Date: **04/02/2026** **CS29204**: Switching Circuits Laboratory

Assignment 4

Design of combinational circuits using functional blocks

Group 36
Dipam Sen (24CS10059)
Mitali Laddha (24CS10111)

Part (a)

Aim

To implement an 8-to-1 multiplexer using two 4-to-1 multiplexers and additional gates. Using the 8-to-1 multiplexer, to implement the 4-variable function $F(A, B, C, D) = \Sigma(2, 3, 5, 8, 10, 11, 12)$.

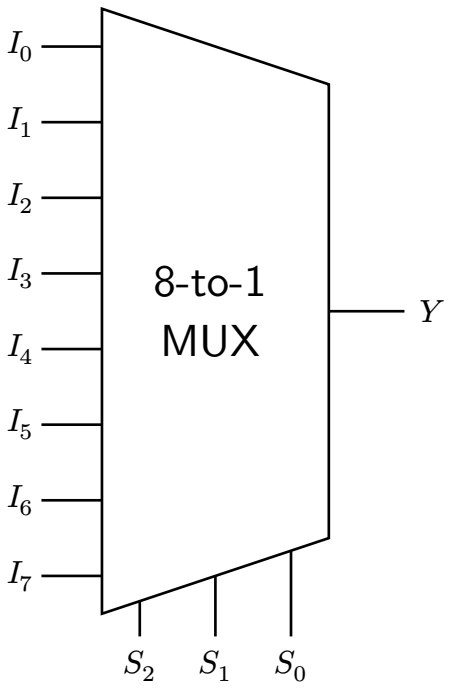
Apparatus

- Breadboard
- IC 74LS173 (TTL Dual 4x1 MUX)
- IC 74LS04 (TTL Hex Inverter)
- IC 74LS00 (TTL Quad NAND Gate)
- Wires, wire cutter, tweezers

Theory

A **multiplexer** is a functional component which acts as a multi-way switch. A 2^m-to-1 multiplexer has 2^m input lines, and m select lines, and depending on the value of the select lines, the output is equal to one of the input lines.

An 8-to-1 MUX has 8 input lines $I_0, ..., I_7$ and three select lines S_2, S_1, S_0. Depending on the value of $S = (S_2S_1S_0)$, the corresponding input line is selected as the output.



$$S = (S_2S_1S_0)$$

$$\begin{aligned} \text{If } S = 000, \text{ then } Y &= I_0. \\ S = 001, \text{ then } Y &= I_1. \\ S = 010, \text{ then } Y &= I_2. \\ &\vdots \\ S = 111, \text{ then } Y &= I_7. \end{aligned}$$

Figure 1: 8-to-1 multiplexer

The following is a standard construction for an 8-to-1 multiplexer, using smaller multiplexers:

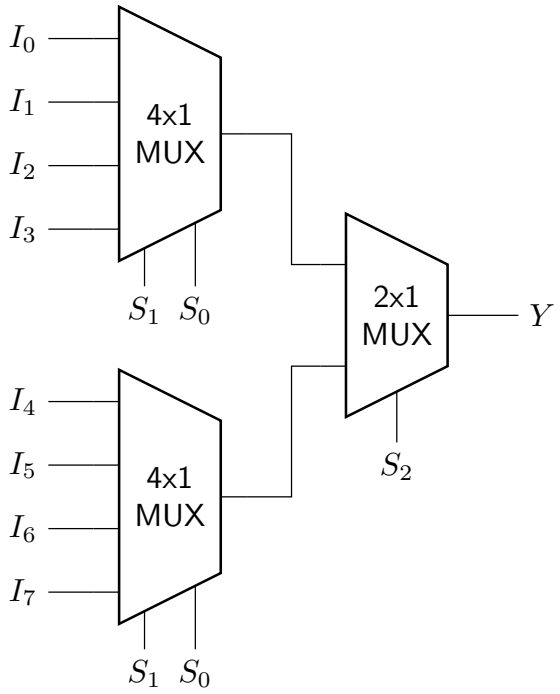


Figure 2: 8-to-1 multiplexer using 4-to-1 and 2-to-1 multiplexers

For this assignment, we can use two 4-to-1 multiplexers, but we need to implement the additional functionality (of the 2-to-1 multiplexer) using other gates. We know that for a 2x1 MUX, the output can be written as $S'A + SB$, if A and B are its inputs. So we can use NOT gates and NAND gates to implement this functionality. Figure 3 shows the final circuit diagram for the 8-to-1 multiplexer.

Once we have implemented the 8-to-1 multiplexer, we can use it to implement the 4-variable function $F(A, B, C, D) = \Sigma(2, 3, 5, 8, 10, 11, 12)$. More generally, we can implement any $(m + 1)$-variable function using an 2^m-to-1 multiplexer. To do this, we first draw the truth table of the function. (Table 1)

Now, in the 8x1 MUX, we will put the inputs A, B, C to the select lines S_2, S_1, S_0. These inputs will select a particular input line of the MUX. So, for each pair of rows in the truth table, we check the output of the MUX, which will be either 0, 1, D or D'. Accordingly, we apply the corresponding input to the input lines of the MUX. The output of the MUX will give the output of the function. Figure 4 is the circuit diagram for the realisation of F.

A	B	C	D	F
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Table 1: Truth table for F

Circuit Diagrams

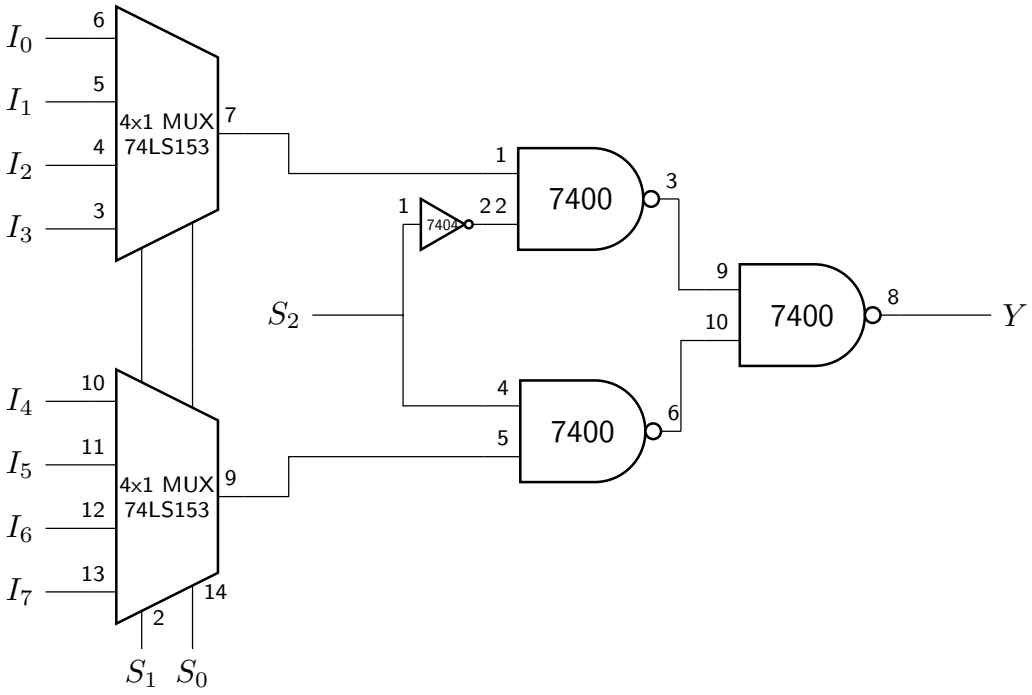


Figure 3: Implementing a 8x1 MUX using two 4x1 MUXes and other gates

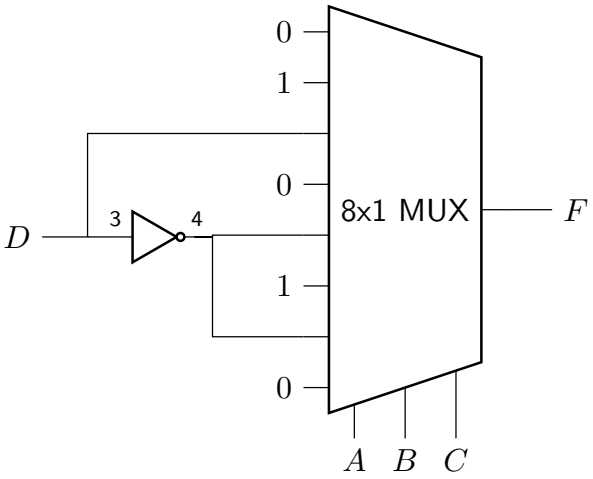


Figure 4: Realisation of $F(A, B, C, D)$ using an 8x1 MUX.

Procedure

1. Realise a 8x1 MUX using the dual 4x1 MUXes (IC 74LS153) and other gates.
2. Verify the functioning of the 8x1 MUX by applying different input combinations to the input and select lines of the MUX.
3. Realise the 4-variable function $F = \Sigma(2, 3, 5, 8, 10, 11, 12)$ using the 8x1 MUX.
4. Verify the circuit by checking the truth table of F.

Result

The 8x1 MUX correctly works as required. The realisation of F works as intended, and gives an output of 1 for the input combinations $2, 3, 5, 8, 10, 11, 12$.

Part (b)

Aim

To design and implement a 4-bit three input data selector circuit with 2 select lines, using the 74LS157 (Quad 2-to-1 multiplexer).

Apparatus

- Breadboard
- IC 74LS157 (Quad 2x1 MUX)
- Wires, wire cutter, tweezers

Theory

We have three 4-bit inputs $A = (A_3, A_2, A_1, A_0)$, $B = (B_3, B_2, B_1, B_0)$ and $C = (C_3, C_2, C_1, C_0)$. We want to select one of the three inputs, depending on the value of two select lines S_1 and S_2:

$$\text{If } S_1 = 0, S_2 = 0 \text{ then } F = A$$
$$\text{If } S_1 = 0, S_2 = 1 \text{ then } F = B$$
$$\text{If } S_1 = 1, S_2 = \text{X} \text{ then } F = C$$

For a single bit, we can perform this three-way selection by using two 2-to-1 multiplexers:

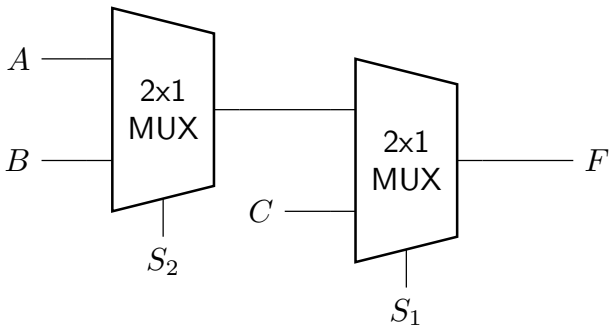


Figure 5: Simple 3-way selector using two 2-to-1 multiplexers

But, our inputs are 4-bit numbers, so we need 4 such pairs of 2x1 MUXes, to get each bit of the output from the corresponding bits of the inputs.

Circuit Diagrams

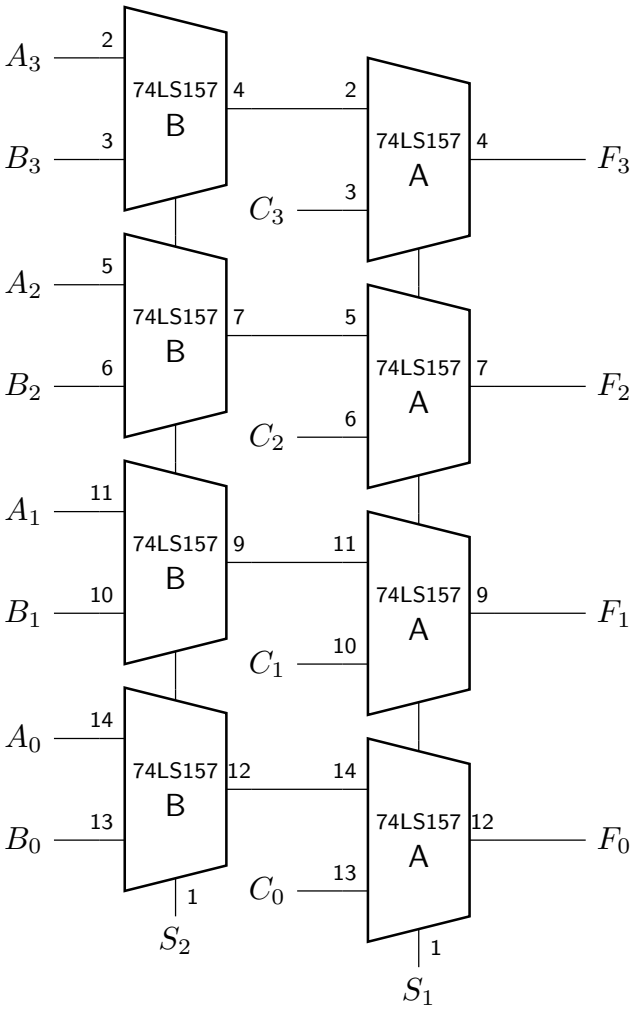


Figure 6: Implementing a 4-bit 3-way selector using two 2x1 MUXes

Procedure

1. Using two ICs 74LS157, realise the circuit shown in Figure 6.
2. Connect the twelve inputs, two select lines and four outputs of the circuits.
3. Verify the circuit by checking the truth table.

Result

The circuit works as required. The realisation of the 4-bit 3-way selector works as intended, and correctly selects the corresponding input depending on the value of the select lines.

Part (c)

Aim

To design a 4-bit binary to hexadecimal 7-segment decoder using a 4K × 8 EPROM.

Apparatus

- Breadboard
- IC 2732 (4K EPROM)
- Common anode 7-segment display
- Wires, wire cutter, tweezers

Theory

We need to design a 4-bit binary to hexadecimal 7 segment decoder. The decoder will take a 4-bit binary number as input, and output a 7-segment display. We will use a 4K EPROM to design the decoder.

Firstly, we need to know the mapping between each 4 bit binary number, and the corresponding LED segments that will be lit in the display.

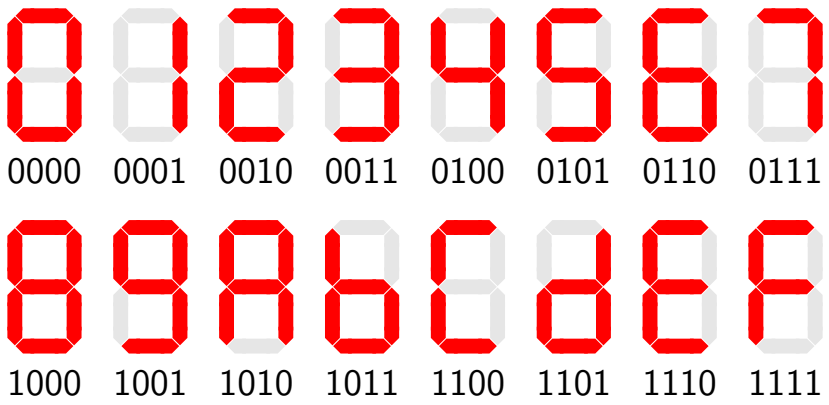


Figure 7: Outputs of the 7-segment display for different binary input combinations

Based on this, we can write a truth table for the decoder. The decoder will take a 4-bit binary number as input, and outputs 7 bits (labelled a to g), corresponding to the 7 segments of the display (Table 2).

Now, we need to program the 4K EPROM with the data from this truth table. We will use addresses 0 through 15 to store the data. We will arbitrarily choose the MSB to be 0 and store the values of a through g in the rest of the seven bits. Note that actually, we will store the complement of the value in the EPROM – this is because our display unit is a Common Anode display, so the inputs will need to be **active low**.

Once the data is stored in the EPROM, we can read the data from the EPROM and use it to display the number on the 7-segment display.

A	B	C	D	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1
1	0	1	0	1	1	1	0	1	1	1
1	0	1	1	0	0	1	1	1	1	1
1	1	0	0	1	0	0	1	1	1	0
1	1	0	1	0	1	1	1	1	0	1
1	1	1	0	1	0	0	1	1	1	1
1	1	1	1	1	0	0	0	1	1	1

Table 2: Truth table for the decoder

Address	A_3	A_2	A_1	A_0	$-$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$	Hex
0	0	0	0	0	0	0	0	0	0	0	0	1	01
1	0	0	0	1	0	1	0	0	1	1	1	1	4F
2	0	0	1	0	0	0	0	1	0	0	1	0	12
3	0	0	1	1	0	0	0	0	0	1	1	0	06
4	0	1	0	0	0	1	0	0	1	1	0	0	4C
5	0	1	0	1	0	0	1	0	0	1	0	0	24
6	0	1	1	0	0	0	1	0	0	0	0	0	20
7	0	1	1	1	0	0	0	0	1	1	1	1	0F
8	1	0	0	0	0	0	0	0	0	0	0	0	00
9	1	0	0	1	0	0	0	0	0	1	0	0	04
10	1	0	1	0	0	0	0	0	1	0	0	0	08
11	1	0	1	1	0	1	1	0	0	0	0	0	60
12	1	1	0	0	0	0	1	1	0	0	0	1	31
13	1	1	0	1	0	1	0	0	0	0	1	0	42
14	1	1	1	0	0	0	1	1	0	0	0	0	30
15	1	1	1	1	0	0	1	1	1	0	0	0	38

Table 3: Data to write to the EPROM

Circuit Diagrams

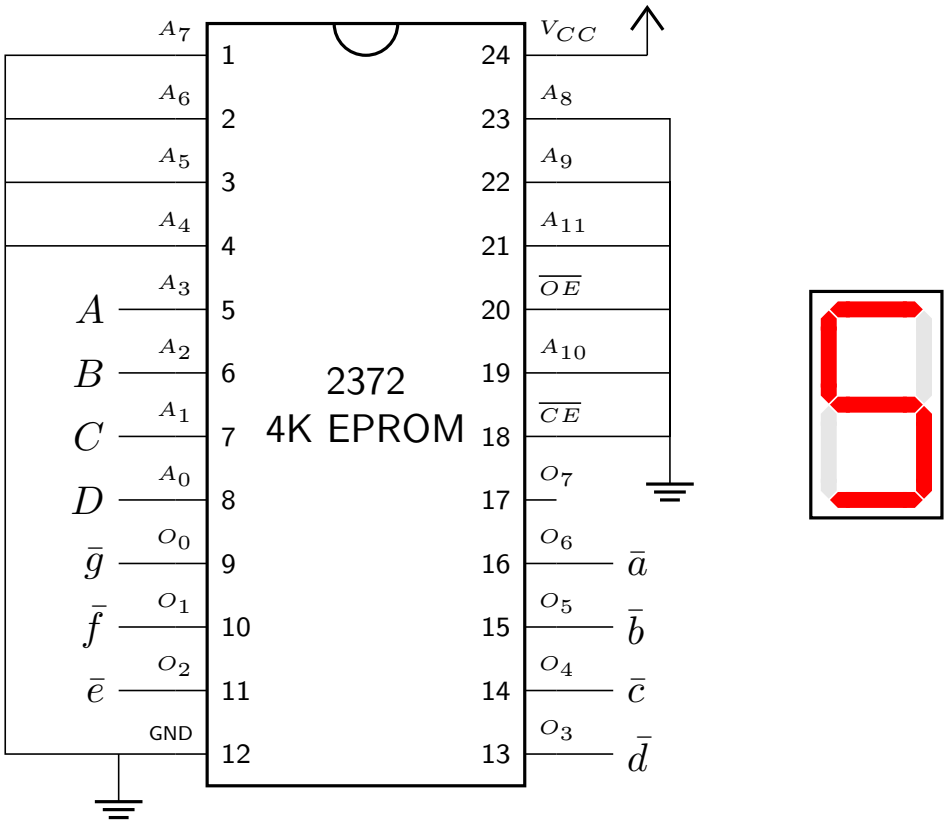


Figure 8: Connection diagram for IC 2732 (4K EPROM)

Procedure

1. Draw the truth table for the decoder.
2. Program the EPROM with the data.
3. Connect the EPROM to the breadboard. Connect the output pins of the EPROM to the inputs of the 7-segment display.
4. Verify that the circuit correctly decodes the input and displays the corresponding output on the 7-segment display.

Result

The circuit correctly displays the corresponding hex value of the input on the 7-segment display.