	StO																				
_ ?	uut/Mul	Sto																				
<u>∎</u>	D/UUT/A	00110000011111001011101110101010	01000	0001000	10111110	1001100	001100000	111110010	11101110	01010											1011011101	1
<u>∎</u> • • • • • • • • • • • • • • • • • • •	D/UUT/B	101101110111111000100001010101010	0011	1111000	01000011	1111101	101101110	111111000	100001010	11010							_	_				
±	opcode	0010000	10100	000	1100000		0000110				0001010				0000101		001000	<u> </u>	0010001		0010010	
_ 📈 …		50	01000	001000	11000011	001011	011110001		01111000		00110000	111110	001100000	11110	01001000	000000	000000	00000000	1000000	000000	000000000000000000000000000000000000000	
.		000000000000000000000000000000000000000	01000	0001000	111000011	001011	011110001		01111000		001100000	111110	001100000	111110	01001000	000000	000000			0000000		0
+	U_ALU		00000				011110001		01111000		001100000	JIIIII0	001100000	111110	01001000			000000000		0000000		0
I ≞ ∑…	/alu1/a	00110000111111001011101110101010	01000	001000	10111110	1001011	001100000			01010							=				1011011101	
	/alu1/a	101101110111111100010000101010101010	00111	1111000	01000011	1111101	101101110	1111110010	100001010	11010											1011011101	4
- - - · · ·	/oncode	0010000	10100	00	110000011		0000110		100001010	11010	0001010				0000101		001000		0010001		0010010	
A 🔜 💿	Now	1900 ns		20	0.00	40		L			1	70						u hun ni	000.00	110	lenen er	
⊖ ∕ ⊝ o	ursor 1	917 ns		30	0118	40	7 115	500	115	600	011S	700	7 118	800	7115	900	917 ns		000 115		ons	

Figure 18: A > *B*

💫 🗸	Msgs																				
Operands																					
🚛	00110000011111001011101110101010	01000	0001000	10111110	1001100	001100000	11111001	11101110	01010											10110111	011
	10110111011111100010000101011010	0011	1111000	01000011	1111101	101101110	11111100	10000101	11010												
Cin																					
🔶tb/Cin	0																				
Opcode	-																				
opcode	0010001	10100	000	1100000		0000110				0001010				0000101		0010000		00100	01	0010010	کاز
Result																					
🗉 🔷tb/ООТ	100000000000000000000000000000000000000	01000	0001000	11000011	001011	011110001	1111111	01111000	1111111	00110000)111110	001100000	111110	01001000	1000000	00000000	000000	10000	0000000000	00000000	000
Status																					
🔶ut/ALU	St1																				
Iut/FPU	St0					ļ												<u> </u>			
I/ARITH	St0																				
ut/CMP	St1																				
structure 🔶 🔶	St0																				
shift 🤣 🖓	St0																				
Iut/Add	St0																				
ut/Sub	Sto																	┼ ┥			
struct/Mul	St0																				
····b/uut/A	00110000011111001011101110101010	01000	0001000	[10111110	1001100	001100000	11111001	11101110	01010									=		10110111	011
b/uut/B	10110111011111100010000101011010	0011	1111000	01000011	1111101	101101110	11111100	10000101	011010												
• opcode	0010001	10100	000	11100000		0000110				0001010				0000101		0010000		00100	01	0010010	
uut/Cin	Stu																				
+	100000000000000000000000000000000000000	01000	0001000	1110000110	001011	011110001	1111111	01111000		00110000	01111110	001100000	1111110	01001000	1000000	000000000	0000000	10000	0000000000	000000000000000000000000000000000000000	000
+	100000000000000000000000000000000000000	00000	000000000	000000000		011110001	111111	01111000		001100000	01111110	001100000	1111110	01001000	1000000	000000000	0000000	10000	0000000000	000000000000000000000000000000000000000	000
	000000000000000000000000000000000000000	01000	0001000	110000110	001011	0000000000	000000000	000000000	00000									=			
+	10110000011111001011101110101010	01000	0001000	10111110	1001100	001100000	11111001	011101110	101010									=		,10110111	011
+/alu1/b	101101110111111000100001010101010	0011	1111000	01000011		101101110	11111100	100001010	11010							0010000				2010010	
Now	1900 ns		di na mana	I	human					humun		humun			hanna	l	harman			1	hum
A Curror 1	1015 pp		30	0 ns	40) ns	50) ns	60	0 ns	70	0 ns	80) ns	90	0 ns	100	10 ns	11	J0 ns	
ARE Now	1900 ns		30	0 ns	40	Ons	50	Ons	60	0 ns	70	0 ns	80) ns	90	0 ns	100)0 ns	11)0 ns	

Figure 19: A < *B*

∕ ⊇.+	Msgs																			
Operands																				
	101101110111111000100001010101010	00110000011	111001011101110	101010											1011	0111011	1111000	100001010	011010	
	101101110111111000100001010101010	10110111011	111100010000101	011010																
Cin																				
tb/Cin	0				1				1											
Opcode																				
opcode	0010010	0000110			0001010				0000101		0010000		0010001		0010	010		0100001		0101001
Result																				
🗉 🔷tb/ОИТ	000000000000000000000000000000000000000	01111000111	1111 01111000	1111111	00110000	1111110	001100000	111110	01001000	000000	000000000	000000	10000000	000000	0000	0000000	0000	010010001	000000	00000
Status																				
Iut/ALU	St1																			
Iut/FPU	St0	┓																		
III ARITH	St0																			
Int/CMP	St1																			
struct/LOG	StO																			
I/SHIFT	StO																			
International Add 🔶	StO																			
Int/Sub	StO																			
Iuut/Mul	St0																			
🖅 💭b/uut/A	101101110111111000100001010111010	00110000011	1110010111101110	101010											101	0111011	1111000	100001010	11010	
b/uut/B	101101110111111000100001010111010	10110111011	111100010000101	011010																
🖅 🥠opcode	0010010	0000110			0001010				0000101		0010000		0010001		0010	010		0100001		0101001
iuut/Cin	St0																			
🖅 🕂 😳 🗄 🗄	000000000000000000000000000000000000000	01111000111	<u>1111 01111000</u>	1111111	00110000)1111110	001100000	111110	01001000	000000	000000000000000000000000000000000000000	000000	10000000	0000000	0000	0000000	00001	010010001	000000	00000
U_ALU	000000000000000000000000000000000000000	01111000111	1111 01111000	1111111	00110000)1111110	001100000	111110	01001000	000000	0000000000	000000	10000000	0000000	0000	0000000	0000	010010001		00000
U_FPU	000000000000000000000000000000000000000	100000000000000000000000000000000000000	000000000000000000000000000000000000000	000000																
🛨 🛷/alu 1/a	101101110111111000100001010101010	100110000011	111001011101110	101010											101	0111011	1111000	100001010	11010	
🛨 🎣/alu1/b	101101110111111000100001010101010	10110111011	111100010000101	011010																
Nancode News	1000	10000110			10001010				10000101		10010000				100.10			1100001		0101001
Now	1900 ns	00 ns	500 ns	60	0 ns	700) ns	800) ns	900) ns	100	0 ns	110	0 ns		1200) ns	130	0 ns
🝵 🌽 🤤 Cursor 1	1123 ns														112	3 ns				

Figure 20: $A \neq B$

- {	Msgs																			
Operands																				
🖅 🔶 /main_tb/A	10110111011111100010000101011010	0011000001	11110010	110111010	1010							10110111	111111100	10000101	011010	1				001100
	1011011101111110001000010101011010	1011011101	11111000	1000010101	1010															í – –
Cin																				
tb/Cin	0																			
Opcode																				
	0101001	0001010				0000101		0010000		0010001		0010010		0100001	j.	101001		0100111		0110000
Result																				
🕀 🕂 🕁 🕞	000000000000000000000000000000000000000	001100000	111110	001100000	111110	01001000	000000	000000000	000000	100000000	000000	. 000000000	000000	01001000	1000000 🕻	000000	000000000	000000000	00000	011000
Status																				
🔶ut/ALU	St1															1				
Iut/FPU	St0															_				
	St0															_				
strate the second secon	St0													i						
stut/LOG	St1															1				j
I/SHIFT	St0																			
	St0															_				
🔶ut/Sub	St0																			
	St0															_				
😐 🍫b/uut/A	101101110111111000100001010111010	0011000001	11110010	11101111010	1010							101101110	11111100	10000101	011010					001100
🖃 🌧b/uut/B	10110111011111100010000101011010	1011011101	11111000	1000010101	1010											1				
🖃 🤹opcode	0101001	0001010				0000101		0010000		0010001		0010010		0100001	j.	101001		0100111		0110000
iuut/Cin	St0	7				1														
🛨 🕁ut/OUT	000000000000000000000000000000000000000	001100000	111110	001100000	111110	01001000	000000	000000000	000000	100000000	000000	. 000000000	000000	01001000	1000000 0	000000	000000000	000000000	00000	011000
U_ALU	000000000000000000000000000000000000000	001100000	111110	001100000	1111110	010010001	000000	000000000	000000	100000000	000000	. 000000000	000000	01001000	1000000 (000000	000000000	000000000	000000	011000
U_FPU	000000000000000000000000000000000000000	0000000000	00000000	0000000000	0000															
庄 🍫/alu1/a	101101110111111000100001010101010	0011000001	11110010	11101111010	1010							101101110	11111100	10000101	011010					001100
😐 🛷/alu1/b	101101110111111000100001010101010	1011011101	11111000	1000010101	1010															
Incode	0101001	10001010				10000101		0010000		0010001		0010010		0100001		101001		0100111		10110000
Now Now	1900 ns)0 ns	70) ns	80	0 ns	900) ns	100	0 ns	11	00 ns	120	0 ns	1300	n <mark>s</mark>	140	0 ns	150	J0 ns
🗟 🎸 🤤 Cursor 1	1313 ns														13	13 ns				

Figure 21: ~*A*&*B*

💊 🗸	Msgs																			
Operands																				
🖅 🔶 /main_tb/A	101101110111111000100001010101010	001100000111	111001011	110111010	1010							10110111	0111111000	1000	101011010					001100
	10110111011111100010000101011010	101101110111	111100010	000010101	1010															
Cin																				
🔶tb/Cin	0					1														
Opcode																				
	0100001	0001010				0000101		0010000		0010001		0010010		0100	01	0101001		0100111		0110000
Result																				
🕀 🔶 tb/ООТ	01001000100000011101111010100101	0011000001	11110	001100000	111110	01001000	000000	00000000	0000000	100000000	000000	00000000	000000	0100	.0001000000	. 00000000	000000000	0000000000	00000	011000
Status																				
Iut/ALU	St1																			
Iut/FPU	St0																			
I/ARITH	St0							<u> </u>												
ut/CMP	St0																			
struct/LOG	St1																			
SHIFT	St0																			4
🔶ut/Add	Sto													_						
ut/Sub	Sto													_						
	Stu																			
····b/uut/A	101101110111111000100001010101010	001100000111	111001011	110111010	1010							10110111	01111111000	0100C	0101011010					001100
+	1011011101111110001000010101011010	101101110111	111100010	000010101	1010															
• opcode	0100001	10001010				10000101		0010000		0010001		0010010		01000	01	0101001		0100111		0110000
	Stu																			
+	01001000100000011101111010101010101	0011000001	111100	001100000	111110	01001000	1000000	00000000	0000000	100000000	0000000	00000000	0000000	0100	0001000000.	. 100000000	000000000		00000	011000
	0100100010000001110111101010100101	0011000001			0000	101001000		,00000000	0000000	,100000000		100000000					000000000	0000000000	00000	011000
	101101110111111000100000000000000000000	000000000000000000000000000000000000000			1010									1000	101011010					001100
/alu1/a	1011011101111110001000010101011010	101100000111	111001011	000010101	1010							,10110111		1000	101011010					.001100
···/alu 1/D	0100001	101101110111		00010101	1010	0000101		0010000		0010001		0010010		0100	01	0101001		0100111		0110000
A ROW	1900 ns	dennite				lennin in	Luu uu	human	huuuu	h na	huuun	linnin	1		indiana a	diminini 1990 - S	dennen	li in		line and the
Current 1	1224 m	Juns	700 1	ns	80	UTIS	900	JINS	100	Uns	110	Uns	120	U ns	1	iuu ns	140	UTIS	150	Uns
Cursor 1	1224 fts													1224	TIS					

Figure 22: ~(*A*&*B*)

💫 🗸	Msgs	[
Operands																					
	00110000011111001011101110101010	0011000	001111100	101110	10110111	011111100	010000101	011010					001100000	111110010	11101110	101010					
	101101110111111000100001010111010	1011011	101111110	00100001	01011010																
Cin																					
🔶tb/Cin	0																				
Opcode																					
opcode	0110111	0010	0010001		0010010		0100001		0101001		0100111		0110000		0110011		0110100		011011	1	
Result																					
🕀tb/ООТ	01011011101111110001000010101101	0000	10000000	000000	00000000	0000000	01001000	.000000	000000000	000000000	000000000000000000000000000000000000000	000000	01100000	111100	01011011	011111	01100000	1111100	010110	111011	111
Status																					
2ut/ALU	St1																				
2ut/FPU	Stu																		—		
WILLIAM WILLIAM	Stu																				
	500																				
	510						-														
ut/Add	Sto																				
sut/Sub	sto																				
🧄uut/Mul	Sto																				
b/uut/A	00110000011111001011101110101010	0011000	001111100	101110	10110111	011111100	010000101	011010					001100000)111110010	11101110	101010					
🖅 🎝b/uut/B	101101110111111000100001010111010	1011011	101111110	00100001	01011010																
🖃 🌙opcode	0110111	0010	0010001		0010010		0100001		0101001		0100111		0110000		0110011		0110100		011011	1	
iuut/Cin	St0																				
🖅 🕂 😳 🖅 🗉	01011011101111110001000010101101	0000	100000000	000000	00000000	0000000	01001000	000000	0000000000	000000000	000000000000000000000000000000000000000	00000	011000001	111100	01011011	011111	01100000	1111100	010110	111011	111
U_ALU	01011011101111110001000010101101	0000	100000000	000000	00000000	000000	01001000	000000	0000000000	000000000	000000000000000000000000000000000000000	00000	011000001	1111100	01011011	011111	01100000	1111100	0101 <mark>1</mark> 0	111011	111
U_FPU	000000000000000000000000000000000000000	0000000	000000000000000000000000000000000000000	00000000	0000000																
🛨 🎝/alu 1/a	00110000011111001011101110101010	0011000	001111100	101110	10110111	0111111100	010000101	011010					001100000)111110010	11101110	101010					
🛨 🌈/alu1/b	1011011101111111000100001010111010	1011011	101111110	00100001	01011010																
A Now	1000 pc	111111	10010001		100 100 10		111111111		1111111111				10.1.10000		11111111		10110100			ntru	
INOW	1900 hs	100	0 ns	110	00 ns	120	0 ns	130	0 ns	140	0 ns	150	0 ns	160) ns	170	00 ns	180	0 ns		1900 ns
Cursor 1	1825 ns																		1825 r	S	

Figure 23: ror B

💫 🕶 Ms	js 🛛																		
Operands																			
	0 001100	000111110	0101110	10110111	011111100	010000101	011010					001100	000111111001	011101110	101010				
Hain_tb/B 10110111011111000100001010101	0 101101	110111111	000100001	01011010															
Cin																			
Opcode																			
opcode 0110000	0010	0010001		0010010		0100001		0101001		0100111		011000		0110011		0110100		0110111	
Result	-																		
на 🕹Ф/ОЛТ 0110000011111001011101110101010	0 0000	10000000	0000000	00000000	0000000	01001000	1000000	00000000	000000000	000000000	000000	011000	01111100	01011011	011111	01100000	1111100	010110111	011111
Status																			
Inut/ALU St1																			
Induction of the second																			
In ARITH Sto																			
Introduction sto																			
Induction Sto																			
St1																			
💠ut/Add St0																			
International Sto Sto																			
💠uut/Mul St0																			
b/uut/A 001100000111110010111011101010	0 001100	000111110	0101110	10110111	011111100	010000101	011010					001100	0011111001	011101110	101010				
••••••••••••••••••••••••••••••••••••••	0 101101	110111111	000100001	01011010															
opcode 0110000	0010	0010001		0010010		0100001		0101001		0100111		011000		0110011		0110100		0110111	
ut/Cin St0																			
••••••••••••••••••••••••••••••••••••••	0 0000	10000000	0000000	00000000	0000000	01001000	1000000	00000000	000000000000000000000000000000000000000	000000000	000000	011000	01111100	01011011	10111111	01100000	1111100	010110111	011111
• ····································	0000	10000000	0000000	00000000	0000000	101001000	1000000	00000000	000000000000000000000000000000000000000	000000000	000000	011000	01111100	01011011	10111111	101100000	1111100	010110111	0111111
	000000	000000000	000000000	00000000															
	001100	000111110	0101110	10110111	0111111100	010000101	011010					001100	000111111001	011101110	101010				
+	0 101101	110111111	000100001	01011010															
		1111111111	dennen	111111111	human	111111111	human	111111111		1111111111	human	011000	uluuuu	111111111		1111111111			huuuuh
19001																			

Figure 24: shl A

- 😜 🗸	Msgs															
Operands																
	010000001111110000000000000000000000000	000000000	010000	01	010000010	101111101	00110000011	11001011101	10101010					10110111011	111100010000	01011010
+	001111100100000000000000000000000000000	000000000	001111	00	001111110	010000111	10110111011	11100010000	01011010							
Cin																
tb/Cin	0										1					
- Opcode																
	1000000	0000000	1100000		11010000	11100000	0000110		0001010		0000101	0010000	0010001	0010010	0100001	0101001
- Result						1	1				1	1	1		1	
н 🔶tb/ОUТ	010000010000000100000000000000000000000	000000000	010000	10	010000010	1110000110	011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	0000000000000
- Status			1			Ĩ	1			1	1	1	1	1	1	
	St0															
🧄ut/FPU	St1 🔶															
ARITH	St0															
🧄ut/CMP	St0															
🧄ut/LOG	St0															
stift state	St0														1	
🧄ut/Add	St1 🔶															
🧄uut/Sub	St0															
🧄uut/Mul	St0															
b/uut/A	010000001111110000000000000000000000000	000000000	010000	01	010000010	101111101	00110000011	11001011101	10101010					10110111011	111100010000	01011010
	001111100100000000000000000000000000000	000000000	001111	00	001111110	010000111	10110111011	11100010000	01011010							
	1000000	0000000	100000		1010000	11100000	0000110		0001010		0000101	0010000	0010001	0010010	0100001	0101001
🦾 🦾uut/Cin	St0															
	0 100000 10000000 100000000000000000000	000000000	010000	10	010000010	1110000110	011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	000000000000
	000000000000000000000000000000000000000	00000000000	000000	00000	00000000		011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	000000000000
U FPU	0 100000 10000000 100000000000000000000	000000000	010000	10	010000010	1110000110	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000							
🚛 🎝/alu1/a	010000001111110000000000000000000000000	000000000	010000	01	010000010	101111101	00110000011	11001011101	10101010					10110111011	111100010000	01011010
	001111100100000000000000000000000000000	000000000	001111	00	001111110	010000111	10110111011	11100010000	01011010							
🗖 📥 opcode	100000	0000000	100000		1010000	11100000	0000110		0001010		0000101	001000	0010001	0010010	0100001	0101001
Arr Now	1900 ns	 ns		200	liiiiiiiiii Ins	40) ns	60) ns	80) ns	100	0 ns	120	0 ns	1400 ns
⊖ ∕ ⊖ Cursor 1	157 ns		15	7 ns				00				100		120		1100 115
		1	10.													

Figure 25: FPU Adder

Floating-point numbers

0000001	111 1100 0000 0000 0000 0000
)1111100	100 0000 0000 0000 0000 0000
xponent	Fraction
0000001	111 1100 0000 0000 0000 0000
1111100	100 0000 0000 0000 0000 0000
000001	1 111 1100 0000 0000 0000 0000
1111100	
1111100	1.100 0000 0000 0000 0000 0000
0000001	1.111 1100 0000 0000 0000 0000
1111100	1.100 0000 0000 0000 0000 0000
101 (shif	ft amount)
0000001	1.111 1100 0000 0000 0000 0000
0000001	0.000 0110 0000 0000 0000 0000 0000
0000001	1.111 1100 0000 0000 0000 0000
0000001	0.000 0110 0000 0000 0000 0000
T	10.000 0010 0000 0000 0000 0000
000004	
000001	10.000 0010 0000 0000 0000 0000 >> 1
0000010	1.000 0001 0000 0000 0000 0000
lo rounding n	ecessary)
0000010	000 0001 0000 0000 0000 0000
	0000001 1111100 xponent 0000001 1111100 0000001 1111100 101 (shi 0000001 0000001 0000001 + 0000001 1 0000001 + 0000001 1 0000001 1 0000001 1 0000001 1 0000001 1 0000001 1 0000001 1 0000001 1 0000001 1 0000001 1 0000001 1 0000001 0 0000001 0 0 0 0 0 0 0 0 0 0 0 0 0

Figure 26: Floating-point Addition

\$ 2.▼	Msgs																
Operands																	
= /main_tb,	A 0100000100011100000000000000000000000	000000000	010000001	0100	000 10	101111101	00110000011	111001011101	110101010					10110111011	111100010000	101011010	
+/main_tb	B 00111111000100000000000000000000000	000000000		0011	11110	010000111	10110111011	111100010000	101011010								
Cin	_																
	0										1						
Opcode																	
opcode	1010000	0000000	1000000	1010	000	1100000	0000110		0001010		0000101	0010000	0010001	0010010	0100001	0101001	
Result	_	-															
🕁 🔶 tb/ОО	010000010001001100000000000000000000000	000000000	0 100000 10	0100	0010	110000110	011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	000000000000000000000000000000000000000	i
Status																	
🔶ut/ALU	St0																
	St1 🔶																
	I St0											1					
	St0														1		
	St0																
	St0																
	St0			1													
	b St1 🔶					T											
	I St0]										
💶 🎝b/uut/	010000010001110000000000000000000000000	000000000	. 010000001	0100	000 10	101111101	00110000011	111001011101	110101010					10110111011	111100010000	101011010	
😐 🌧b/uut/	3 0011111100010000000000000000000000000	000000000	. 001111100	0011	11110	010000111	10110111011	111100010000	101011010								
🖃 🌧 opcode	1010000	0000000	1000000	1010	000	1100000	0000110		0001010		0000101	0010000	0010001	0010010	0100001	0101001	
shi 👍 😓 🦛	St0										1						
😐 👍ut/OU	010000010001001100000000000000000000000	000000000	. 010000010	0100	000 10	110000110	011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	00000000000000000	i
U_ALU	000000000000000000000000000000000000000	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000	000		011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	000000000000000000000000000000000000000	i
U_FPU	010000010001001100000000000000000000000	000000000	. 010000010	0100	00010	110000110	0000000000000000	000000000000000000000000000000000000000	00000000								
🖅/alu1/a	010000010001110000000000000000000000000	000000000	. 010000001	0100	00010	101111101	00110000011	1110010111101	110101010					10110111011	111100010000	101011010	
🗜 🌧/alu 1/b	001111110001000000000000000000000000000	000000000	. 001111100	0011	11110	010000111	10110111011	111100010000	101011010								
ncade	101000	000000	1000000	1010	00	1100000	0000110		000 10 10		0000101	001000	0010001	0010010	0 10000 1	0101001	
A 📰 💿 🛛 Nov	/ 1900 ns	ns	20	0 ns		40	Ons	60	0 ns	80	Oins	100	00 ns	120	0 ns	140	0 ns
🔓 🎸 🤤 Cursor	1 242 ns			24	2 ns												

Figure 27: FPU Subtraction



Floating-point numbers

Figure 28: Floating-point Subtraction

Operands																
💶 /main_tb/A		000000000	010000001	010000010	1011	11101	001100000111	11001011101	10101010					101101110111	111000100001	01011010
😐 🧄 /main_tb/B	010000111111101000100000000000000000000	000000000	001111100	001111110	0100	0111	101101110111	11100010000	01011010							
Cin																
🔶tb/Cin	0															
Opcode																
opcode		0000000	1000000	1010000	1100	00	0000110		0001010		0000101	0010000	0010001	0010010	0100001	0101001
Result																
н 🔶tb/ОUТ	11000011000101100001001100110011	000000000	010000010	010000010	1100	0110	011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	000000000000
Status																
	St0															
🧄ut/FPU	St1 🔶															
	St0															
	St0															
🔶ut/LOG	St0															
I/SHIFT	St0															
🔶ut/Add	St0															
🔶uut/Sub	St0															
🧄uut/Mul	St1 🔶															
😐 🌛b/uut/A	101111101001100110011001100110010	000000000	010000001	010000010	1011	11101	001100000111	11001011101	10101010					101101110111	111000100001	01011010
🕳 🌛b/uut/B	010000111111101000100000000000000000000	000000000	001111100	001111110	0100	0111	101101110111	11100010000	01011010							
🖬 🎝 opcode	1100000	0000000	1000000	1010000	1100	00	0000110		0001010		0000101	0010000	0010001	0010010	0100001	0101001
👍uut/Cin	St0															
💶 👍ut/OUT	11000011000101100001001100110011	000000000	010000010	010000010	1100	0110	011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	000000000000
😐 🥎 U_ALU	000000000000000000000000000000000000000	000000000000	000000000000000000000000000000000000000	00000000			011110001	011110001	001100000	001100000	010010001	000000000	100000000	000000000	010010001	0000000000000
🖅 🕂U_FPU	11000011000101100001001100110011	000000000	010000010	010000010	1100	0110	000000000000000000000000000000000000000	000000000000000000000000000000000000000	00000000							
😐 👍/alu 1/a	1011111010011001100110011001100100	000000000	010000001	010000010	1011	11101	001100000111	11001011101	10101010					101101110111	111000100001	01011010
😐 👍/alu 1/b	010000111111101000100000000000000000000	000000000	001111100	001111110	0100	0111	101101110111	11100010000	01011010							
🗖 📥 🛛 opcoda	110000	000000	1000000	1010000	11100	00	0000110		0001010		0000101	001000	0010001	0010010	0 10000 1	0101001
🚔 📰 🕤 🛛 Now	1900 ns	ns	200) ns		400	ns	600	ns	800) ns	100	0 ns	120	Ons	1400 ns
🚊 🌽 🤤 Cursor 1	338 ns				338	3 ns										

Figure 28: FPU Multiplication



Figure 30: Floating-point multiplicat

5. Conclusion

While working on this project we sought to gain practical experience and knowledge in the field of computer engineering and computer organization. Tools such as ModelSim Simulator, used in this project offer a unique way of design testing and verification by enabling signal.

REFERENCES

- 1) Stallings, William, *Computer Organization & Architecture: Designing for Performance,* 7th *ed.* Upper Saddle River, NJ: Prentice-Hall, 2006.
- 2) Computer Architecture ALU Design: Division and Floating Point, <u>http://www.ann.ece.ufl.edu/courses/eel4713_12spr/slides/Lec8-division.pdf</u>