Middle East Technical University - Department of Computer Engineering

CENG 232

Logic Design

Spring 2022-2023

Lab Assignment 2

Due date: 19 April 2023, 23:59

1 Introduction

This laboratory aims to get you familiar with some of the most important IC components like multiplexers and decoders. You are expected to draw the circuit using the Logisim tool with the given gates below.

2 IC Pool

- 74LS08 (AND)
- 74LS32 (OR)
- 74LS04 (NOT, Inverter)
- 74LS02 (NOR)
- 74LS00 (NAND)
- 74LS153 (Multiplexer)
- 74LS86 (XOR)
- 74LS266 (XNOR)
- 74LS138 (Decoder)
- 74LS241 (3-state Buffer, Controlled Buffer)

3 Lab Work

In this LAB, you are expected to design a simple calculation unit for a 9-bit computer that is specialized in several operations. The calculation unit receives 8-bit long input signals and generates output signals. Every 8-bit input consists of the 4-bit opcode (operation code, which decides the operation type that will be performed), a 2-bit number (first number) (that involves in calculations), another 2-bit number (second number) (that involves in calculations) as shown below.

$$\text{Input: } 00101110 \rightarrow 0010 - 11 - 10$$

$$\text{opcode: } 0010, \text{first number: } 11, \text{second number: } 10$$

Since the opcode is 4-bit long, the unit can perform at most 16 different operations but for this assignment the number of operations supported is 9. The following are the operations supported by the unit: number comparison, subtraction, addition, division, multiplication, modulo, bitwise and, bitwise or, and bitwise xor. The table below depicts the operations with their corresponding opcodes.

Operation Set and Input Parts								
Opcode				First Number		Second Number		Operation
4^{th} bit	3^{rd}bit	2^{nd}bit	1^{st} bit	2^{nd} bit	1^{st} bit	2^{nd} bit	1^{st} bit	
O3	O2	O1	O0	A1	A0	B1	B0	
0	0	0	0	a1	a0	b1	b0	A%B
0	0	0	1	a1	a0	b1	b0	A*B
0	0	1	0	a1	a0	b1	b0	A/B
0	0	1	1	a1	a0	b1	b0	$A + B$
0	1	0	0	a1	a0	b1	b0	$A - B$
0	1	0	1	a1	a0	b1	b0	Comparison
0	1	1	0	a1	a0	b1	b0	A $\wedge$ B
0	1	1	1	a1	a0	b1	b0	A $\oplus$ B
1	0	0	0	a1	a0	b1	b0	A $\vee$ B

$$\text{a1,a0,b1,b0} \in \{0, 1\}$$

$$\text{Opcode} = \text{O3O2O1O0, First Number} = \text{A} = \text{A1A0, Second Number} = \text{B} = \text{B1B0}$$

$$\wedge : \text{ bitwise AND}, \vee : \text{ bitwise OR}, \oplus : \text{ bitwise XOR}$$

3.1 Output

The output of the unit consists of 17 signals in total. 7 signals are special purpose outputs whereas the remaining 10 outputs each of which represents a numerical value between 0-9 indicate numerical results of the performed operations. The special purpose outputs are as follows: sign (S), carry (C), zero (Z), parity (P), equal (E), greater (G), and less (L). The sign output shows whether the subtraction operation has resulted in a negative number or not. Similarly, the carry output indicates whether the addition operation results in a value that needs more than 2 bits to be represented (carry bit). The parity output signifies whether there are an odd number of 1s in the binary representation of a result. Finally, in order to represent the comparison result of two numbers there are "greater than" (G), "less than" (L), and equal (E) outputs.

The numeric inputs of the unit (A and B) are unsigned 2-bit integers. When the unit issues an operation and yields a result, the outputs associated with the operation are activated (numerical value outputs, special purpose outputs) Output names and their detailed descriptions are as follows:

- Numerical values. They indicate the numerical value of a result:
  - L9: 9, L8: 8, L7 : 7, L6 : 6, L5 : 5, L4 : 4, L3 : 3, L2 : 2, L1 : 1, L0 : 0.
- Sign output:
  - S: 1, the result of the operation is negative.
  - S: 0, the result of the operation is positive (0 is included).
- Carry output:
  - C: 1, the addition operation yields an overflow/carry bit (the result is greater than 3).
  - C: 0, otherwise
- Zero output:
  - Z: 1, the result of the operation is 0.
  - Z: 0, otherwise
- Parity output:

    - P: 1, the number of 1s in the binary representation of the result of the operation is odd.
    - O: 0, otherwise.

- Comparison operation outputs:
    - E (Equal):
        * 1: if A = B
        * 0: otherwise
    - G (Greater):
        * 1: if A >B
        * 0: otherwise
    - L (Less):
        * 1: if A < B
        * 0: otherwise

Notes:

- A/B yields an integer value (integer-integer division). For instance, when A=5, B=2, the unit should activate the L2 output.

- For A/B and $A\%B$ operations, you may assume the input B will never be set to 0.

- The sign output is only associated with the subtraction operation (for other operations it should generate a value of 0). Similarly, the carry output is only used for the addition operation (for other operations it should generate a value of 0).

- The parity output for subtraction operation results should be determined from the absolute values of the results ($|A - B|$).

- The E, G, and L outputs can yield a value of 1 only when the comparison operation is selected via the opcode and their conditions are met. Otherwise, they should output a value of 0. In addition, no numeric output should be activated for the comparison operation (hence the zero output and the parity output should yield a value of 0).

Hints:

- We highly recommend you use multiplexers and decoders in your design.
- In order to reduce the number of components you may utilize tree-state buffers (controlled buffers).
- Please try not to leave any input pin of components unconnected (floating, since it may cause components not to work, it may be a problem for especially decoders and multiplexers). To remedy this problem, you may need to use constant values (Constant pin) in order to provide an input value to unused input pins.

For further clarification, here are some input-output examples for the computation unit, which include opcodes.

Opcode				A		B		Outputs																	
O3	O2	O1	O0	A1	A0	B1	B0	S	C	Z	P	E	G	L	L9	L8	L7	L6	L5	L4	L3	L2	L1	L0
0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	0	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	1	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	0	1	1	0	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	1	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	0	0	1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	0	0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	1	0	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	1	0	1	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0	1	1	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
0	1	1	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0
0	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	1	1	1	1	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

You are expected to implement the described computation unit as your second lab work. You need to use "input pins" for 8-bit inputs (4-bit opcode, 2-bit first number, 2-bit second number) and "output pins" for the outputs of the unit from the Toolbar at the top of Logisim. Please set their labels correctly using the following names. **Please, only set the "label" property of the "pin" objects (input and output), please do not just add a "label" object instead onto the Logisim canvas.**

4 Labelling Specifications

- You have to use **pins** for your inputs and outputs. Please only set the **label property** of the **pin** objects, please do not add a **label/text object** instead.

- Your input pins should be labeled as O3, O2, O1, O0, A1, A0, B1, B0.
- Your output pins should be labeled as S, C, Z, P, E, G, L, L9, L8, L7, L6, L5, L4, L3, L2, L1, L0.
- Label properties are case-sensitive. Note that all labels consist of an uppercase letter followed by a number. **Please be very careful in naming the labels correctly.**
- If you need to feed any input with a constant value, you can use a constant pin. This pin is under CENG232 gates. During evaluation, we will only set values for O3, O2, O1, O0, A1, A0, B1, B0.
- You will receive grade **penalty** unless labeling is done properly.

5 Deliverables

- Please submit your circuit design by naming e1234567.circ, which should be prepared in Logisim (CENG 232 version), by the specified deadline. Please do not forget to replace e1234567 with your 7-digit student ID.
- The evaluation of the submission will be a **black-box test**.
- You should use the CENG version of Logisim which is available on the ODTUClass course page. Circuits designed with other Logisim versions, other tools, or that are not named properly **will not be graded**!

6 Cheating Policy

All the lab work should be individual and there is a zero-tolerance policy for cheating. See the course website for further information about the cheating policy.

7 References

CENG Logisim Version.