

ESE 345: Computer Architecture

Pipelined SIMD Multimedia Unit Design

Part 1: Multimedia Arithmetic Logic Unit

Kyle Han 113799110

Summer Wang 113961753

Table of Contents

About-----	3
Load Immediate-----	3
Load Immediate Test 1-----	3
Load Immediate Test 2-----	3
Load Immediate Test 3-----	3
R4 Instructions-----	4
R4 Instruction: 000 - Integer Multiply-Add Low w/ Saturation-----	4
R4 Instruction: 001 - Integer Multiply-Add High w/ Saturation-----	5
R4 Instruction: 010 - Integer Multiply-Subtraction Low w/ Saturation-----	6
R4 Instruction: 011 - Integer Multiply-Subtraction High w/ Saturation-----	7
R4 Instruction: 100 - Long Multiply-Add Low w/ Saturation (Overflow and Underflow)-----	8
R4 Instruction: 100 - Long Multiply-Add Low w/ Saturation (Normal Operation)-----	9
R4 Instruction: 101 - Long Multiply-Add High w/ Saturation (Normal Operation)-----	10
R4 Instruction: 101 - Long Multiply-Add High w/ Saturation (Overflow and Underflow)-----	11
R4 Instruction: 110 - Long Multiply-Subtraction Low w/ Saturation (Normal Operation)-----	12
R4 Instruction: 110 - Long Multiply-Subtraction Low w/ Saturation (Overflow and Underflow)-----	13
R4 Instruction: 111 - Long Multiply-Subtraction High w/ Saturation (Normal Operation)-----	14
R4 Instruction: 111 - Long Multiply-Subtraction High w/ Saturation (Overflow and Underflow)-----	15
R3 Instructions-----	16
R3 Instruction: 0000 - NOP-----	16
R3 Instruction: 0001 - SHRHI (Test 1)-----	16
R3 Instruction: 0001 - SHRHI (Test 2)-----	16
R3 Instruction: 0001 - SHRHI (Test 3)-----	16
R3 Instruction: 0010 - AU-----	16
R3 Instruction: 0011 - CNT1H-----	17
R3 Instruction: 0100 - AHS-----	17
R3 Instruction: 0101 - OR-----	18
R3 Instruction: 0110 - BCW-----	18
R3 Instruction: 0111 - MAXWS-----	18
R3 Instruction: 1000 - MINWS-----	18
R3 Instruction: 1001 - MLHU-----	19
R3 Instruction: 1010 - MLHSS-----	19
R3 Instruction: 1011 - AND-----	19
R3 Instruction: 1100 - INVB-----	20
R3 Instruction: 1101 ROTW-----	20
R3 Instruction: 1110 SFWU-----	20
R3 Instruction: 1111 - SFHS-----	21

About

This project report shows the various testing steps and implementation strategies of the ALU. All registers have been hard coded as 00000, as the registers are not yet specified in this part of the ALU. We will be loading the registers values through the inputs Rs1, Rs2, and Rs3. Rd is our output no matter what. Our Opcode is defined here as “wordIn”, since R3 instructions technically have their own opcode.

Code

All code for this project can be found at the following github repo:

<https://github.com/Kyleh2420/Pipelined-SIMD-Multimedia-Unit>

For the sake of conciseness, I will not be including the code functions themselves in this document. All code has been written conditionally using only VHDL language that can be synthesized. The hope is to be able to synthesize and actually implement the whole project on a FPGA. For the sake of this project, we have coded the ALU behaviorally.

Load Immediate

Load Immediate Test 1

Word In: 0 001 0000 1111 0000 1111 00000

Explanation: Rd, the destination register should now have the 16 bit value defined (0x0F0F0) in the 1st index.

Simulation Result:

Signal name	Value	10	12	14	16	18	20
clk	0						
wordIn	0001000011110000111100000						19 555 ps
rs3	00000000000000000000000000000000						
rs2	000000040000000030000000040000001F						
rs1	0123456789ABCDEF0123456789ABCDEF						
rd	70123456F13579BD701234560F0F9BDF						

Load Immediate Test 2

Word In: 0 010 1100111100101111 00000

Explanation: Rd, the destination register should now have the 16 bit value defined (0xCF2F) in the 2st index.

Simulation Result:

Signal name	Value	20	22	24	26	28	30	ns
clk	0							
wordIn	0010110011110010111100000							28 921 ps
rs3	00000000000000000000000000000000							
rs2	000000040000000030000000040000001F							
rs1	0123456789ABCDEF0123456789ABCDEF							
rd	70123456F13579BD7012CF2F0F0F9BDF							

Load Immediate Test 3

Word In: 0 000 11000000000100011 00000

Explanation: Rd, the destination register should now have the 16 bit value defined (0x4023) in the 0st index.

Simulation Result:

R4 Instructions

R4 Instruction: 000 - Integer Multiply-Add Low w/ Saturation

Function: Integer Multiply-Add Low w/ Saturation

WordIn: 1 0 000 00000 00000 00000 00000

Rs1: 0x80000000100000007FFFFFFC00000003

Rs2: 0x000080000000000000007FFF00000005

Rs3: 0x00007FFF00000000000000002000000005

Explanation: Multiply the two low 16 bit values in R2 and R3 together. Then, add the 32-bit value in R1.

--3rd	--2nd	--1st	--0th
--Testing the normal operation	--Testing the normal operation	--Testing the overflow portion	--Set the 0th set of 32 bits of rs1 to 3
--Set the 1st set of 32 bits of rs1 to 0x8000_0000, the lowest 32 bit number	--Set the 1st set of 32 bits of rs1 to 0x1000_0000, the lowest 32 bit number	--Set the 1st set of 32 bits of rs1 to 0x7FFF_FFFC, the highest 32 bit number - 3	--Set the 0th set of 32 bits of rs2 to 5
--Set the 1st set of 16 bits of rs2 to 0x8000, The lowest 16 bit number	--Set the 1st set of 16 bits of rs2 to 0x0000, 0	--Set the 1st set of 16 bits of rs2 to 0x7FFF, the highest 16 bit	--Set the 0th set of 32 bits of rs3 to 5
--Set the 1st set of 16 bits of rs3 to 0x7FFF, The Highest 16 bit number	--Set the 1st set of 16 bits of rs3 to 0x0000, 0	--Set the 1st set of 16 bits of rs3 to 0x0002, 2	--That is: $(5*5) + 3 = 28$, which is 0x0_001C
--That is: $(32,767*32,768) = -1,073,709,056 - 2147483647 = \text{Saturate to } 8000_0000$	--That is: $(0*0) = 0 - 4,294,967,296 = -4,294,967,296$, or 0x1000_0000	--That is: $(32,767*2) = 65,534 + 4,294,967,296 = 4,295,032,830$, which overflowed	
		--Therefore, saturation should set it at 0x7FFF_FFFF	

Simulation Result:

Signal name	Value	100	102	104	106	108	110
clk	0						
wordIn	10000000000000000000000000000000						
rs3	00007FFF00000000000000002000000005						
rs2	000080000000000000007FFF0000000005						
rs1	80000000100000007FFFFFFC000000003						
rd	80000000100000007FFFFFFF00000001C						

R4 Instruction: 001 - Integer Multiply-Add High w/ Saturation**Function:** Integer Multiply-Add High w/ Saturation**WordIn:** 1 0 001 00000 00000 00000 00000**Rs1:** 0x80000000100000007FFFFFFC00000003**Rs2:** 0x80000000000000007FFF000000050000**Rs3:** 0x7FFF0000000000000002000000050000**Explanation:** Multiply the two high 16 bit values in R2 and R3 together. Then, add the 32 bit value in R1.

--3rd	--2nd	--1st	--0th
--Testing the underflow operation	--Testing the normal operation	--Testing the overflow portion	--Set the 0th set of 32 bits of rs1 to 3
--Set the 1st set of 32 bits of rs1 to 0x8000_0000, the lowest 32 bit number	--Set the 1st set of 32 bits of rs1 to 0x1000_0000, the lowest 32 bit number	--Set the 1st set of 32 bits of rs1 to 0x7FFF_FFFC, the highest 32 bit number - 3	--Set the 0th set of 32 bits of rs2 to 5
--Set the 1st set of 16 bits of rs2 to 0x8000, The lowest 16 bit number	--Set the 1st set of 16 bits of rs2 to 0x0000, 0	--Set the 1st set of 16 bits of rs2 to 0x7FFF, the highest 16 bit	--Set the 0th set of 32 bits of rs3 to 5
--Set the 1st set of 16 bits of rs3 to 0x7FFF, The Highest 16 bit number	--Set the 1st set of 16 bits of rs3 to 0x0000, 0	--Set the 1st set of 16 bits of rs3 to 0x0002, 2	--That is: $(5*5) + 3 = 28$, which is 0x0_001C
--That is: $(32,767 * -32,768) = -1,073,709,056 - 2147483647 = \text{Saturate to } 8000_0000$	--That is: $(0*0) = 0 - 4,294,967,296 = -4,294,967,296$, or 0x1000_0000	--That is: $(32,767*2) = 65,534 + 4,294,967,296 = 4,295,032,830$, which overflowed	
		--Therefore, saturation should set it at 0x7FFF_FFFF	

Simulation Result:

Signal name	Value	. 110 . . . 112 . . . 114 . . . 116 . . . 118 . . . 120 .
clk	0	
wordIn	10001000000000000000000000000000	X 10001000000000000000000000000000 X
rs3	7FFF0000000000000002000000050000	X 7FFF0000000000000002000000050000 X
rs2	80000000000000007FFF000000050000	X 80000000000000007FFF000000050000 X
rs1	80000000100000007FFFFFFC00000003	X 80000000100000007FFFFFFC00000003 X
rd	80000000100000007FFFFFFF0000001C	X 80000000100000007FFFFFFF0000001C X

R4 Instruction: 010 - Integer Multiply-Subtraction Low w/ Saturation**Function:** Integer Multiply-Subtraction Low w/ Saturation**WordIn:** 1 0 010 00000 00000 00000 00000**Rs1:** 0x80000000800000007FFFFFFFFF00000003**Rs2:** 0x00007FFF00007FFF00000000100000005**Rs3:** 0x00007FFF00007FFF00008000000000005**Explanation:** Multiply the two Low 16 bit values in R2 and R3 together. Then, subtract the 32 bit value in R1.

--3rd	--2nd	--1st	--0th
--Testing the Overflow operation	--Testing the Overflow operation	--Testing the Underflow portion	--Set the 0th set of 32 bits of rs1 to 3
--Testing the Overflow operation	--Set the 1st set of 32 bits of rs1 to 0x8000_0000, the lowest 32 bit number	--Set the 1st set of 32 bits of rs1 to 0x7FFF_FFFF, the highest 32 bit number	--Set the 0th set of 32 bits of rs2 to 5
--Set the 1st set of 32 bits of rs1 to 0x8000_0000, the lowest 32 bit number	--Set the 1st set of 16 bits of rs2 to 0x7FFF, the highest 16 bit	--Set the 1st set of 16 bits of rs2 to 0x0001, 1	--Set the 0th set of 32 bits of rs3 to 5
--Set the 1st set of 16 bits of rs2 to 0x7FFF, the highest 16 bit	--Set the 1st set of 16 bits of rs3 to 0x7FFF, the highest 16 bit	--Set the 1st set of 16 bits of rs3 to 0x8000, The lowest 16 bit number	--That is: $(5*5) - 3 = 22$, which is 0b1_0110
--Set the 1st set of 16 bits of rs3 to 0x7FFF, the highest 16 bit	--That is: $(32,767*32,767) = 1,073,676,289 - 2,147,483,648 = 3,221,159,937$, which overflowed	--That is: $(-32,768*1) = -32768 - 2,147,483,647 = -2,147,516,415$, which underflowed	
--That is: $(32,767*32,767) = 1,073,676,289 - 2,147,483,648 = 3,221,159,937$, which overflowed	--Therefore, saturation should set it at 0x7FFF_FFFF	--Therefore, saturation should set it at 0x8000_0000	
--Therefore, saturation should set it at 0x7FFF_FFFF			

Simulation Result:

Signal name	Value	122	124	126	128	130
clk	0					
wordIn	1001000000000000000000000000	1001000000000000000000000000				129 157 ps
rs3	00007FFF00007FFF000080000000005	00007FFF00007FFF000080000000005				
rs2	00007FFF00007FFF00000000100000005	00007FFF00007FFF00000000100000005				
rs1	80000000800000007FFFFFFFFF00000003	80000000800000007FFFFFFFFF00000003				
rd	7FFFFFFFF7FFFFFFFF8000000000000016	7FFFFFFFF7FFFFFFFF8000000000000016				

R4 Instruction: 011 - Integer Multiply-Subtraction High w/ Saturation**Function:** Integer Multiply-Subtraction High w/ Saturation**WordIn:** 1 0 010 00000 00000 00000 00000**Rs1:** 0x80000000800000007FFFFFFFF00000003**Rs2:** 0x00007FFF00007FFF00000000100000005**Rs3:** 0x00007FFF00007FFF00008000000000005**Explanation:** Multiply the two high 16 bit values in R2 and R3 together. Then, subtract the 32 bit value in R1.

--3rd	--2nd	--1st	--0th
--Testing the Overflow operation	--Testing the Overflow operation	--Testing the Underflow portion	--Set the 0th set of 32 bits of rs1 to 3
--Testing the Overflow operation	--Set the 1st set of 32 bits of rs1 to 0x8000_0000, the lowest 32 bit number	--Set the 1st set of 32 bits of rs1 to 0x7FFF_FFFF, the highest 32 bit number	--Set the 0th set of 32 bits of rs2 to 5
--Set the 1st set of 32 bits of rs1 to 0x8000_0000, the lowest 32 bit number	--Set the 1st set of 16 bits of rs2 to 0x7FFF, the highest 16 bit	--Set the 1st set of 16 bits of rs2 to 0x0001, 1	--Set the 0th set of 32 bits of rs3 to 5
--Set the 1st set of 16 bits of rs2 to 0x7FFF, the highest 16 bit	--Set the 1st set of 16 bits of rs3 to 0x7FFF, the highest 16 bit	--Set the 1st set of 16 bits of rs3 to 0x8000, The lowest 16 bit number	--That is: $(5*5) - 3 = 22$, which is 0b1_0110
--Set the 1st set of 16 bits of rs3 to 0x7FFF, the highest 16 bit	--That is: $(32,767*32,767) = 1,073,676,289 - 2,147,483,648 = 3,221,159,937$, which overflowed	--That is: $(-32,768*1) = -32768 - 2,147,483,647 = -2,147,516,415$, which underflowed	
--That is: $(32,767*32,767) = 1,073,676,289 - 2,147,483,648 = 3,221,159,937$, which overflowed	--Therefore, saturation should set it at 0x7FFF_FFFF	--Therefore, saturation should set it at 0x8000_0000	
--Therefore, saturation should set it at 0x7FFF_FFFF			

Simulation Result:

Signal name	Value	130 . . . 132 . . . 134 . . . 136 . . . 138 . . . 140 . .
clk	0	
wordIn	10011000000000000000000000000000	10011000000000000000000000000000 139 204 ps
rs3	7FFF00007FFF000080000000000050000	7FFF00007FFF000080000000000050000
rs2	7FFF00007FFF00000001000000050000	7FFF00007FFF00000001000000050000
rs1	80000000800000007FFFFFFFF00000003	80000000800000007FFFFFFFF00000003
rd	7FFFFFFFF7FFFFFFFF80000000000000016	7FFFFFFFF7FFFFFFFF80000000000000016

R4 Instruction: 100 - Long Multiply-Add Low w/ Saturation (Overflow and Underflow)

Function: Long Multiply-Add Low w/ Saturation

WordIn: 1 0 100 00000 00000 00000 00000

Rs1: 0x800000000800000007FFFFFFFF000000003

Rs2: 0x7FFF00007FFF000000001000000050000

Rs3: 0x7FFF00007FFF000080000000000050000

Explanation: Multiply the two low 32 bit values in R2 and R3 together. Then, add the 64 bit value in R1.

--1st --Testing the Underflow portion --Set the 1st set of 32 bits of rs1 to 0x8000_0000_0000_0000, the lowest 64 bit number --Set the 1st set of 16 bits of rs2 to 0x8000_0000, The lowest 32 bit number --Set the 1st set of 16 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --This should saturate at 0x8000_0000_0000_0000	--0th --Overflow Operation --Set the 0th set of 64 bits of rs1 to 0x7FFF_FFFF_FFFF_FFFF, the highest 64 bit number --Set the 0th set of 32 bits of rs2 to 0x7FFF_FFFF --Set the 0th set of 32 bits of rs3 to 0x7FFF_FFFF --This should cap out at 0x7FFF_FFFF_FFFF_FFFF
--	--

Simulation Result:

Signal name	Value	140	142	144	146	148	ns
<input type="checkbox"/> μ clk	0						
<input type="checkbox"/> μ wordIn	10100000000000000000000000000000						148 625 ps
<input type="checkbox"/> μ rs3	000000007FFFFFFFF000000007FFFFFFFF						
<input checked="" type="checkbox"/> μ rs2	00000000800000000000000007FFFFFFFF						
<input type="checkbox"/> μ rs1	800000000000000007FFFFFFFFFFFFFFFF						
<input type="checkbox"/> μ rd	800000000000000007FFFFFFFFFFFFFFFF						

R4 Instruction: 100 - Long Multiply-Add Low w/ Saturation (Normal Operation)

Function: Long Multiply-Add Low w/ Saturation

WordIn: 1 0 100 00000 00000 00000 00000

Rs1: 0x8000000080000007FFFFFFFF00000003

Rs2: 0x7FFF00007FFF00000001000000050000

Rs3: 0x7FFF00007FFF00008000000000050000

Explanation: Multiply the two Low 32 bit values in R2 and R3 together. Then, add the 64 bit value in R1.

<p>--1st --Normal Operation --Set the 1st set of 64 bits of rs1 to 0x0000_0000_000F_4240, the lowest 64 bit number --Set the 1st set of 32 bits of rs2 to 0x7FFF_FFFF, The highest 32 bit number --Set the 1st set of 32 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --That is: $(2147483647 * 2147483647) + 1000000 = 4,611,686,014,133,420,609$ --Expect: 3FFFFFFFF000F4241</p>	<p>--0th --Normal Operation --Set the 1st set of 64 bits of rs1 to 0x0000_0000_000F_4240, the lowest 64 bit number --Set the 1st set of 32 bits of rs2 to 0x7FFF_FFFF, The highest 32 bit number --Set the 1st set of 32 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --That is: $(2147483647 * 2147483647) + 1000000 = 4,611,686,014,133,420,609$ --Expect: 3FFFFFFFF000F4241</p>
--	--

Simulation Result:

Signal name	Value	150	152	154	156	158	160	n
clk	0							
wordIn	10100000000000000000000000000000						158 515 ps	
rs3	000000007FFFFFFFF00000007FFFFFFFF							
rs2	000000007FFFFFFFF00000007FFFFFFFF							
rs1	0000000000F424000000000000F4240							
rd	3FFFFFFFF000F42413FFFFFFFF000F4241							

R4 Instruction: 101 - Long Multiply-Add High w/ Saturation (Normal Operation)

Function: Long Multiply-Add High w/ Saturation

WordIn: 1 0 101 00000 00000 00000 00000

Rs1: 0x000000000000F424000000000000F4240

Rs2: 0x7FFFFFFF000000007FFFFFFF00000000

Rs3: 0x7FFFFFFF000000007FFFFFFF00000000

Explanation: Multiply the two high 32 bit values in R2 and R3 together. Then, add the 64 bit value in R1.

<p>--1st --Normal Operation --Set the 1st set of 64 bits of rs1 to 0x0000_0000_000F_4240, the lowest 64 bit number --Set the 1st set of 32 bits of rs2 to 0x7FFF_FFFF, The highest 32 bit number --Set the 1st set of 32 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --That is: $(2147483647 * 2147483647) + 1000000 = 4,611,686,014,133,420,609$ --Expect: 3FFFFFFF000F4241</p>	<p>--0th --Normal Operation --Set the 1st set of 64 bits of rs1 to 0x0000_0000_000F_4240, the lowest 64 bit number --Set the 1st set of 32 bits of rs2 to 0x7FFF_FFFF, The highest 32 bit number --Set the 1st set of 32 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --That is: $(2147483647 * 2147483647) + 1000000 = 4,611,686,014,133,420,609$ --Expect: 3FFFFFFF000F4241</p>
---	---

Simulation Result:

Signal name	Value	158	160	162	164	166	168	ns
clk	0							
wordIn	1010100000000000000000000000							
rs3	7FFFFFFF000000007FFFFFFF00000000							
rs2	7FFFFFFF000000007FFFFFFF00000000							
rs1	0000000000F424000000000000F4240							
rd	3FFFFFFF000F42413FFFFFFF000F4241							

R4 Instruction: 101 - Long Multiply-Add High w/ Saturation (Overflow and Underflow)

Function: Long Multiply-Add High w/ Saturation

WordIn: 1 0 101 00000 00000 00000 00000

Rs1: 0x80000000000000007FFFFFFFFFFFFFFFFF

Rs2: 0x80000000000000007FFFFFFFF00000000

Rs3: 0x7FFFFFFFF000000007FFFFFFFF00000000

Explanation: Multiply the two high 32 bit values in R2 and R3 together. Then, add the 64 bit value in R1.

--1st
--Testing the Underflow portion
--Set the 1st set of 32 bits of rs1 to
0x8000_0000_0000_0000, the lowest 64 bit number
--Set the 1st set of 16 bits of rs2 to 0x8000_0000,
The lowest 32 bit number
--Set the 1st set of 16 bits of rs3 to 0x7FFF_FFFF,
The highest 32 bit number
--This should saturate at 0x8000_0000_0000_0000

--0th
--Overflow Operation
--Set the 0th set of 64 bits of rs1 to
0x7FFF_FFFF_FFFF_FFFF, the highest 64 bit
number
--Set the 0th set of 32 bits of rs2 to 0x7FFF_FFFF
--Set the 0th set of 32 bits of rs3 to 0x7FFF_FFFF
--This should cap out at 0x7FFF_FFFF_FFFF_FFFF

Simulation Result:

Signal name	Value	170	172	174	176	178	180	ns
clk	0							
wordIn	10101000000000000000000000000000						179 068 ps	
rs3	7FFFFFFFF000000007FFFFFFFF00000000							
rs2	80000000000000007FFFFFFFF00000000							
rs1	80000000000000007FFFFFFFFFFFFFFFFF							
rd	80000000000000007FFFFFFFFFFFFFFFFF							

R4 Instruction: 110 - Long Multiply-Subtraction Low w/ Saturation (Normal Operation)

Function: Long Multiply-Subtraction Low w/ Saturation

WordIn: 10110000000000000000000000000000

Rs1: 0x80000000000000007FFFFFFFFFFFFFFFFF

Rs2: 0x0000000080000000000000007FFFFFFFFF

Rs3: 0x000000007FFFFFFFFF000000007FFFFFFFFF

Explanation: Multiply the two low 32 bit values in R2 and R3 together. Then, subtract the 64 bit value in R1.

--1st --Testing the Normal operation --Set the 1st set of 32 bits of rs1 to 0x8000_0000_0000_0000, the lowest 64 bit number --Set the 1st set of 16 bits of rs2 to 0x8000_0000, The lowest 32 bit number --Set the 1st set of 16 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --This should result in to 0xBFFF FFFF 0000 0002	--0th --Normal Operation --Set the 0th set of 64 bits of rs1 to 0x7FFF_FFFF_FFFF_FFFF, the highest 64 bit number --Set the 0th set of 32 bits of rs2 to 0x7FFF_FFFF --Set the 0th set of 32 bits of rs3 to 0x7FFF_FFFF --This should result in 0x4000 0000 8000 0000
--	---

Simulation Result:

Signal name	Value	180	182	184	186	188	190
clk	0						188 969 ps
wordIn	10110000000000000000000000000000	X		10110000000000000000000000000000			
rs3	000000007FFFFFFFFF000000007FFFFFFFFF	X		000000007FFFFFFFFF000000007FFFFFFFFF			
rs2	0000000080000000000000007FFFFFFFFF	X		0000000080000000000000007FFFFFFFFF			
rs1	80000000000000007FFFFFFFFFFFFFFFFF			80000000000000007FFFFFFFFFFFFFFFFF			X
rd	4000000080000000BFFFFFFFFF00000002	X		4000000080000000BFFFFFFFFF00000002			X

R4 Instruction: 110 - Long Multiply-Subtraction Low w/ Saturation (Overflow and Underflow)

Function: Long Multiply-Subtraction Low w/ Saturation

WordIn: 1011000000000000000000000000

Rs1: 0x7FFFFFFFFFFFFFFFFF8000000000000000

Rs2: 0x00000000800000000000000007FFFFFFFF

Rs3: 0x000000007FFFFFFFFF000000007FFFFFFFF

Explanation: Multiply the two low 32 bit values in R2 and R3 together. Then, subtract the 64 bit value in R1.

--1st --Testing the Underflow Operation --Set the 1st set of 32 bits of rs1 to 0x7FFF_FFFF_FFFF_FFFF, the highest 64 bit number --Set the 1st set of 16 bits of rs2 to 0x8000_0000, The lowest 32 bit number --Set the 1st set of 16 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --This should saturate to 0x8000 0000 0000 000	--0th --Overflow Operation --Set the 0th set of 64 bits of rs1 to 0x8000_0000_0000_0000, the lowest 64 bit number --Set the 0th set of 32 bits of rs2 to 0x7FFF_FFFF --Set the 0th set of 32 bits of rs3 to 0x7FFF_FFFF --This should result in 0x7FFF_FFFF_FFFF_FFFF
--	--

Simulation Result:

Signal name	Value	190	192	194	196	198	200
clk	0						
wordIn	1011000000000000000000000000	1011000000000000000000000000					199 146 ps
rs3	000000007FFFFFFFFF000000007FFFFFFFF	000000007FFFFFFFFF000000007FFFFFFFF					X
rs2	0000000080000000000000007FFFFFFFF	0000000080000000000000007FFFFFFFF					X
rs1	7FFFFFFFFFFFFFFFFF8000000000000000	X	7FFFFFFFFFFFFFFFFF8000000000000000				X
rd	80000000000000007FFFFFFFFFFFFFFFFF	X	80000000000000007FFFFFFFFFFFFFFFFF				X

R4 Instruction: 111 - Long Multiply-Subtraction High w/ Saturation (Normal Operation)
Function: Long Multiply-Subtraction High w/ Saturation
WordIn: 10111000000000000000000000000000
Rs1: 0x000000000000F424000000000000F4240
Rs2: 0x7FFFFFFF000000007FFFFFFF00000000
Rs3: 0x7FFFFFFF000000007FFFFFFF00000000
Explanation: Multiply the two high 32 bit values in R2 and R3 together. Then, subtract the 64 bit value in R1.

<div>--1st --Normal Operation --Set the 1st set of 64 bits of rs1 to 0x0000_0000_000F_4240, the lowest 64 bit number --Set the 1st set of 32 bits of rs2 to 0x7FFF_FFFF, The highest 32 bit number --Set the 1st set of 32 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --That is: (2147483647 * 2147483647) - 1000000 = 4,611,686,014,133,420,609 --Expect: 3FFFFFFF000F4241</div>	<div>--0th --Normal Operation --Set the 1st set of 64 bits of rs1 to 0x0000_0000_000F_4240, the lowest 64 bit number --Set the 1st set of 32 bits of rs2 to 0x7FFF_FFFF, The highest 32 bit number --Set the 1st set of 32 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number --That is: (2147483647 * 2147483647) + 1000000 = 4,611,686,014,131,420,609 --Expect: 3FFFFFFFEFFF0BDC1</div>
--	---

Simulation Result:

Signal name	Value	200	202	204	206	208	210
clk	0						
wordIn	10111000000000000000000000000000	X					X
rs3	7FFFFFFF000000007FFFFFFF00000000	X					X
rs2	7FFFFFFF000000007FFFFFFF00000000	X					X
rs1	0000000000F424000000000000F4240	X					X
rd	3FFFFFFFEFFF0BDC13FFFFFFFEFFF0BDC1	X					X

R4 Instruction: 111 - Long Multiply-Subtraction High w/ Saturation (Overflow and Underflow)

Function: Long Multiply-Subtraction High w/ Saturation

WordIn: 10111000000000000000000000

Rs1: 0x7FFFFFFFFFFFFFFFFF8000000000000000

Rs2: 0x80000000000000007FFFFFFFFF00000000

Rs3: 0x7FFFFFFFFF000000007FFFFFFFFF00000000

Explanation: Multiply the two high 32 bit values in R2 and R3 together. Then, subtract the 64 bit value in R1.

<p>--1st</p> <p>--Testing the Underflow Operation</p> <p>--Set the 1st set of 32 bits of rs1 to 0x7FFF_FFFF_FFFF_FFFF, the highest 64 bit number</p> <p>--Set the 1st set of 16 bits of rs2 to 0x8000_0000, The lowest 32 bit number</p> <p>--Set the 1st set of 16 bits of rs3 to 0x7FFF_FFFF, The highest 32 bit number</p> <p>--This should saturate to 0x8000 0000 0000 0000</p>	<p>--0th</p> <p>--Overflow Operation</p> <p>--Set the 0th set of 64 bits of rs1 to 0x8000_0000_0000_0000, the lowest 64 bit number</p> <p>--Set the 0th set of 32 bits of rs2 to 0x7FFF_FFFF</p> <p>--Set the 0th set of 32 bits of rs3 to 0x7FFF_FFFF</p> <p>--This should result in 0x7FFF_FFFF_FFFF_FFFF</p>
--	---

Simulation Result:

Signal name	Value	210	212	214	216	218	220
clk	0						
wordIn	10111000000000000000000000						218.877 ps
rs3	7FFFFFFFFF000000007FFFFFFFFF00000000						
rs2	80000000000000007FFFFFFFFF00000000						
rs1	7FFFFFFFFFFFFFFFFF8000000000000000						
rd	80000000000000007FFFFFFFFFFFFFFFFF						

R3 Instructions

R3 Instruction: 0000 - NOP

Function: NOP

Word In: 1 1 0000 0000 00000 00000 00000

Explanation: The ALU in this case will output only all 0s. This is to verify that the NOP is actually being selected. However, this function will really be implemented through the register file, and not the ALU.

Simulation Result:

Signal name	Value	150	152	154	156	158	160
JLr clk	0						
JLr wordIn	1800000	X			1800000		X
JLr rs3	7FFFFFFF000000007FFFFFFF00000000				7FFFFFFF000000007FFFFFFF00000000		
JLr rs2	80000000000000007FFFFFFF00000000				80000000000000007FFFFFFF00000000		
JLr rs1	7FFFFFFF000000008000000000000000				7FFFFFFF000000008000000000000000		X
JLr rd	00000000000000000000000000000000	X			00000000000000000000000000000000		X

R3 Instruction: 0001 - SHRHI (Test 1)

Function: Shift right halfword immediate. The immediate value is provided in the rs2 field (13 downto 10)

Word In: 1 1 0000 0001 00000 00000 00000

Rs1: 0x1234 0000 2345 0000 0000 0000 2345 1234

Rs2 Field: 0b0001

Explanation: Rotate each variable by 1.

Expect: 0x091A 0000 11A2 0000 0000 0000 11A2 091A

R3 Instruction: 0001 - SHRHI (Test 2)

Function: Shift right halfword immediate. The immediate value is provided in the rs2 field (13 downto 10)

Word In: 1 1 0000 0001 00000 00000 00000

Rs1: 0x1234 0000 2345 0000 0000 0000 2345 1234

Rs2 Field: 0b0100

Explanation: Shift each index right by 4

Expect: 0x0123 0000 0234 0000 0000 0000 0234 0123

R3 Instruction: 0001 - SHRHI (Test 3)

Function: Shift right halfword immediate. The immediate value is provided in the rs2 field (13 downto 10)

Word In: 1 1 0000 0001 00000 00000 00000

Rs1: 0xFFFE 0000 2345 0000 0000 0000 1FFF FFFF

Rs2 Field: 0b1000

Explanation: Shift each index right by 8

Expect: 0x00FF 0000 0023 0000 0000 0000 001F 00FF

Simulation Result:

Signal name	Value	152	156	160	164	168	172	176	180
JLr clk	0								
JLr wordIn	180A000	X	1808400	X	1809000	X	180A000	X	17
JLr rs2	80000000000000007FFFFFFF00000000				80000000000000007FFFFFFF00000000				X
JLr rs1	FFFE000023450000000000001FFFFFFF	X	12340000234500000000000023451234	X	12340000234500002350009023451234	X	FFFE000023450000000000001FFFFFFF	X	
JLr rd	00FF0000002300000000000001F00FF	X	091A000011A200000000000011A2091A	X	01230000023400000235000902340123	X	00FF0000002300000000000001F00FF	X	

R3 Instruction: 0010 - AU

Function: Add Unsigned

Word In: 1 1 0000 0010 00000 00000 00000

Rs1: 0x65432101000000000000000000F4240

Rs2: 0x65432101000000007FFFFFFF000C0000

Explanation: Add every 32 bit values: expected value is 0xCA864202 00000000 7FFFFFFF 001B4240

Simulation Result:

Signal name	Value	150	152	154	156	158
clk	0					
wordIn	110000001000000000000000	X		110000001000000000000000		158 079 ps
rs2	65432101000000007FFFFFFF000C0000	X		65432101000000007FFFFFFF000C0000		
rs1	654321010000000000000000F4240	X		654321010000000000000000F4240		
rd	CA864202000000007FFFFFFF001B4240	X		CA864202000000007FFFFFFF001B4240		

R3 Instruction: 0011 - CNT1H

Function: Count the number of 1s in each halfword

Word In: 1 1 0000 0011 00000 00000 00000

Rs1: 0xF000000000F404000000000000F4240

Explanation: Add up the 1s in each halfword. Expected value is 0004 0000 0004 0002 0000 0000 0004 0003

Simulation Result:

Signal name	Value	162	164	166	168	170
clk	0					
wordIn	110000001100000000000000	X		110000001100000000000000		167 828 ps
rs2	F000000000000000000000000000	X		F000000000000000000000000000		
rs1	F000000000F4040000000000F4240	X		F000000000F4040000000000F4240		
rd	000400000040002000000000040003	X		000400000040002000000000040003		

R3 Instruction: 0100 - AHS

Function: Add halfword saturated

Word In: 1 1 0000 0100 00000 00000 00000

Explanation: Add each corresponding halfword together. Employ saturation.

--3rd	--2nd	--1st	--0th
--Normal Operation	--Overflow Condition	--Underflow Operation	--Normal Operation
--Set 3rd set of rs2 to 0x0005, which is dec 5	--Set 2nd set of rs2 to 0x7FFF, which is most positive number	--Set last set of rs2 to 0x8000, which is most negative number	--Set zeroth set of rs2 to 0x1AB5, which is 6,837
--Set 3rd set of rs1 to 0x0007, which is dec 7	--Set 2nd set of rs1 to 0x0005, which is 5	--Set last set of rs1 to 0x8000, which is most negative number	--Set zeroth set of rs1 to 0x9263, which is 37475
--Answer should be placed in last 16 bits of rd	--Answer should be placed in last 16 bits of rd	--Answer should underflow - 16 bits can't get more negative than 8000	--Answer should be placed in zeroth 16 bits of rd
--Answer is 12, which is 000C	--Answer should overflow - you can't get anymore positive	--Answer should 0x8000	--Answer is 44,312, answer is 0xAD18
	--Saturation will kick in, answer is 0x7FFF		

Simulation Result:

Signal name	Value	200	202	204	206	208
clk	0					
wordIn	1820000	X		1820000		208 094 ps
rs2	00057FFF80001AB500057FFF80001AB5	X		00057FFF80001AB500057FFF80001AB5		
rs1	00070005800092630007000580009263	X		00070005800092630007000580009263		
rd	000C7FFF8000AD18000C7FFF8000AD18	X		000C7FFF8000AD18000C7FFF8000AD18		

R3 Instruction: 0101 - OR

Function: Bitwise Logical Or

Word In: 1 1 0000 0101 00000 00000 00000

Explanation: We expect to see only the 1s from R2 and R1 to propagate down to Rd

Simulation Result:

Signal name	Value	190	192	194	196	198
clk	0					
wordIn	1828000	X		1828000		198 974 ps
rs2	0000000007FFFFFFFFC0000000000000	X		0000000007FFFFFFFFC0000000000000		
rs1	0000001FFFFFFFF000000000000000000	X		0000001FFFFFFFF000000000000000000		
rd	0000001FFFFFFFF000000000000000000	X		0000001FFFFFFFF000000000000000000		

R3 Instruction: 0110 - BCW

Function: Broadcast the 0th 32 bits of Rs1 to each of the other 3 indices.

Word In: 1 1 0000 0110 00000 00000 00000

Rs1: 0xF000000000F404000000000123F4240

Explanation: Copy 0x123F 4240 to everything else. Expect 0x123F 4240 123F 4240 123F 4240 123F 4240

Simulation Result:

Signal name	Value	200	202	204	206	208
clk	0					
wordIn	1830000	X		1830000		208 433 ps
rs2	10000000000000000000000000000000	X		10000000000000000000000000000000		
rs1	F000000000F404000000000123F4240	X		F000000000F404000000000123F4240		
rd	123F4240123F4240123F4240123F4240	X		123F4240123F4240123F4240123F4240		

R3 Instruction: 0111 - MAXWS

Function: For each 32 bit word slot, select the maximum signed value to be placed in Rd.

Word In: 1 1 0000 0111 00000 00000 00000

Rs1: 0xF000000F000F4040000000000020000

Rs2: 0x123F4240000000000123F4240123F4240

Explanation: Expect to see the highest for each slot. Expect: 123F 4240 000F 4040 123F 4240 123F 4240

Simulation Result:

Signal name	Value	212	214	216	218	220
clk	0					
wordIn	11000001110000000000000000	X		11000001110000000000000000		219 119 ps
rs2	123F424000000000123F4240123F4240	X		123F424000000000123F4240123F4240		
rs1	F000000F000F4040000000000020000	X		F000000F000F4040000000000020000		
rd	123F4240000F4040123F4240123F4240	X		123F4240000F4040123F4240123F4240		

R3 Instruction: 1000 - MINWS

Function: For each 32 bit word slot, select the most minimum signed value to be placed in Rd

Word In: 1 1 0000 1000 00000 00000 00000

Rs1: 0xF000000000000000F0000000F0000000

Rs2: 0x10000000F00000001000000000000000

Explanation: F would represent a negative number, while 1 represents a positive. Thus, we should only see the F functions in the register Rd.

Simulation Result:

Signal name	Value	222	224	226	228	230
JLr clk	0					
JLr wordIn	1840000		1840000			229 390 ps
JLr rs3	7FFFFFFF000000007FFFFFFF00000000		7FFFFFFF000000007FFFFFFF00000000			
JLr rs2	10000000F00000001000000000000000		10000000F00000001000000000000000			
JLr rs1	F000000000000000F0000000F0000000		F000000000000000F0000000F0000000			
JLr rd	F0000000F0000000F0000000F0000000		F0000000F0000000F0000000F0000000			

R3 Instruction: 1001 - MLHU

Function: Multiply Low Unsigned

Word In: 1 1 0000 1001 00000 00000 00000

Rs1: 0xFF00FFFF000F40400000010400000003

Rs2: 0x110000010000FFFF0000000200000003

Explanation:

--3rd	--2nd	--1st	--0th
--FFFF * 0001 = FFFF [Hex]	--4040 * FFFF = 1043F3BBFC0 [Hex]	--104 * 2 = 208 [Hex] --260 * 2 = 520 [Dec]	--3*3 = 9
--65535 * 1 = 65535 [Dec]	--16448 * 65535 = 1,077,919,680 [Dec]		

Simulation Result:

Signal name	Value	230	232	234	236	238	240
JLr clk	0						
JLr wordIn	1848000		1848000			237 918 ps	
JLr rs3	7FFFFFFF000000007FFFFFFF00000000		7FFFFFFF000000007FFFFFFF00000000				
JLr rs2	110000010000FFFF0000000200000003		110000010000FFFF0000000200000003				
JLr rs1	FF00FFFF000F40400000010400000003		FF00FFFF000F40400000010400000003				
JLr rd	0000FFFF403FBFC00000020800000009		0000FFFF403FBFC00000020800000009				

R3 Instruction: 1010 - MLHSS

Function: Multiply by Sign Saturated

Word In: 1 1 0000 1010 00000 00000 00000

Explanation:

--7th	--6th	--5th	--4th	--3rd	--2nd	--1st	--0th
--FFFF * 0001 = FFFF[Hex]	-- C56A * 8000 = 3A96 [Hex]	-- 5136 * 5691 = 5136 [Hex]	-- C56A * 0001 = C56A [Hex]	--7FFF * 0001 = 7FFF [Hex]	--saturate	-- -3 * -1 = 3	-- 3*-1 = -3

$1 = -1$ [Dec]	$-14998 * -1$ = 14998 [Dec]	$5430 * 1 =$ 5430 [Dec]	$-14998 * 1$ = -14998 [Dec]	$32767 * 1$ = 32767 [Dec]	$8000 * -1$ = 7FFF [Hex]	$FFFD * -1$ = 3 [Hex]	$3 * 8000 =$ FFFD [hex]
					$-32768 * -1$ = 32767 [Dec]		

Simulation Result:

Signal name	Value	252	254	256	258	260 ns
clk	0					
wordIn	1850000					259 610 ps
rs3	7FFFFFFF00000007FFFFFFF00000000					
rs2	00018000569100010001800080008000					
rs1	FFFFC56A5136C56A7FFF8000FFFD0003					
rd	FFFF3A965136C56A7FFF7FFF0003FFFD					

R3 Instruction: 1011 - AND

Function: Bitwise And Rs1 and Rs2

Word In: 1 1 0000 1011 00000 00000 00000

Explanation: We load some 1s into both Rs1 and Rs2. The bits which overlap should be shown in Rd.

Simulation Result:

Signal name	Value	252	254	256	258	260 ns
clk	0					
wordIn	11000010110000000000000000					259 546 ps
rs2	0000001FFFFFF000000000000000000					
rs1	FFFFFFFF800000000000000000000000					
rd	0000001F800000000000000000000000					

R3 Instruction: 1100 - INVB

Function: Bitwise invert Rs1

Word In: 1 1 0000 1100 00000 00000 00000

Explanation: We load in 1s in some part of Rs1. We expect to see the entire register inverted

Simulation Result:

Signal name	Value	260	262	264	266	268 ns
clk	0					
wordIn	1860000					269 365 ps
rs2	0000001FFFFFF000000000000000000					
rs1	FFFFFFFFFFFFFFFF0000000000000000					
rd	0000000000000000FFFFFFFFFFFFFFFF					

R3 Instruction: 1101 ROTW

Function: Rotate each 32 bit word in Rs1 by the corresponding value in Rs2.

Word In: 1 1 0000 1101 00000 00000 00000

Rs1: 0x0123 4567 89AB CDEF 0123 4567 89AB CDEF

Rs2: 0x0000 0004 0000 0003 0000 0004 0000 001F

Explanation: We're rotating the words by 4, 3, 4, and 31, respectively.

Expecting: 0x 7012 3456 F135 79BD 7012 3456 1357 9BDF

Simulation Result:

Signal name	Value	300	302	304	306	308
clk	0					
wordIn	1868000			1868000		309 583 ps
rs2	0000000400000003000000040000001F			0000000400000003000000040000001F		
rs1	0123456789ABCDEF0123456789ABCDEF			0123456789ABCDEF0123456789ABCDEF		
rd	70123456F13579BD7012345613579BDF			70123456F13579BD7012345613579BDF		

R3 Instruction: 1110 SFWU

Function: Subtract from Word Unsigned

Word In: 1 1 0000 1110 00000 00000 00000

Explanation:

--1st	--3rd	--1st	--0th
--Testing edge case for SFWU	--Set last set of rs2 to 0xFFFF_FFFF, which is dec 4,294,967,295	--Testing edge case for SFWU	--Set 0th set of rs1 to 0x0100_0000, which is dec 16,777,216
--Set last set of rs2 to 0x1000_0000, which is dec 16,777,216	--Set last set of rs1 to 0x0000_1CF3, which is dec 7,411	--Set last set of rs2 to 0x0000_0005, which is dec 5	--Set 0th set of rs2 to 0x1000_0000, which is dec 268,435,456
--Set last set of rs1 to 0x0100_0000, which is dec 268,435,456	--Answer is negative when done in decimal.	--Set last set of rs1 to 0x0000_0007, which is dec 7	--Answer should be placed in last 32 bits of rd
--Answer should be placed in last 32 bits of rd.	--Answer is should underflow to FFFF_E30C	--Answer is negative when done in decimal.	--Answer is 0x0F00_0000, or 251,658,240
--Answer is 0x0F00_0000, or 251,658,240		--Answer is should underflow to FFFF_FFFE	

Simulation Result:

Signal name	Value	310	312	314	316	318
clk	0					
wordIn	110000111000000000000000			110000111000000000000000		318 927 ps
rs2	10000000FFFFFFF0000000500000005			10000000FFFFFFF0000000500000005		
rs1	0100000000001CF30000000700000002			0100000000001CF30000000700000002		
rd	0F000000FFFFE30CFFFFFFFE00000003			0F000000FFFFE30CFFFFFFFE00000003		

R3 Instruction: 1111 - SFHS

Function: Subtract from Halfword Saturated

Word In: 1 1 0000 1111 00000 00000 00000

Rs1: 0x000780008000000010007800080000001

Rs2: 0x00057FFF8000800000057FFF80008000

Explanation: These 4 test cases are repeated in indices 7, 6, 5, 4

--3rd	--2nd	--1st	--0th
--Normal Operation	--Overflow Condition	--Regular Operation	--Underflow Condition
--Set 3rd set of rs2 to 0x0005, which is dec 5	--Set 2nd set of rs2 to 0x7FFF, which is most positive number	--Set last set of rs2 to 0x8000, which is most negative number	--Set zeroth set of rs2 to 0x8000, which is most negative number
--Set 3rd set of rs1 to 0x0007, which is dec 7	--Set 2nd set of rs1 to 0x8000, which is most negative number	--Set last set of rs1 to 0x8000, which is most negative number	--Set zeroth set of rs1 to 0x0001, which is 1
--Answer should be placed in last 16 bits of rd	--Answer should be placed in last 16 bits of rd	--Answer should be placed in 1st 16 bits of rd	--Answer should be placed in zeroth 16 bits of rd
--Answer is -2, which is FFFE	--Answer should overflow - you can't get anymore positive	--Answer should 0x0000	--Answer should underflow - you can't get any more negative
	--Saturation will kick in, answer is 0x7FFF		--Saturation will kick in, answer is 0x8000

Simulation Result:

Signal name	Value	320	322	324	326	328	ns
clk	0						
wordin	1878000						329 517
rs2	00057FFF8000800000057FFF80008000						
rs1	00078000800000010007800080000001						
rd	FFFE7FFF00008000FFFE7FFF00008000						