# 7 Complex MSI combinational circuits

## 7.1 Objectives

The structure of several complex MSI combinational circuits is studied, such as: the multiplexer with multiple data bits on datapath, the adder, the arithmetic-logic unit and the BCD-7 segment decoder. An adder-subtractor unit is implemented using only adder units, and its functionality is analyzed. Cascading is used as strategy to extend the bit width.

## 7.2 Theoretical considerations

Complex MSI combinational circuits are widespread in applications based on digital circuits. For instance, the most frequent operations in computational devices are arithmetic and logic, therefore arithmetic-logic units are ubiquitous. Extending the number of bits on the datapath is an implicit consequence of having most computations driven on operands with a multiple of 8 bits (1 byte). In most cases the results are displayed in base 10. The BCD-7 segment decoders represent a common solution to converting binary values to the format specific to 7-segment display units.

### 7.2.1 Multiplexers and demultiplexers with multiple bits on the datapath

The multiplexers forward one input from a list to a single output. The demultiplexers forward a unique input value to one of several outputs. The selection is based on the value present on the selection bits. Both multiplexers and demultiplexers can have multiple bit datapath. For instance, the data inputs and output of a MUX 4:1 unit with 3-bit datapath consist in 3-bit data lines (buses). This can be interpreted as having three 1-bit MUX 4:1 units, sharing 2 selection bits, as in Figure 7. 1.


Figure 7. 1 The MUX 4:1 with 3-bit datapath – implementation with three 1-bit MUX 4:1 units (left) and the symbol (right)

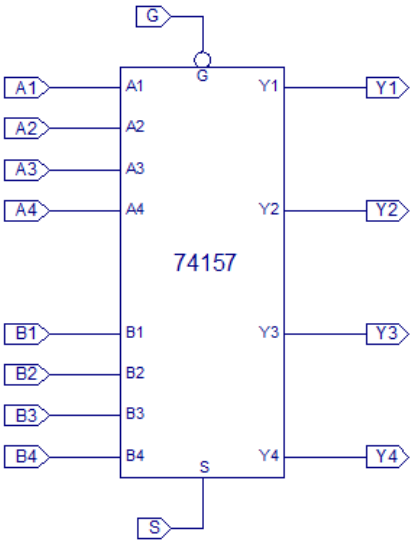
Similarly, a DMUX 1:4 with 3-bit datapath can be implemented with three 1-bit DMUX 1:4 units, sharing the selection bits, as in Figure 7. 2.


Figure 7. 2 The DMUX 1:4 unit with 3-bit datapath – implementation with three 1-bit DMUX 1:4 units (left) and the symbol (right)

The TTL 74157 implements a 4-bit MUX 2:1 unit. The symbol of the circuit is highlighted in Figure 7. 3. Data inputs $A_{4:1}$, $B_{4:1}$ and the output $Y_{4:1}$ are 4-bit wide. The selection bit S decides the input forwarded to the output:

$$Y_{4:1} = \begin{cases} A_{4:1}, \text{if } S = 0 \\ B_{4:1}, \text{if } S = 1 \end{cases} \quad (7.\ 1)$$

The G input (Strobe) enables the circuit, when G=0. If G=1, the outputs are 0ed.




| Inputs | | | | Output |
|---|---|---|---|---|
| G (Strobe) | S (Select) | $A_i$ | $B_i$ | $Y_i$ |
| 1 | X | X | X | 0 |
| 0 | 0 | 0 | X | 0 |
| 0 | 0 | 1 | X | 1 |
| 0 | 1 | X | 0 | 0 |
| 0 | 1 | X | 1 | 1 |

Figure 7. 3 Multiplexer 74157: the symbol (left) and the function table (right)

### 7.2.2 Circuits implementing arithmetic and logic operations

The basic arithmetic operations are summation and subtraction. The TTL 74283 circuit (Figure 7. 4) implements 4-bit binary summation of operands $A_{4:1}$ and $B_{4:1}$. The result is calculated on output $S_{4:1}$. The circuit has a carry input $C_0$ of rank 0, and a carry output $C_4$, of rank 4. The $C_4$ line signals the overflow. The carry lines can be used to extend the number of bits by cascading several 74283 units. **Note**: Setting $C_0$=1 is equivalent to incrementing the result by +1. The arithmetic expression implemented by 74283 is:

$$(C_4S_4S_3S_2S_1)_2 = (A_4A_3A_2A_1)_2 + (B_4B_3B_2B_1)_2 + (000C_0)_2 \quad (7.\ 2)$$

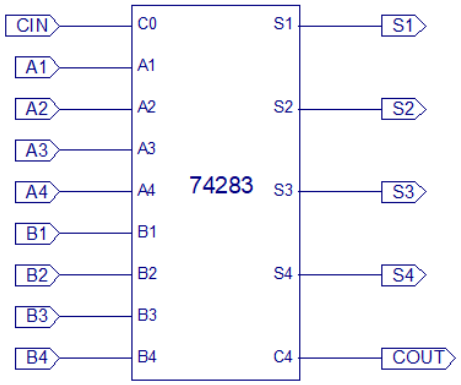



Figure 7. 4 The 4-bit adder 74283

#### 7.2.2.1 Implementing the 4-bit adder-subtractor

In 2's Complement representation, the subtraction can be implemented by adding the first term with the 2's complement of the second term. The 2's complement is the 1's complement plus 1, meaning that: $A - B = A + \overline{\overline{B}} = A + \overline{B} + 1$. On 4-bit operands the expression becomes:

$$(A_4A_3A_2A_1)_2 - (B_4B_3B_2B_1)_2 = (A_4A_3A_2A_1)_2 + (\overline{B}_4\overline{B}_3\overline{B}_2\overline{B}_1)_2 + (0001)_2 \quad (7.3)$$

In boolean algebra, an expression ɸ follows the rules: ɸ ⊕ 0 = ɸ and ɸ ⊕ 1 = $\overline{ɸ}$. Consequently, addition and subtraction admit the next XOR-based formulation:

$$\begin{cases}(A_4A_3A_2A_1)_2 + (B_4B_3B_2B_1)_2 = (A_4A_3A_2A_1)_2 + (B_4B_3B_2B_1 \oplus 0000)_2 + (0000)_2 \\ (A_4A_3A_2A_1)_2 - (B_4B_3B_2B_1)_2 = (A_4A_3A_2A_1)_2 + (B_4B_3B_2B_1 \oplus 1111)_2 + (0001)_2\end{cases}$$

Both operations can be unified as $(C_4S_4S_3S_2S_1)_2 = (A_4A_3A_2A_1)_2 + (B_4B_3B_2B_1 \oplus \text{Sel Sel Sel Sel})_2 + (0\ 0\ 0\ \text{Sel})_2$, where **Sel=0 for addition and Sel=1 for subtraction**. It is possible to implement the expression with a 74283 unit and 4 XOR gates, as follows:

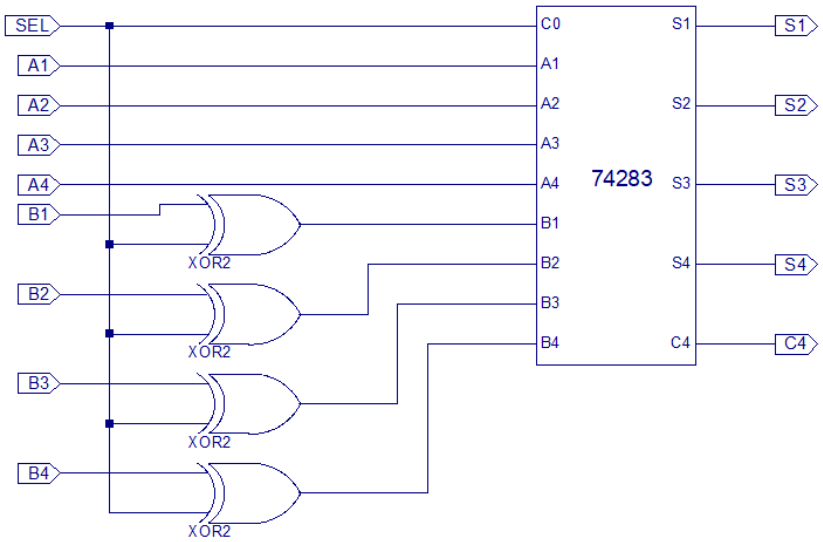



Figure 7. 5 The 4-bit adder-subtracter unit implemented with 74283

### 7.2.2.2 Implementing the 8-bit adder-subtractor

The adder-subtractor expression extended to 8 bits has the following formulation:

$$(C_8S_8S_7S_6S_5S_4S_3S_2S_1)_2 = (A_8A_7A_6A_5A_4A_3A_2A_1)_2 + \\ + (B_8B_7B_6B_5B_4B_3B_2B_1 \oplus \text{Sel Sel Sel Sel Sel Sel Sel Sel})_2 + (0\ 0\ 0\ 0\ 0\ 0\ 0\ \text{Sel})_2 \quad (7.4)$$

When implemented in hardware, the expression uses two cascaded 74283 adders and 8 XOR gates. One adder will perform summation on bits ranked 1÷4, having the Sel signal connected to its $C_0$ input. The other adder will perform summation on bits ranked 5÷8, with $C_0$ connected to the carry output $C_4$ of the other unit:

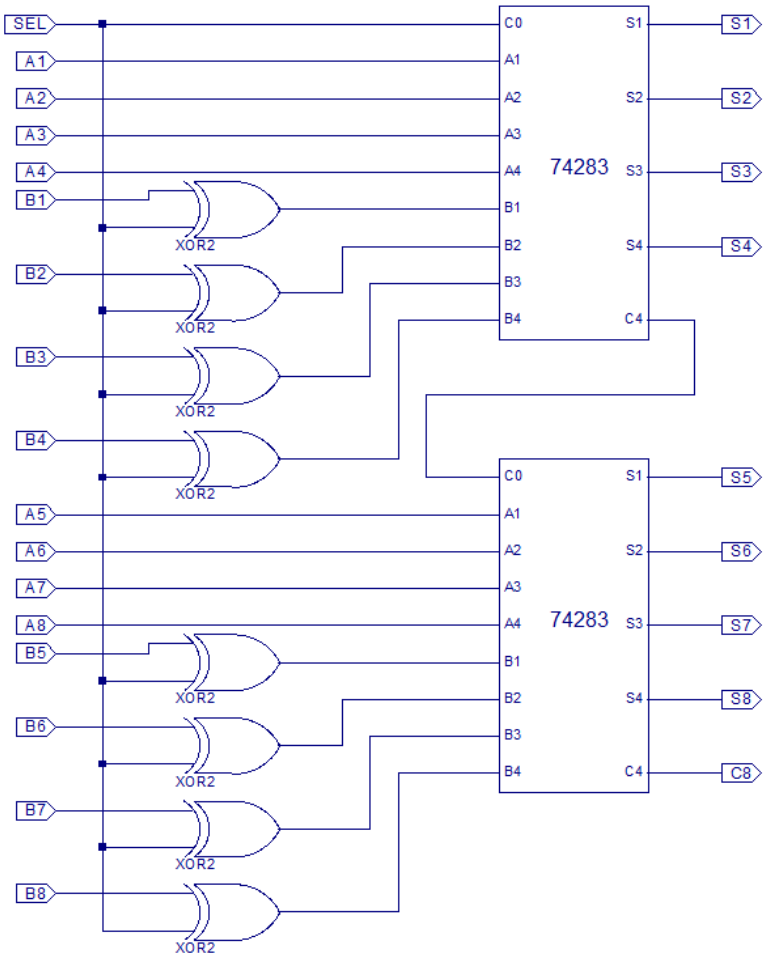
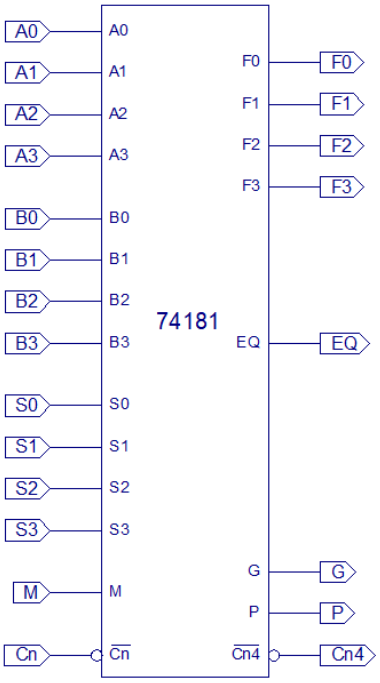



Figure 7. 6 The 8-bit adder-subtracter implemented by cascading two 74283 units

### 7.2.2.3 Arithmetic-logic units

The 74181 circuit (Figure 7. 7) performs 2's complement arithmetic and logic operations on two 4-bit operands $A_{3:0}$ and $B_{3:0}$. The current operation is defined by the binary code on selection inputs $S_{3:0}$. Each code represents two operations: a bitwise logic operation is performed when input M=1, and an arithmetic operation is performed when input M=0. The result is output on $F_{3:0}$. The carry input $C_n$ and carry output $C_{n4}$ signals are active low. They play a role in cascading several 74181 units to extend the bit width. The output EQ (also A=B) tests the equality between $A_{3:0}$ and $B_{3:0}$, when performing the operation A minus B minus 1. The table below highlights the complete set of operations:




| Selection inputs | | | | Operations | |
|---|---|---|---|---|---|
| $S_3$ | $S_2$ | $S_1$ | $S_0$ | Logic M = 1 | Arithmetic M = 0, $C_n$ = 1 |
| 0 | 0 | 0 | 0 | $\overline{A}$ | A |
| 0 | 0 | 0 | 1 | $\overline{A+B}$ | $A+B$ |
| 0 | 0 | 1 | 0 | $\overline{A}\cdot B$ | $A+\overline{B}$ |
| 0 | 0 | 1 | 1 | 0 | minus 1 |
| 0 | 1 | 0 | 0 | $\overline{A\cdot B}$ | A plus $(A\cdot\overline{B})$ |
| 0 | 1 | 0 | 1 | $\overline{B}$ | $(A+B)$ plus $(A\cdot\overline{B})$ |
| 0 | 1 | 1 | 0 | $A\oplus B$ | A minus B minus 1 |
| 0 | 1 | 1 | 1 | $A\cdot\overline{B}$ | $(A\cdot B)$ minus 1 |
| 1 | 0 | 0 | 0 | $\overline{A}+B$ | A plus $(A\cdot B)$ |
| 1 | 0 | 0 | 1 | $\overline{A\oplus B}$ | A plus B |
| 1 | 0 | 1 | 0 | B | $(A+\overline{B})$ plus $(A\cdot B)$ |
| 1 | 0 | 1 | 1 | $A\cdot B$ | $(A\cdot B)$ minus 1 |
| 1 | 1 | 0 | 0 | 1 | A plus A |
| 1 | 1 | 0 | 1 | $A+\overline{B}$ | $(A+B)$ plus A |
| 1 | 1 | 1 | 0 | $A+B$ | $(A+\overline{B})$ plus A |
| 1 | 1 | 1 | 1 | A | A minus 1 |

Figure 7. 7 Circuit 74181: the symbol (left) and its operations (right)
(+ = OR, · = AND, ⊕ = XOR, plus = summation, minus = subtraction)

### 7.2.3 Displaying decimal digits

The testing board has eight 7-segment display units used for displaying decimal digits (Figure 7. 8) [1]. A decimal digit appears on each display by lighting up a subset of the 7 segments (Figure 7. 9). The status of the segments is controlled by 7 signals, called *cathodes*, indexed A÷G. The cathodes are shared by all displays. Each display has an individual enable signal, called *anode*. The anodes of the displays are indexed 0÷7, in right to left order. **Note**: Because cathodes are shared, all displays will highlight the same decimal digit, however there are techniques to display different digits, for decimal values with more digits. The anodes and cathodes are active low. By default, the cathodes have a value of 1 and the anodes have a value of 0, meaning that displays are active, but their segments are inactive (by default, the displays are blank).


Figure 7. 8 The 7-segment display units on the testing board

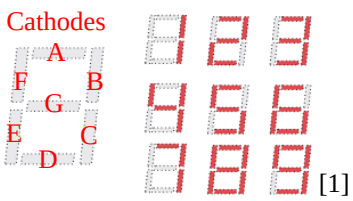



Figure 7. 9 The segments on the 7-segment display and the decimal digits configuration

Circuit 7447 is a BCD-7 segment decoder, which converts 4-bit binary codes to the cathodes' configuration associated with the corresponding decimal digit on the 7-segment display. The complete set of associations is highlighted in Figure 7. 10.

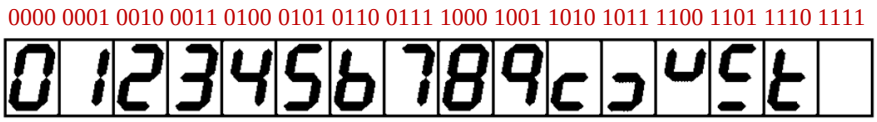



Figure 7. 10 Symbols on the 7-segment display associated to binary codes

The outputs of the decoder representing the A÷G cathodes are active low (Figure 7. 11). The inputs $\overline{LT}$, $\overline{RBI}$, $\overline{BI}/\overline{RBO}$ are reserved for testing, therefore they are disabled in a normal regime, by connecting to VCC.

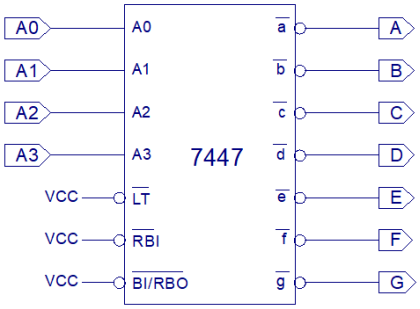



Figure 7. 11 The 7447 BCD-7 segments decoder

In Logisim, the 7-segment display is available in *Input/Output* library and has active high inputs, therefore the outputs of the 7447 decoder must be inverted with NOT gates.

In Project Navigator, the cathodes and the anodes are found in a separate section of the .ucf file:

```
## 7 segment display
NET "A" LOC=T10 | IOSTANDARD=LVCMOS33; # cat a
NET "B" LOC=R10 | IOSTANDARD=LVCMOS33; # cat b
. . .
```

### 7.3 Assignments

1. Implement and test on the board the 74157 multiplexer with 4-bit datapath.
2. Implement and test on the board the 74283 4-bit adder.
3. Implement and test on the board the adder-subtractor implemented with 74283.
4. Implement and test on the board the 7447 BCD-7 segment decoder.
5. Implement and test in Logisim the 8-bit adder using 74283 units.
6. Implement and test in Logisim the 8-bit adder-subtractor using 74283 units.
7. Implement and test in Logisim the 4-bit arithmetic-logic unit 74181.

### 7.4 Bibliography

[1] Digilent, Nexys A7 Reference Manual, [Online]. Available: https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual