6 Basic MSI combinational circuits

6.1 Objectives

The most common circuits from the MSI (Medium Scale Integration) family are analyzed and their functionality is tested. We enumerate the demultiplexers, the multiplexers and the decoders. From the same family, we also study the priority encoder and the binary-Gray converter.

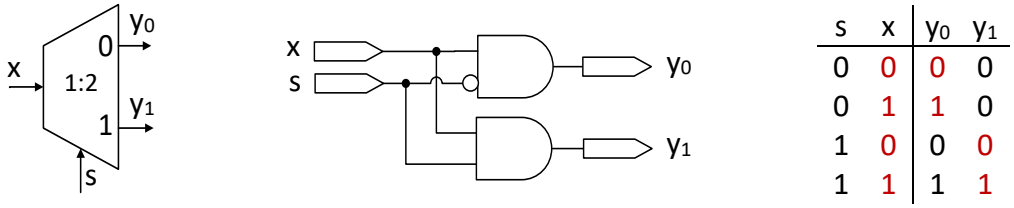
6.2 Theoretical considerations

The TTL circuits in the MSI family encompass around 50÷500 transistors. Considering their complex functionality, the MSI family is often preferred when implementing digital circuits, mostly because they reduce the size of the circuit, while replacing a significant number of basic logic gates and their interconnections. In practice, a hybrid approach is preferred, which combines MSI circuits with basic logic gates. Next, we will study several MSI circuits, with simple structures, which are commonly used in hardware synthesis.

6.2.1 Demultiplexers

A 1:*n* demultiplexer (DMUX 1:*n*) has an input **x**, *n* outputs $\mathbf{y}_i$, and *m* selection signals $\mathbf{s}_k$, where $n=2^m$. The output indicated by the value on selection signals will take the value of the input, while the rest of the outputs will be 0ed. The effect can be described as: $y_i = x$, if $i = \sum_{k=0}^{m-1} s_k \times 2^k$, otherwise $y_i = 0$.

The DMUX 1:2 implementation, its symbol and the function table are highlighted in Figure 6. 1. Calculating the Minimal Disjunctive Normal Form will conclude the following expressions for the outputs: $y_0 = x \cdot \overline{s_0}$, $y_1 = x \cdot s_1$.



s	x	y_0	y_1
0	0	0	0
0	1	1	0
1	0	0	0
1	1	1	1

Figure 6. 1 DMUX 1:2 symbol (left), its implementation (middle) and the function table (right)

Demultiplexers with a higher number of inputs can by synthesized using basic logic gates or by cascading smaller demultiplexers. For instance, a DMUX 1:4 can be implemented with three DMUX 1:2 units, cascaded on two logic levels:

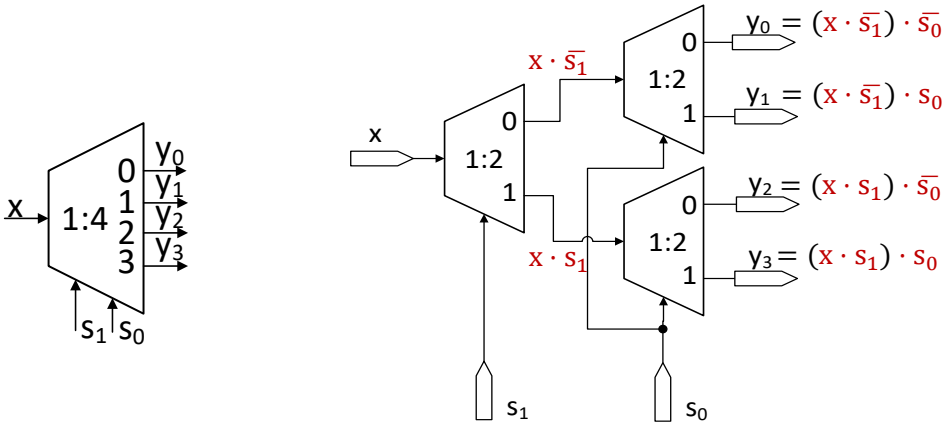


Figure 6. 2 DMUX 1:4 symbol (left) and its implementation with DMUX 1:2 units (right)

Note: When input x = 1, the demultiplexer has the functionality of a decoder, because it enables the output indicated on the selection bits.

There are demultiplexers with a higher number of inputs such as DMUX 1:8, 1:16, 1:32, etc.

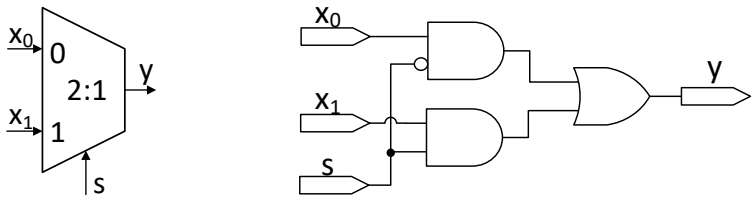
In Logisim, the demultiplexers are available in the *Plexers* library. Their size can be set using the `Select Bits` attribute, which refers to the number of selection inputs. The width of data input and data output signals can be set to 1 bit by having attribute `Data Bits = 1`. Optionally, the demultiplexer can have an additional enable input, which can be removed by setting `Include Enable = No`.

In Project Navigator, 1:4, 1:8 and 1:16 demultiplexers can be implemented using the decoders D2_4E, D3_8E, and D4_16E respectively, which are available in the *Decoder* category. The E (Enable) input of the decoder can be used as data input of the demultiplexer. The data inputs A_i can be used as selection bits. A 1:2 demultiplexer is available in the local library of the *ttl_env* project. Its name is D1_2.

6.2.2 Multiplexers

A *n*:1 multiplexer (MUX *n*:1) has *n* inputs $\mathbf{x}_i$, an output **y**, and *m* selection signals $\mathbf{s}_k$, where $n=2^m$. The output will get the value of the input indicated by the selection bits. Multiplexers are useful when one signal should be selected out of several inputs. Its effect can be expressed as: $y = x_i$, where $i = \sum_{k=0}^{m-1} s_k \times 2^k$.

The 2:1 multiplexer highlighted in Figure 6. 3 selects the input indicated by the signal s. Using the function table and calculating the Minimal Disjunctive Normal Form, the resulting expression for the output signal becomes: $y = x_0 \cdot \overline{s_0} + x_1 \cdot s_1$.



s	x_0	x_1	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Figure 6. 3 The symbol of MUX 2:1 (left), its implementation (middle) and the function table (right)

Designing multiplexers with a higher number of inputs is possible through cascading smaller multiplexers, or by using basic logic gates. For instance, a MUX 4:1 can be implemented with three multiplexers MUX 2:1, cascaded over two logic levels:

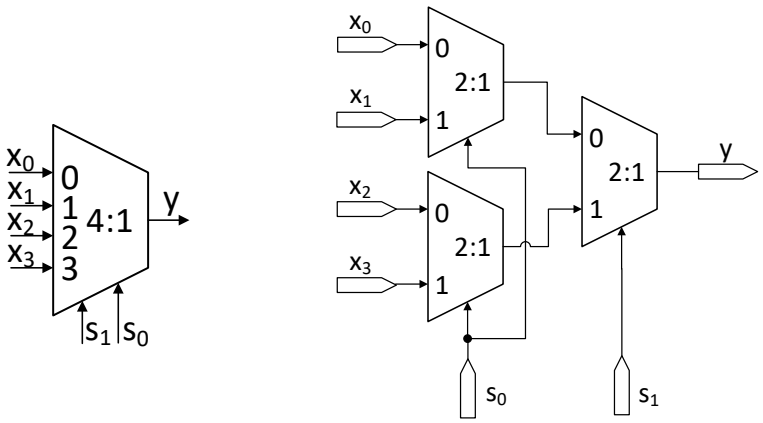


Figure 6. 4 The MUX 4:1 symbol (left) and its implementation with cascaded units (right)

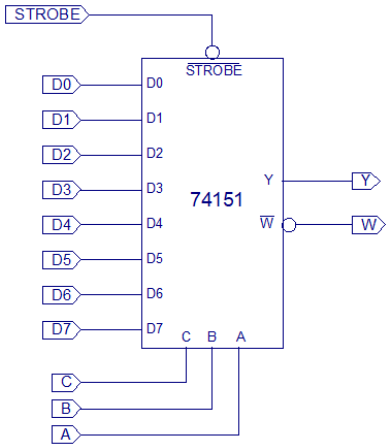
The multiplexers can be extended to a power of 2 inputs: MUX 8:1, 16:1, 32:1, etc.

In Logisim, the multiplexers are available in the *Plexers* library. Attribute `Select Bits` controls the number of selection bits; `Data Bits` controls the number of bits on data inputs and output. The enable can be removed by setting `Include Enable = No`.

In Project Navigator, the multiplexers are available in the *MUX* category. Their names are M2_1, M2_1E, M4_1E and M8_1E, respectively. The suffix E indicates the presence of the enable. It must be connected to VCC, otherwise the output will be 0.

The TTL circuit 74151 in Figure 6. 5 implements a MUX 8:1 unit with inputs D_0, …, D_7, having selections A, B, C and two complementary outputs: Y and W, respectively. Output Y is active high and output W is active low. They are related by the following expression: $W = \overline{Y}$. The selection bits generate the binary number CBA_2. There is an additional enable input called STROBE (S), which is active low. If STROBE = 0 the circuit functions normally, otherwise Y = 0 and $W = \overline{Y} = 1$.

In Logisim, the TTL circuits are available inside the *TTL* library. In the project *test_env*, the TTL circuits are available in the local library. Their name is prefixed by **TTL_**, followed by their code.



STROBE	C	B	A	Y	W
1	X	X	X	0	1
0	0	0	0	D_0	$\overline{D_0}$
0	0	0	1	D_1	$\overline{D_1}$
0	0	1	0	D_2	$\overline{D_2}$
0	0	1	1	D_3	$\overline{D_3}$
0	1	0	0	D_4	$\overline{D_4}$
0	1	0	1	D_5	$\overline{D_5}$
0	1	1	0	D_6	$\overline{D_6}$
0	1	1	1	D_7	$\overline{D_7}$

Figure 6. 5 The TTL circuit 74151 implements a MUX 8:1 unit

6.2.3 Decoders

The decoder has *n* inputs and maximum 2^n outputs. The decoder activates the output indicated by the input bits. The rest of outputs are disabled. The decoders can have active high or active low outputs.

The TTL 7442 circuit in Figure 6. 6 (left) is a BCD-decimal decoder with 4 inputs (the BCD code represented by $DCBA_2$), and 10 active low outputs, numbered 0 to 9. The function table is highlighted in Table 6. 1. For input values in range $1010_2 \div 1111_2$ the outputs are disabled (set to 1).

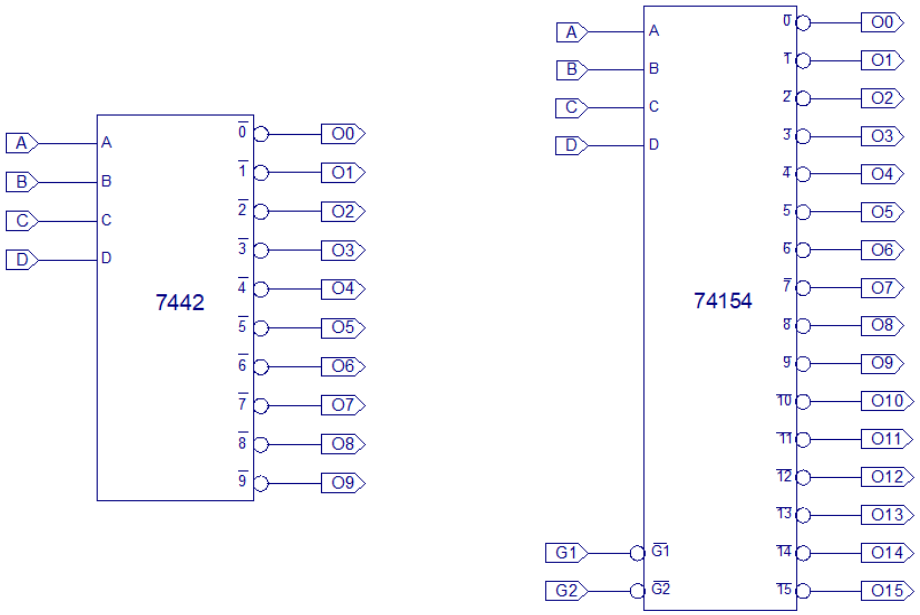


Figure 6. 6 The decimal decoder 7442 and the hexadecimal decoder 74154

The 16-bit output version is implemented by TTL 74154. This is a hexadecimal decoder DCD 4:16. Any input value in range 0÷15 will enable a corresponding output (with value 0). This circuit has two additional active low enable inputs G_1 and G_2, respectively. To enable the circuit, G_1 and G_2 should be set to 0 (GND), otherwise the outputs will be disabled (value 1).

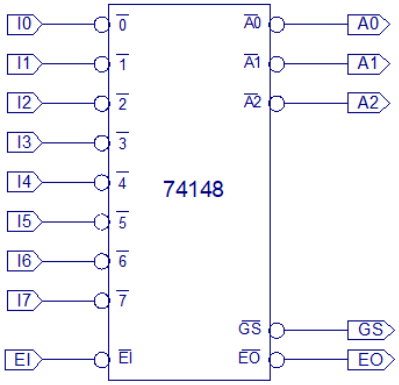
Table 6. 1 Function table of the decimal decoder 7442

BCD				Decimal									
D	C	B	A	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	1	1	1	1	1	1	1	1	1
0	0	0	1	1	0	1	1	1	1	1	1	1	1
0	0	1	0	1	1	0	1	1	1	1	1	1	1
0	0	1	1	1	1	1	0	1	1	1	1	1	1
0	1	0	0	1	1	1	1	0	1	1	1	1	1
0	1	0	1	1	1	1	1	1	0	1	1	1	1
0	1	1	0	1	1	1	1	1	1	0	1	1	1
0	1	1	1	1	1	1	1	1	1	1	0	1	1
1	0	0	0	1	1	1	1	1	1	1	1	0	1
1	0	0	1	1	1	1	1	1	1	1	1	1	0
1	0	1	0	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	1	1	1	1	1	1	1	1	1	1
1	1	0	1	1	1	1	1	1	1	1	1	1	1
1	1	1	0	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1

6.2.4 Encoders – the priority encoder

The priority encoder has 2^n inputs and n outputs. The code generated on the outputs indicates the active input with the highest priority. The priority increases with the index.

The circuit 74178 is a priority encoder with 8 inputs and 3 outputs. All terminals are active low (Figure 6. 7). Additionally, there is an active low enable input EI (0 – enable, 1 – disable) and two outputs, GS and EO, respectively. GS is enabled (GS = 0) when the output code on $A_{2:0}$ is valid. A code is invalid when the circuit is either disabled (EI = 1) or there is no active input. The EO output signals an error. It is activated (EO = 0) if the circuit is enabled and all inputs are disabled. The function table is highlighted below:



	Inputs								Outputs				
EI	0	1	2	3	4	5	6	7	A_2	A_1	A_0	GS	EO
1	X	X	X	X	X	X	X	X	1	1	1	1	1
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	X	X	X	X	X	X	X	0	0	0	0	0	1
0	X	X	X	X	X	X	0	1	0	0	1	0	1
0	X	X	X	X	X	0	1	1	0	1	0	0	1
0	X	X	X	X	0	1	1	1	0	1	1	0	1
0	X	X	X	0	1	1	1	1	1	0	0	0	1
0	X	X	0	1	1	1	1	1	1	0	1	0	1
0	X	0	1	1	1	1	1	1	1	1	0	0	1
0	0	1	1	1	1	1	1	1	1	1	1	0	1

Figure 6. 7 The priority encoder 74148 (left) and its function table (right)

6.2.5 Conversion units – the 4-bit binary-Gray converter

The converters perform the transition from one code to another. Usually, they are used for communication purposes, linking systems using different encoding schemes.

The 4-bit binary-Gray converter has 4 inputs and 4 outputs. A binary code on the inputs will generate a Gray code on the outputs (the inverse conversion is also possible). The function table is presented next:

B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

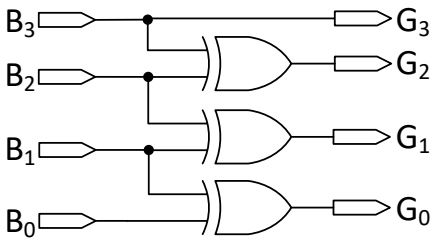


Figure 6. 8 The 4-bit binary-Gray conversion (left) and the design of the circuit (right)

Each output G_i is a separate boolean function of 4 variables (the input bits). Using minimization and boolean algebra laws, the output expressions can be reduced to: $G_3 = B_3, G_2 = B_2 \oplus B_3, G_1 = B_1 \oplus B_2, G_0 = B_0 \oplus B_1$.

6.3 Assignments

1. Implement and test on the board the 74151 demultiplexer.
2. Implement and test on the board the 7442 decimal decoder.
3. Implement and test on the board the 74148 priority encoder.
4. Implement and test in Logisim a DMUX 1:2 unit, using basic logic gates.
5. Implement and test in Logisim a DMUX 1:4 unit, by cascading DMUX 1:2 units.
6. Implement and test in Logisim a MUX 4:1 unit, by cascading MUX 2:1 units.
7. Implement and test in Logisim the hexadecimal decoder 74154.
8. Implement and test in Logisim the 4-bit binary-Gray converter.
9. Implement function $f(A, B, C, D, E) = \overline{B} \cdot \overline{C} \cdot D + C \cdot \overline{D} \cdot E + B \cdot \overline{C} \cdot \overline{E} + A \cdot \overline{B} \cdot \overline{D} + A \cdot B \cdot C \cdot D$ using one 16:1 multiplexer and constants 0, 1. The variables cannot be inverted.