PROJECT 2:
ARITHMETIC LOGIC UNIT (ALU)

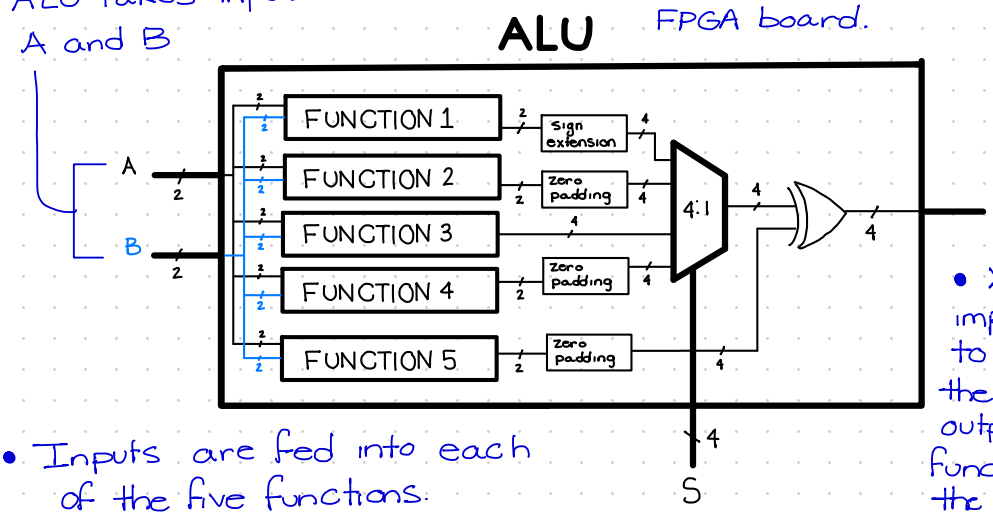
Arithmetic Logic Unit (ALU) Overview

- digital system that accepts binary numerical input values
- computes one out of several possible output values based on an arithmetic or logical function of the inputs.
- Also accepts another binary value, an opcode, which dictates which function is computed.

System Block Diagram

ALU takes inputs A and B

- 4:1 MUX implemented, selects which functions is output to FPGA board.



- Inputs are fed into each of the five functions.
- Sign extension and zero padding blocks ensure that we have 4-bits displayed on the output, Y.
- XOR gate implemented to ensure the board outputs a function from the MUX, or function 5, but not both.

TRUTH TABLE

	S	Y
ALL BTN OFF ←	0 0 0 0	A > B ← Function 5
BTN1 ON	0 0 0 1	A + B (signed) ← Function 1
BTN2 ON	0 0 1 0	A ASR B ← Function 2
BTN3 ON	0 1 0 0	AB ← Function 3
BTN4 ON	1 0 0 0	A XOR B ← Function 4

- Each value of S on the MUX corresponds to 1 of 4 push buttons on the FPGA board.
- When no buttons are pressed, Function 5 is output to the board.

Function 1	y = A + B, A and B represent two input operands and are signed binary numbers.
Function 2	Arithmetic shift right of A by B positions
Function 3	y = AB, the 4-bit product of A and B, both inputs are unsigned binary numbers
Function 4	y = A XOR B
Function 5	y = A > B, LSb of Y = '1' if A > B and '0' otherwise.

Project Summary | Function1.vhd *

E:/FPGA/ALU/ALU.srcs/sources_1/new/Function1.vhd

```
26  -- arithmetic functions with Signed or Unsigned values
27  use IEEE.NUMERIC_STD.ALL; --uncomment this line allows me to use signed and unsigned types
28
29  -- Uncomment the following library declaration if instantiating
30  -- any Xilinx leaf cells in this code.
31  --library UNISIM;
32  --use UNISIM.VComponents.all;
33
34  entity Function1 is
35      Port ( A : in STD_LOGIC_VECTOR (1 downto 0); -- 1 downto 0 = 2-bit std_logic_vector input A
36             B : in STD_LOGIC_VECTOR (1 downto 0); -- 1 downto 0 = 2-bit std_logic_vector input B
37             Y : out STD_LOGIC_VECTOR (3 downto 0)); -- 3 downto 0 = 2-bit std_logic_vector output Y
38  end Function1;
39
40  architecture Behavioral of Function1 is
41
42  --Function 1
43  signal A_signed : signed(1 downto 0);
44  signal B_signed : signed(1 downto 0);
45  signal sum : signed(1 downto 0);
46  signal sum1 : signed(3 downto 0);
47  signal f1 : std_logic_vector(3 downto 0);
48
49  --Function 2
50  signal f2 : std_logic_vector(3 downto 0);
51
52  --Function 3
53  signal A_unsigned : unsigned(1 downto 0);
54  signal B_unsigned : unsigned(1 downto 0);
55  signal Y_unsigned : unsigned(3 downto 0);
56  signal f3 : std_logic_vector(3 downto 0);
57
58  --Function 4
59  signal f4 : std_logic_vector(3 downto 0);
60
61  --Function 5
62  signal f5 : std_logic_vector(3 downto 0);
63
64
65  begin
66
67
68  end Behavioral;
69
```

Needed to add S input for opcode, NEXT PAGE

Type casts are utilized to do this

In order to use arithmetic operators like "+" and "*" the signal must be converted to signed or unsigned

only functions 1 and 3 require type casting since those are the only functions with arithmetic operations.

```
1  ----------------------------------------------------------------------------...
20
21
22  library IEEE;
23  use IEEE.STD_LOGIC_1164.ALL;
24
25  -- Uncomment the following library declaration if using...
27  use IEEE.NUMERIC_STD.ALL; --uncomment this line allows me to use signed and unsigned types
28
29  -- Uncomment the following library declaration if instantiating...
33
34  entity Function1 is
35      Port ( A : in STD_LOGIC_VECTOR (1 downto 0); -- 1 downto 0 = 2-bit std_logic_vector input A
36             B : in STD_LOGIC_VECTOR (1 downto 0); -- 1 downto 0 = 2-bit std_logic_vector input B
37             Y : out STD_LOGIC_VECTOR (3 downto 0); -- 3 downto 0 = 2-bit std_logic_vector output Y
38             S : in STD_LOGIC_VECTOR (3 downto 0)
39             ); -- S declared here. Refer to system block diagram.
40  end Function1;
```

in tbench

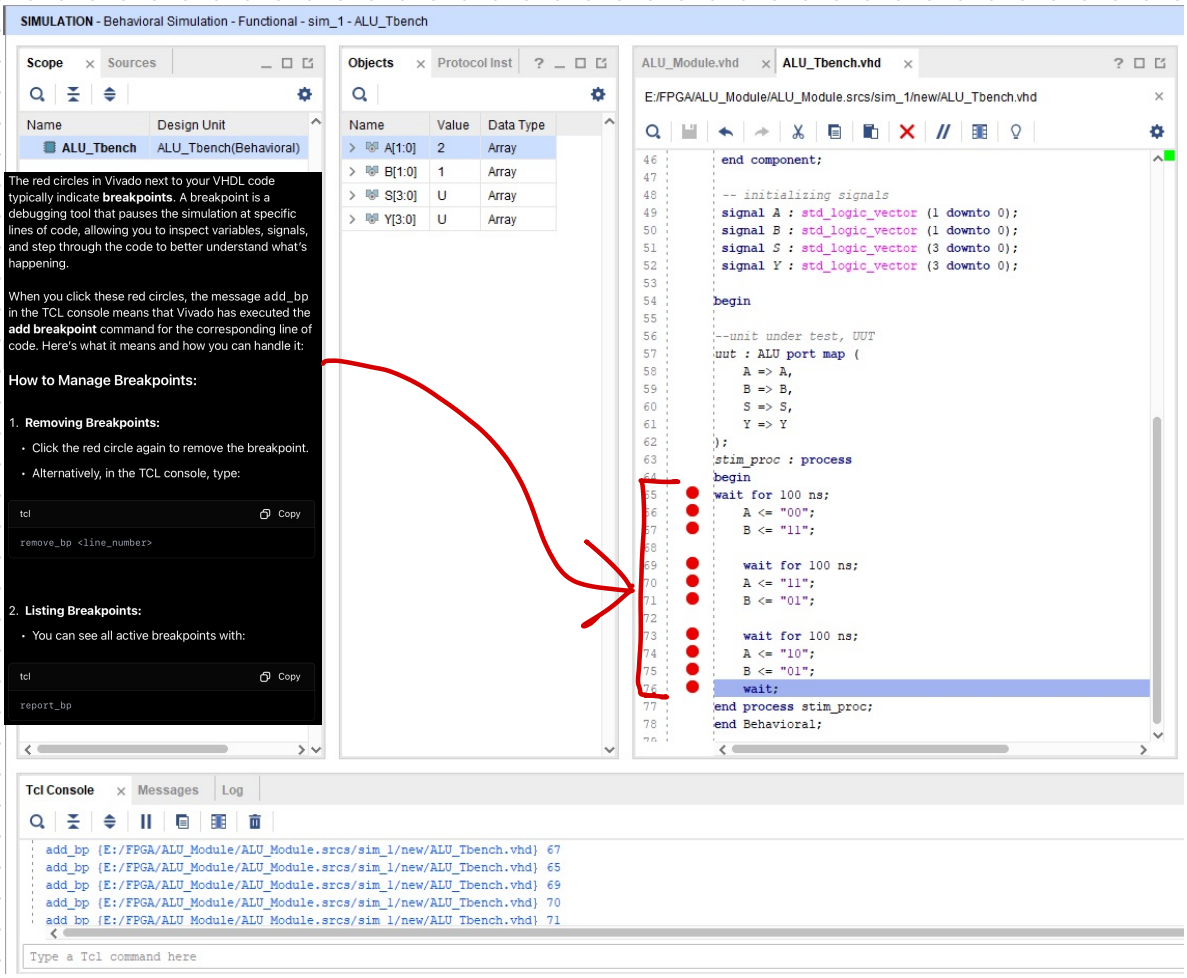
Entity name must match component declaration

Added S input for opcode

LOGIC FOR 5 Functions

```
67  begin
68
69  --Function 1
70  A_signed <= signed(A);
71  B_signed <= signed(B);
72  sum <= (A_signed + B_signed);
73  sum1 <= sum(1) & sum(1)& sum;
74  f1 <= std_logic_vector(sum1);
75
76  --Function2
77  f2 <= ("00" &  A(1 downto 0)) when B = "00" else ("00" & ( A(1)& A(1))) when B = "01" else "0000"; -- Zero padding to ensure 4-bits
78
79  --Function 3
80  A_unsigned <= unsigned(A);
81  B_unsigned <= unsigned(B);
82  Y_unsigned <= A_unsigned * B_unsigned;
83  f3 <= std_logic_vector(Y_unsigned);
84
85  --Function 4
86  f4 <= "00" & (A XOR B); -- Zero padding to ensure 4-bits are output to Y
87
88  --Function 5
89  f5 <= "0001" when A > B else "0000"; -- final implementation
90
91   --                   initial implementation
92  --f5 <= "0001" when (A = "01" and B = "00") or (A = "10" and B = "00")
93  -- or (A = "10" and B = "01") or (A = "11" and B = "00") or (A = "11" and B = "01")
94  -- or (A = "11" and B = "10") else "0000";
95
96  Y <= f5 when S <= "0000" else f1 when S <= "0001" else f2 when S <= "0010" else f3 when S <= "0100" else f4 when S <= "1000";
97  -- S corresponds to 1 of 4 push buttons on the FPGA board. When no buttons are pressed (i.e S = 0000) function 5 is output to Y.
98
99
100 end Behavioral;
101
```

This line makes the S value control which function is output to the board



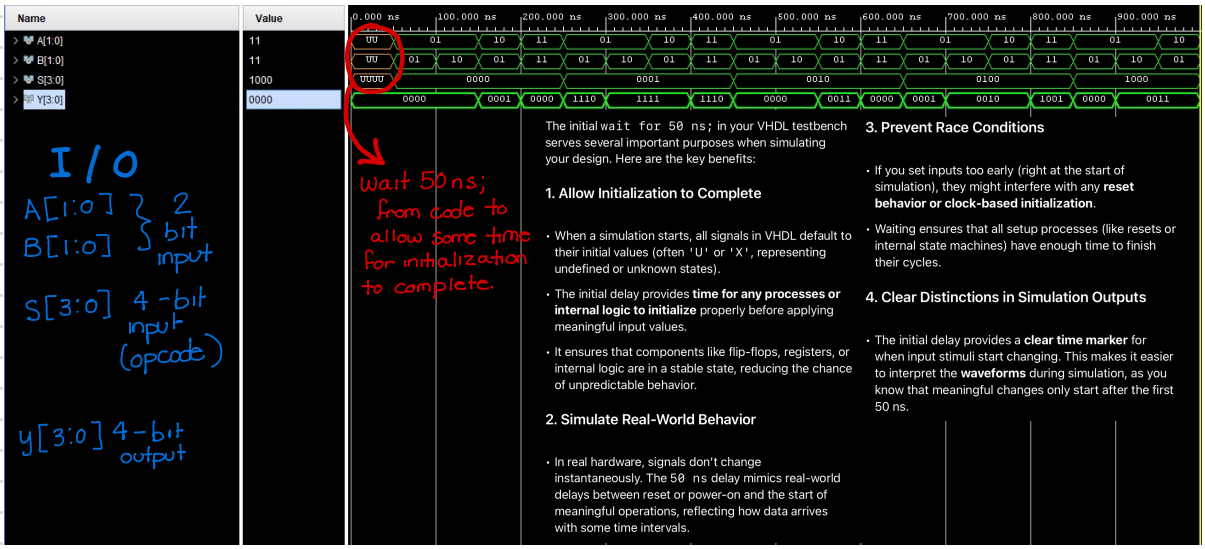
NOTE: Came across issues with simulation. Kept simulating the wrong source file.

TROUBLESHOOTING METHODS

- Component declaration name in testbench was different from entity name in design file. These **must** match. Renamed both to "ALU1"

- Also deleted old source files from project (vivado design hierarchy) AND same folder as current project

Simulation



Function 5 Verification

S[3:0] = 0000 → FUNCTION 5

A	B	y
01	01	0000
01	10	0000
10	01	0001
11	11	0000

y = "0001" when A > B

LED ON IN THIS CASE

Function 1 Verification

S[3:0] = 0001 → FUNCTION 1

A	B	y
01	01	1110
01	10	1111
10	01	1111
11	11	1110

y = A + B (signed)

possible error in function 1 implementation

$$\begin{array}{r} 2 \\ +\ 2 \\ \hline 4 \end{array} \qquad \begin{array}{r} {\scriptstyle 1} \\ 0\ 1 \\ +\ 0\ 1 \\ \hline 1\ 0 \end{array}$$

EXPECTED OUTPUT

y = 0010

zero padding to make y output 4-bits

SIMULATION OUTPUT

X y = 1110

reanalyzing my hardware design for function 1

```
68
69    --Function 1
70    A_signed <= signed(A);
71    B_signed <= signed(B);
72    sum <= (A_signed + B_signed);
73    suml <= sum(1) & sum(1)& sum;
74    f1 <= std_logic_vector(suml);
75
```

- Zero padding should be used here is function 1.
- I speculate that sum(1) & sum(1) are responsible for y = 1110
- and sum is responsible for sum being the actual sum of inputs A and B

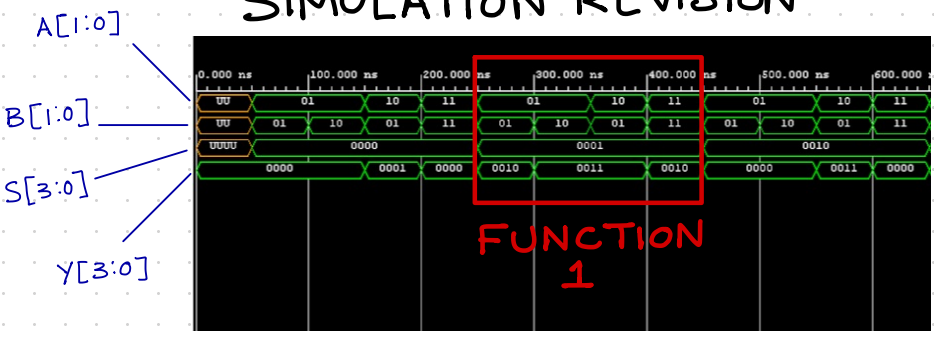
sum(1) is the Most significant bit (MSB) of sum variable

FUNCTION 1 REVISION

```
68
69    --Function 1
70    A_signed <= signed(A); -- convert signed to unsigned, input A
71    B_signed <= signed(B); -- convert signed to unsigned, input B
72    sum <= (A_signed + B_signed); -- signed addition
73
74    -- using the concatenation operator (&), sum(1) is the MSB of sum variable
75     --suml <= sum(1) & sum(1)& sum; -- resulted in errors in function 1 simulation
76    suml <= "0" & "0" & sum;
77
78    -- revised function 1 logic
79    --suml <= ("0"& sum); -- Trunacte to 4-bits
80    f1 <= std_logic_vector(suml); -- Convert back to std_logic_vector
81
```

Added Zero padding instead of Sum(1)

SIMULATION REVISION



Function 1 Reverification

S[3:0] = 0001 → FUNCTION 1

A	B	y	
01	01	0010	✓
01	10	0011	✓
10	01	0011	✓
11	11	0010	X

y = A+B (signed)

```
  1
  1 1
+ 1 1
-----
1 1 0
```

EXPECTED OUTPUT

y = 0110

↓

zero padding to make y output 4-bits

SIMULATION OUTPUT

y = 0010 X

CONCLUSION:

When the sum of A and B is 3 bits the zero padding doesn't output the correct value to y.

TROUBLESHOOTING STEPS:

Since only one combination of A and B results in a 3 bit sum, I'll add conditional logic for that scenario into function 1.

FUNCTION 1 REVISION 2

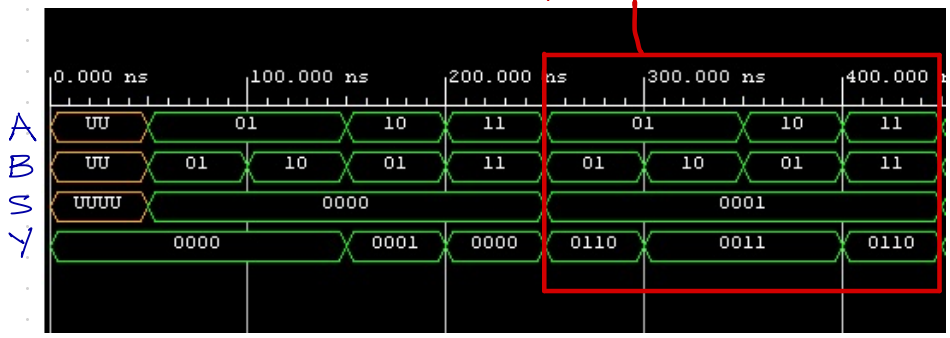
```
--Function 1
A_signed <= signed(A); -- convert signed to unsigned, input A
B_signed <= signed(B); -- convert signed to unsigned, input B
sum <= (A_signed + B_signed); -- signed addition

-- using the concatenation operator (&), sum(1) is the MSB of sum variable
--sum1 <= sum(1) & sum(1)& sum; -- resulted in errors in function 1 simulation
-- sum1 <= "0" & "0" & sum;
process(A, B, sum)
begin
    if sum = "110" then
        sum1 <= "0110";
        else sum1 <= "0" & "0" & sum;
    end if;
end process;

-- revised function 1 logic
--sum1 <= ("0"& sum); -- Trunacte to 4-bits
f1 <= std_logic_vector(sum1); -- Convert back to std_logic_vector
```

if A + B = "110"
(i.e. A = "11"
B = "11")
sum1 = "0110"

FUNCTION 1



S[3:0] = 0001 → FUNCTION 1

A	B	y
01	01	0110 ✗
01	10	0011 ✓
10	01	0011 ✓
11	11	0110 ✓

y = A+B (signed)

← This is wrong now

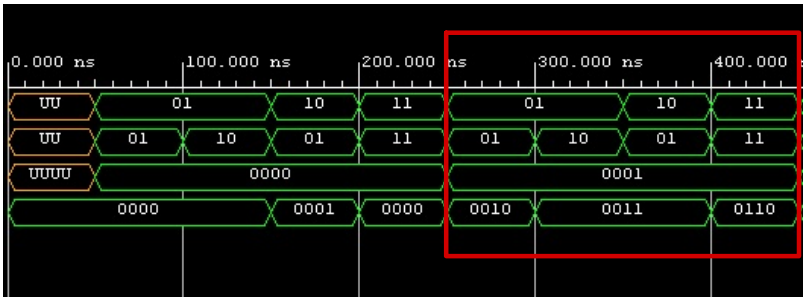
FUNCTION 1 REVISION 3

```
--Function 1
A_signed <= signed(A); -- convert signed to unsigned, input A
B_signed <= signed(B); -- convert signed to unsigned, input B
sum <= (A_signed + B_signed); -- signed addition

-- using the concatenation operator (&), sum(1) is the MSB of sum variable
--sum1 <= sum(1) & sum(1)& sum; -- resulted in errors in function 1 simulation
-- sum1 <= "0" & "0" & sum;
process(A, B, sum)
begin
   if A = "11" and B = "11" then
       sum1 <= "0110";
       else sum1 <= "0" & "0" & sum;
   end if;
end process;

-- revised function 1 logic
--sum1 <= ("0"& sum); -- Trunacte to 4-bits
f1 <= std_logic_vector(sum1); -- Convert back to std_logic_vector
```

changed to output sum <= "0110" only when A = "11" and B = "11"



S[3:0] = 0001

A	B	y
01	01	0010 ✓
01	10	0011 ✓
10	01	0011 ✓
11	11	0110 ✓

Function 2 Verification

S[3:0] = 0010 FUNCTION 2 → Arithmetic shift right of A by B positions

y = A ASR B

A	B	y
01	01	0000
01	10	0000
10	01	0011
11	11	0000

EXPECTED OUTPUT

A = 01 B = 01

shift A by 1 position

A = 01 → shift right, bit is "thrown away"

New A value

A = 00

y = 0000 (zero padding, new A value)

REFER TO S[3:0] = 0010 IN SIMULATION

Function 3 Verification

S[3:0] = 0100 FUNCTION 3

y = AB

A	B	y	
01	01	0001	✓
01	10	0010	X
10	01	0010	X
11	11	1001	X

EXPECTED OUTPUT

A	B	AND Y = AB
0	0	0
0	1	0
1	0	0
1	1	1

01 AND 01 = 01

01 AND 10 = 00

10 AND 01 = 00

11 AND 11 = 11

SIMULATION OUTPUT