Practical 5: ALU

Ronak Mehta (MHTRON001) and Vikyle Naidoo (NDXVIK005)
University of Cape Town, South Africa
EEE3096S – 2019

***Abstract* - This practical was aimed at understanding the Arithmetic Logical Unit (ALU) and the variety of arithmetic, logical and bit shift operations it can perform.**

I. ALU DESIGN

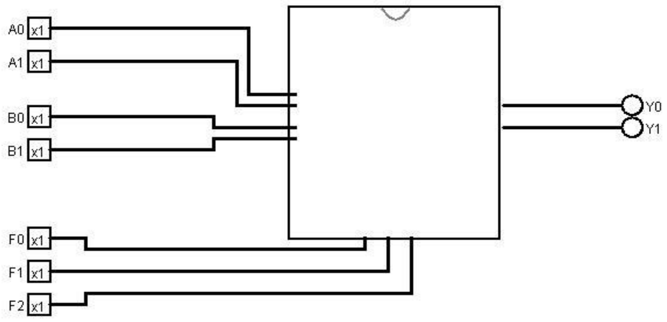


Figure 1: Simple ALU Design

Figure 1 above is a representation of the 2-bit wide ALU indicating its inputs, outputs and a black box chip containing the logical and arithmetic sub circuit blocks. The inputs are 2-bit parallel buses (A0, A1, B0, B1) and a 3-bit opcode (F0, F1, F2) which selects a specific arithmetic, logical or shift operation to be performed by the ALU on the two inputs A and B; and the output is a 2-bit Y (Y0, Y1).

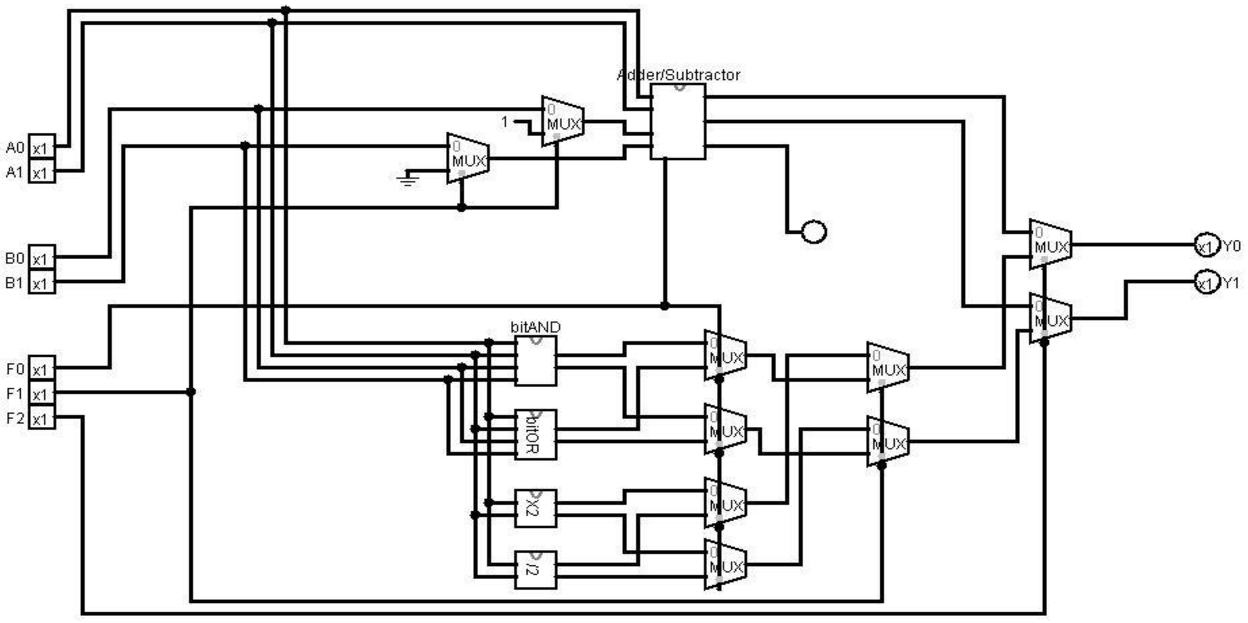


Figure 2: Sub circuits making up the ALU black box chip

Figure 2 above gives a detailed view of all the sub circuits used in making the bigger ALU system with its multiplexers. The sub circuit building blocks used in this report were the Adder/Subtractor block consisting of four sub circuits (namely Addition, Subtraction, Increment and Decrement), Multiply by 2 block (X2), Divide by 2 block (/2), Bitwise AND block (bitAND) and Bitwise OR block (bitOR). These individual sub circuits were interlinked using multiplexers (MUX).

II. SUB CIRCUIT BUILDING BLOCKS

These are the individual logical, arithmetic and bit shift operations building blocks which form a part of the bigger 2-bit wide ALU system.

A. ADDER/SUBTRACTOR BLOCK

This block consists of four operations:

i) 2-bit Full Addition:
The role of this sub circuit in the bigger system is to give the sum between the two inputs. This gives the sum of two 2-bit inputs including a carry over. *Figure 3* below illustrates that the 2-bit full adder is constructed by adding up two 1-bit full adders with inputs A and B and Cin to give output S and Cout. The Select line is set at 0. From *Figure 4* below, it is seen that B and Sel are connected to an XOR gate and its output is added to its corresponding input A bit using the Adder arithmetic block from Logisim of which gives the final output S. The truth table for this adder sub circuit is shown in *Table 1* below. When the opcode in the ALU is set to '000', this 'Addition' Function is selected and output at Y is the sum of the two inputs A and B.

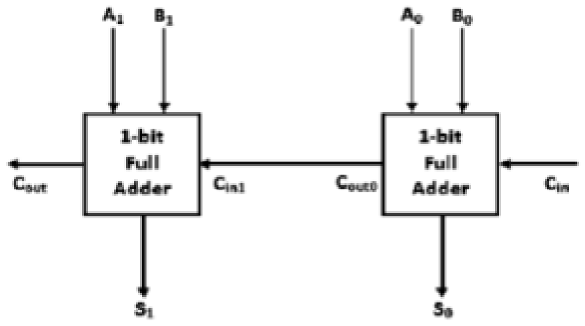


Figure 3: Illustration of a 2-bit Full Adder

ii) 2-bit Full Subtraction:
The role of this sub circuit in the bigger system is to give the difference between the two inputs. This gives the difference of two 2-bit inputs including a borrow over. It has two inputs A and B and a borrow bit to give the difference output and borrow. The Select line here is set at 1.
From *Figure 5* below, it is seen that the connections for a 2-bit full subtractor is very similar to that of a 2-bit full Adder except that for the Subtractor, there is inversion of the Select line. Input B and Sel are connected to an XOR gate and its output is added to its corresponding A bit using the Adder arithmetic block from Logisim of which gives the final output S. The truth table for this subtractor sub circuit is shown in *Table 2* below. When the opcode in the ALU is set to '001', this 'Subtraction' Function is selected and the output at Y is the difference of the two inputs A and B.

iii) Increment:
The role of this sub circuit in the bigger system is to increase the value of input A by 1. i.e. Y = A+1
To achieve this using *Figure 4,* input B is kept constantly at 1 (B0, B1 = 1) and the Select line is set at 0 which indicates to add from the previous explanation of 2-bit full Adder. When the opcode in the ALU is set to '010', this 'Increment' function is selected and the output at Y is an increment of A by one.

iv) Decrement:
The role of this sub circuit in the bigger system is to decrease the value of input A by 1. i.e. Y = A-1
To achieve this using *Figure 4,* input B is kept constantly at 1 (B0, B1 = 1) and the Select line is set at 1 which indicates to subtract from the previous explanation of 2-bit full Subtractor. When the opcode in the ALU is set to '011', this 'decrement' function is selected and the output at Y is a decrement of A by one.

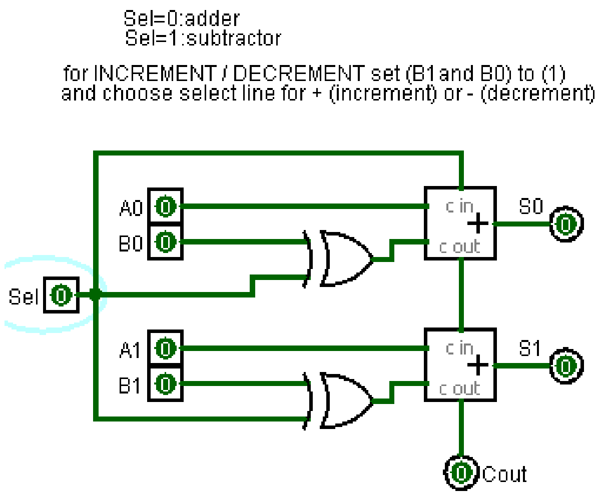

Sel=0:adder
Sel=1:subtractor
for INCREMENT / DECREMENT set (B1 and B0) to (1)
and choose select line for + (increment) or - (decrement)
A0
B0
Sel
A1
B1
c in
c out
S0
S1
Cout

Figure 4: Sub-circuit for 2-bit full Adder block

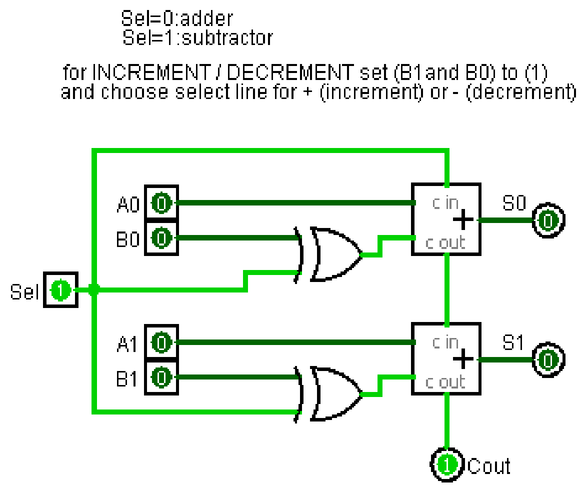
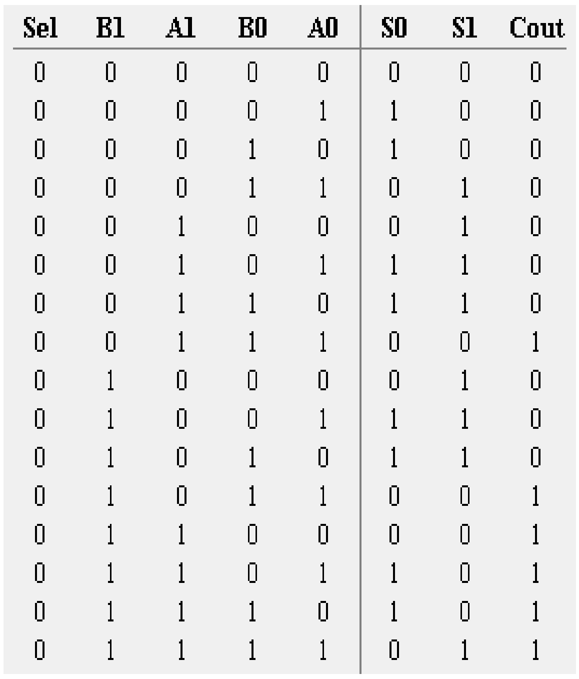
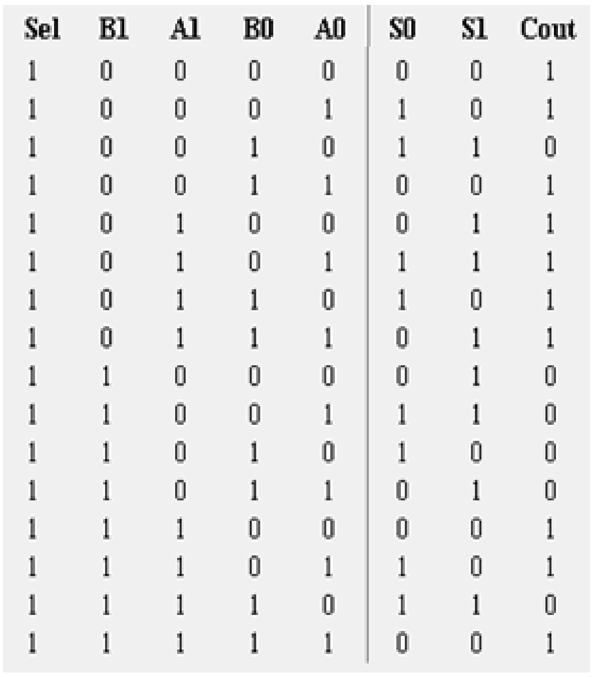

Sel=0:adder
Sel=1:subtractor
for INCREMENT / DECREMENT set (B1 and B0) to (1)
and choose select line for + (increment) or - (decrement)
A0
B0
Sel
A1
B1
c in
c out
S0
S1
Cout

Figure 5: Sub-circuit for 2-bit full Subtractor block



Sel	B1	A1	B0	A0	S0	S1	Cout
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0
0	0	0	1	0	1	0	0
0	0	0	1	1	0	1	0
0	0	1	0	0	0	1	0
0	0	1	0	1	1	1	0
0	0	1	1	0	1	1	0
0	0	1	1	1	0	0	1
0	1	0	0	0	0	1	0
0	1	0	0	1	1	1	0
0	1	0	1	0	1	1	0
0	1	0	1	1	0	0	1
0	1	1	0	0	0	0	1
0	1	1	0	1	1	0	1
0	1	1	1	0	1	0	1
0	1	1	1	1	0	1	1

Table 1: Truth table for 2-bit Full adder



Sel	B1	A1	B0	A0	S0	S1	Cout
1	0	0	0	0	0	0	1
1	0	0	0	1	1	0	1
1	0	0	1	0	1	1	0
1	0	0	1	1	0	0	1
1	0	1	0	0	0	1	1
1	0	1	0	1	1	1	1
1	0	1	1	0	1	0	1
1	0	1	1	1	0	1	1
1	1	0	0	0	0	1	0
1	1	0	0	1	1	1	0
1	1	0	1	0	1	0	0
1	1	0	1	1	0	1	0
1	1	1	0	0	0	0	1
1	1	1	0	1	1	0	1
1	1	1	1	0	1	1	0
1	1	1	1	1	0	0	1

Table 2: Truth table for 2-bit Full subtractor

B. MULTIPLY BY 2 BLOCK

The role of this sub circuit in the bigger system is to multiply the value of input A by 2 i.e. Y = A*2. In order to achieve this, a 2-bit logical left shifter is used. A logical left shift operation shifts the binary bits of input A to the left and replaces the empty bits by zeros.
As seen from *Figure 6* below, this operation can be constructed using two 1-bit multiplexers and forcing the shift to be 1. The shift has to be 1 in order to achieve a multiplication by 2 because when the shift is 0, the output bit remains same as the corresponding input bit. When Shift =1, Y0 = 0 and Y1 = A0. When the opcode in the ALU is set to '100', this 'Multiply by 2' function is selected. The truth table for this operation is seen in *Table 3* below.

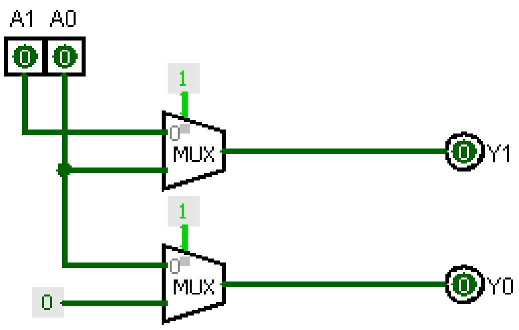
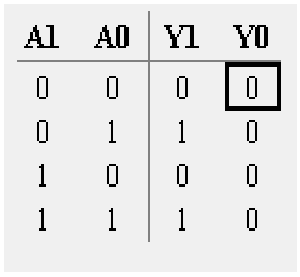


Figure 6: Sub-circuit for multiply by 2 block



A1	A0	Y1	Y0
0	0	0	0
0	1	1	0
1	0	0	0
1	1	1	0

Table 3: Truth table for multiply by 2 block

C. DIVIDE BY 2 BLOCK

The role of this sub circuit in the bigger system is to divide the value of input A by 2 i.e. Y = A/2. In order to achieve this, a 2-bit logical right shifter is used. A logical right shift operation shifts the binary bits of input A to the right and replaces empty bits by zeros.
As seen from *Figure 7* below, this operation can be constructed using two 1-bit multiplexers and forcing the shift to be 1. The shift has to be 1 in order to achieve a division by 2 because when the shift is 0, the output bit remains same as the corresponding input bit. When Shift =1, Y0 = A1 and Y1 = 0. When the opcode in the ALU is set to '101', this 'Division by 2' function is selected. The truth table for this operation is seen in *Table 4* below.

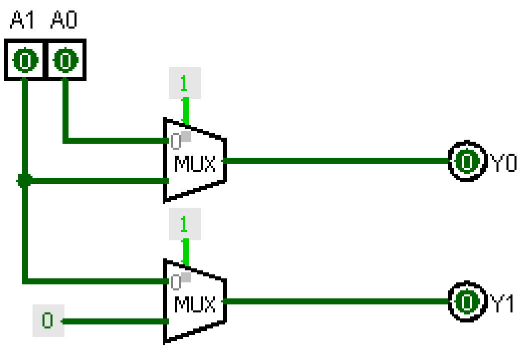
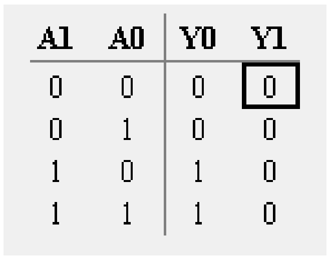


Figure 7: Sub-circuit for divide by 2 block



A1	A0	Y0	Y1
0	0	0	0
0	1	0	0
1	0	1	0
1	1	1	0

Table 4: Truth table for divide by 2 block

D. BITWISE AND BLOCK

The role of this sub circuit in the bigger system is to compare each bit of input A to the corresponding bit of input B, and iff both bits are 1, only then the corresponding output Y bit is set to 1 otherwise it is set to 0. As seen from *Figure 8* below, this bitwise AND function is obtained by connecting each corresponding input bits of A and B with an AND gate. When the opcode in the ALU is set to '110', this 'bitwise AND' function is selected. The truth table for this operation is seen in *Table 5* below.

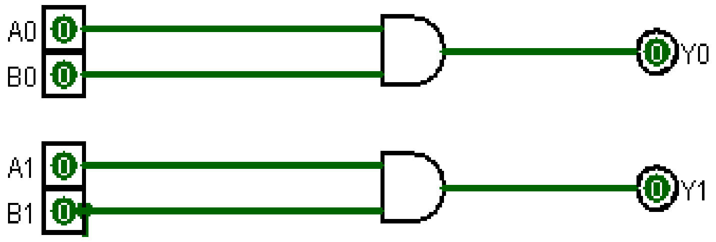
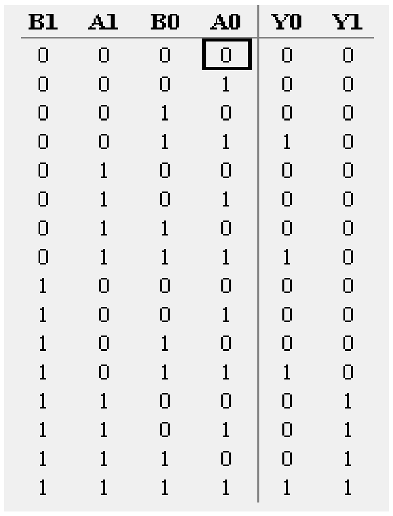


Figure 8: Sub-circuit for bitwise AND block



B1	A1	B0	A0	Y0	Y1
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	1

Table 5: Truth table for bitwise AND block

E. BITWISE OR BLOCK

The role of this sub circuit in the bigger system is to compare each bit of input A to the corresponding bit of input B, and if any one or both bits are 1, then the corresponding output Y bit is set to 1. As seen from *Figure 9* below, this bitwise OR function is obtained by connecting each corresponding input bits of A and B with an OR gate. When the opcode in the ALU is set to '111', this 'bitwise OR' function is selected. The truth table for this operation is seen in *Table 6* below.

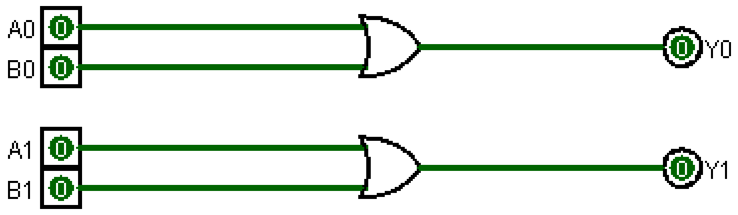
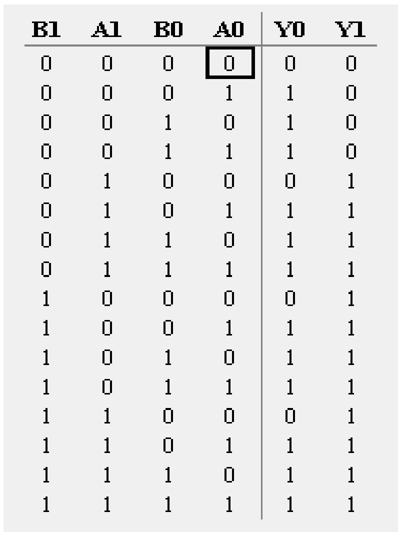


Figure 9: Sub-circuit for bitwise OR block



B1	A1	B0	A0	Y0	Y1
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	1	1
1	0	1	1	1	1
1	1	0	0	0	1
1	1	0	1	1	1
1	1	1	0	1	1
1	1	1	1	1	1

Table 6: Truth table for bitwise OR block