Experiment - 3 **Monsoon 2018**

Programmable 1-bit ALU

In this experiment, an Arithmetic and Logic Unit (ALU) capable of performing 8 Arithmetic/Logic functions on 1-bit operands, as listed in **Table 3.1**, will be designed, assembled and tested. Note that the first 4 functions are Logic functions generating 1-bit output Y_0, while the last four are Arithmetic functions generating 2-bit output Y_1Y_0. The circuit will consist of two 8-input multiplexers (74LS151), one quad 2-input multiplexer (74LS157) and one quad 2-input XOR gate (74LS86), all belonging to the TTL family.

Table 3.1 ALU Function Table

$F_2F_1F_0$	ALU Function	Y_1	Y_0
000	0 (Zero)	-	0
001	A OR B	-	A + B
010	A AND B	-	A • B
011	A EXOR B	-	A ⊕ B
100	A PLUS B	Carry	Sum
101	A MINUS B	Borrow	Difference
110	A PLUS B PLUS C	Carry	Sum
111	A MINUS B MINUS C	Borrow	Difference

The pin connections of these ICs are given in **Fig. 3.1** below. X0, X1,.....denote the data inputs and Q denotes the data output for each multiplexer. S is the select input for a 2-input multiplexer and S_2, S_1, S_0 are the select inputs for an 8-input multiplexer. Thus for a 2-input multiplexer, Q = X0 if S = 0 and Q = X1 if S =1, while for an 8-input multiplexer, the output Q = Xn (selected data input) if $S_2S_1S_0$ = n (in binary code). EN' is the (negative-logic) output enable input, i.e. the corresponding multiplexer output is equal to the selected data input only if EN' = 0. Q-outputs of multiplexers are LOW if EN' = 1.

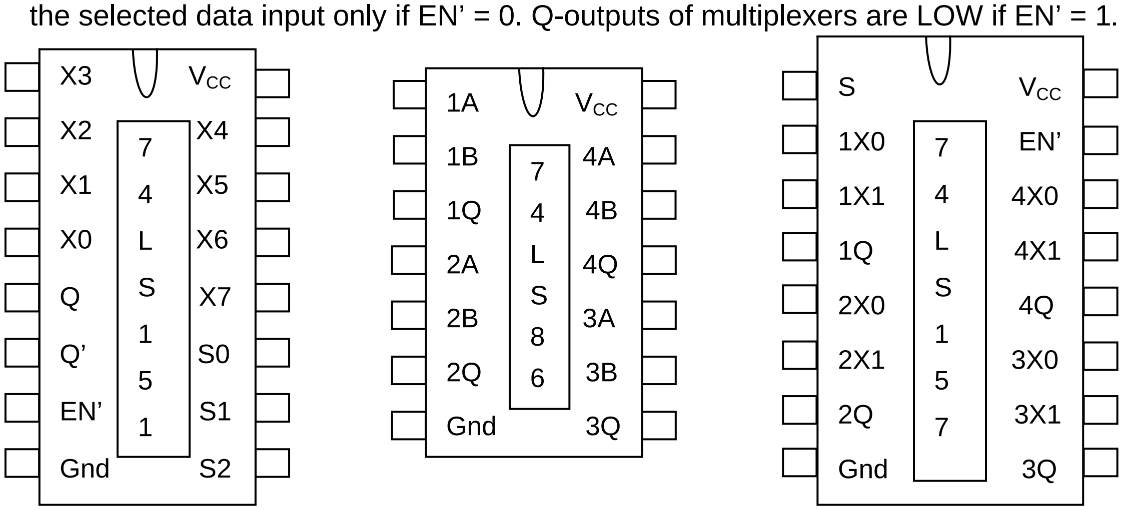

Fig. 3.1 Pin Connections of the ICs used

1. Test the three given multiplexer chips one by one by connecting V_{CC} and Gnd appropriately and applying EN' and appropriate inputs (X0, X1….) from the Input Switches. Verify the multiplexer function by tabulating the values of the Q output(s) for all input combinations. Test all gates in the given quad XOR chip as done in Experiment 2.

2. The final ALU output bits $\mathbf{Y_0}$ and $\mathbf{Y_1}$ will be generated by the two 8-input multiplexers – referred to as $\mathbf{MUX_0}$ and $\mathbf{MUX_1}$ respectively. The required data, select and output enable inputs of $\mathbf{MUX_0}$ and $\mathbf{MUX_1}$ are shown in **Fig. 3.2**. Note that $\mathbf{MUX_0}$ is always enabled, while **MUX_1** is enabled only when $F_2 = 1$, i.e. for Arithmetic functions only. This is because **Y_1** is required only to provide the CARRY/BORROW output for Arithmetic functions. Verify theoretically that **MUX_0** and **MUX_1** do generate the outputs **Y_0** and **Y_1** as required by **Table 3.1**.

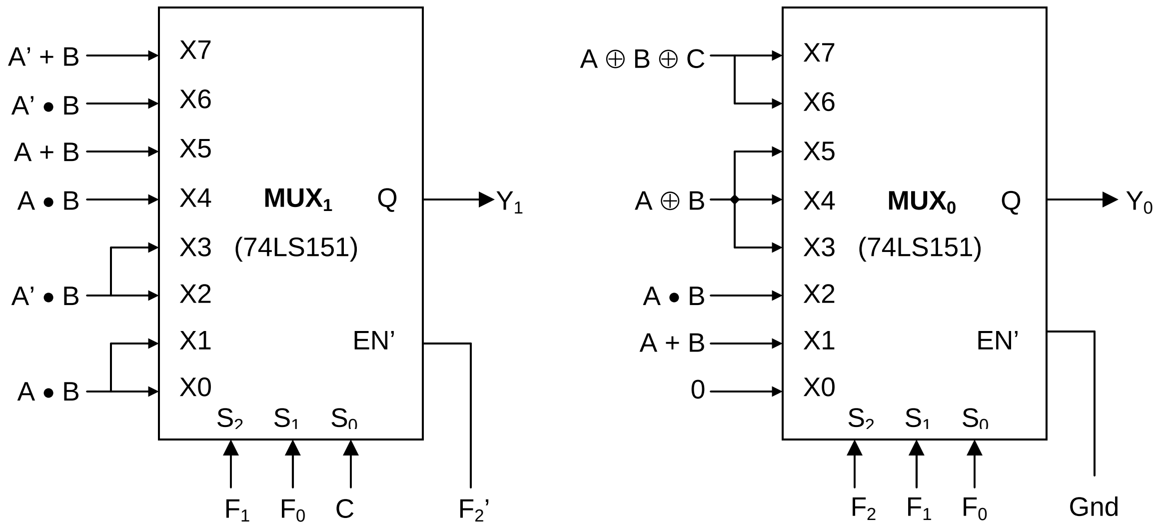

Fig. 3.2 Input connections for MUX_0 and MUX_1

3. Note that though the two 8-input multiplexers **MUX_0** and **MUX_1** require 16 inputs, they involve only 6 distinct Boolean functions of A, B, C – **A • B, A' • B, A + B, A' + B, A ⊕ B** and **A ⊕ B ⊕ C**. The first four terms are realised by four 2-input multiplexers as shown in **Fig. 3.3**. This implementation requires only one chip, whereas any implementation using gates would need more, as a single gate chip contains only one kind of gate (AND/OR/NAND/NOR). Assemble the circuit given in **Fig. 3.3** and verify its operation by actual tabulation of the observed outputs for all combinations of values of A and B applied from two Input Switches.

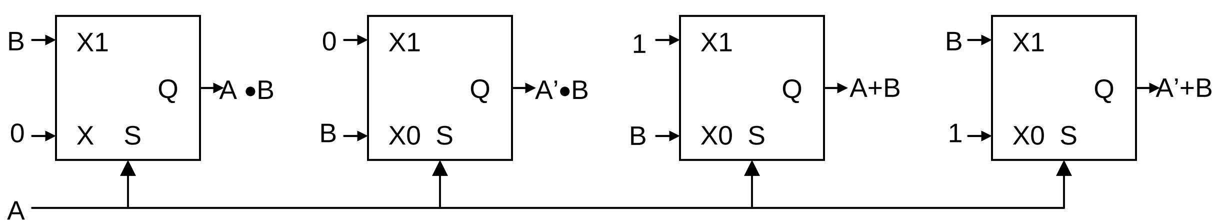

Fig. 3.3 Boolean Functions of A and B using a quad 2-input Multiplexer

4. Connect the same A and B inputs to the inputs of one of the gates in the XOR chip to generate **A ⊕ B**. Generate **(A ⊕ B) ⊕ C** by applying **(A ⊕ B)** and **C** to a second XOR gate. Verify the logic of these two outputs for all combinations of values of **A**, **B** and **C**.

5. Assemble the complete circuit by adding two 74LS151 chips, used as **MUX_0** and **MUX_1**, to the circuit assembled so far, making connections to all the inputs of **MUX_0** and **MUX_1** according to **Fig. 3.2**. Use a third gate from the XOR chip to generate **F_2'** (= $F_2 \oplus 1$), providing the Enable input for **MUX_1**.

6. Apply all the combinations of the Function select inputs $F_2F_1F_0$ one by one and tabulate the observed outputs **Y_0** and **Y_1** for as many combinations of the data inputs A, B, C as possible. Verify that the tabulated results conform to the ALU functions given in **Table 3.1**.