Student Name: Tin T Vu V00830285

CMSC 311 Computer Organization
LAB 2: Digital Circuits Design by Multimedia Logic
Instructor: Dr. Cang Ye **Email:** cye@vcu.edu
TA: Lingqiu Jin **Email:** jinl@mymail.vcu.edu

Instructions:

Lab report for this session includes answers to all questions. Grading is based on your answers and lab participation.

Submit your report via blackboard. Please use "Lab2 report" as the subject of your email and name your file as Lab2_report_LastName. Please submit your report as a single file. An incomplete report will not be accepted. Please also attach the MMLogic files (Lab2_ LastName_n.lgi, where n is the number of the circuit, e.g. you should use Lab2_ LastName_1.lgi and Lab2_ LastName_2.lgi if you have two circuits.). A past due report will not be accepted.

Part One (Simple Examples):

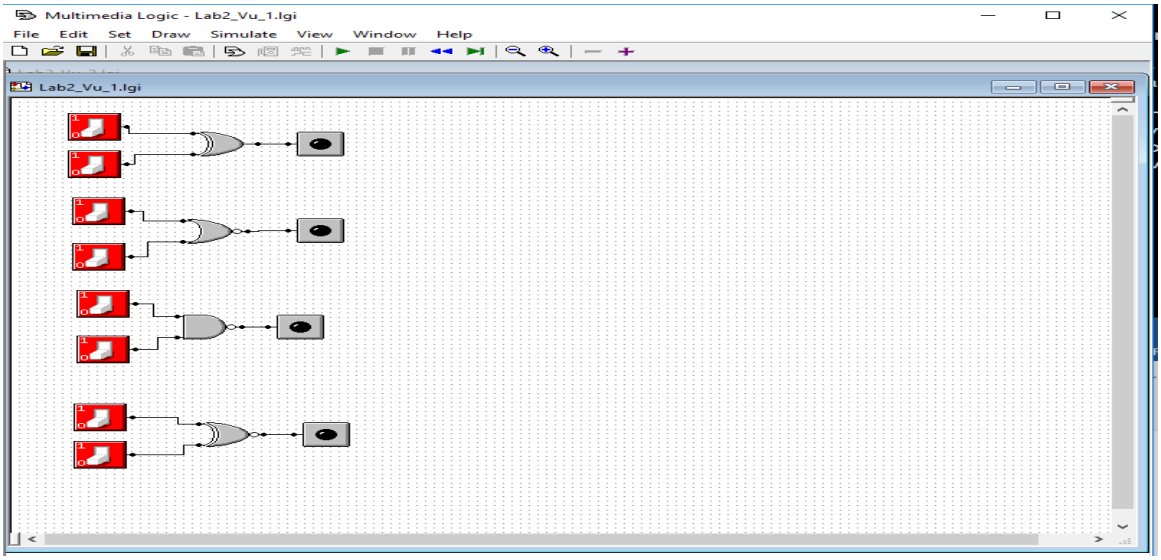
1. Produce the truth tables for the following gates using the MMLogic: XOR, NOR, NAND, XNOR. (3 points)

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

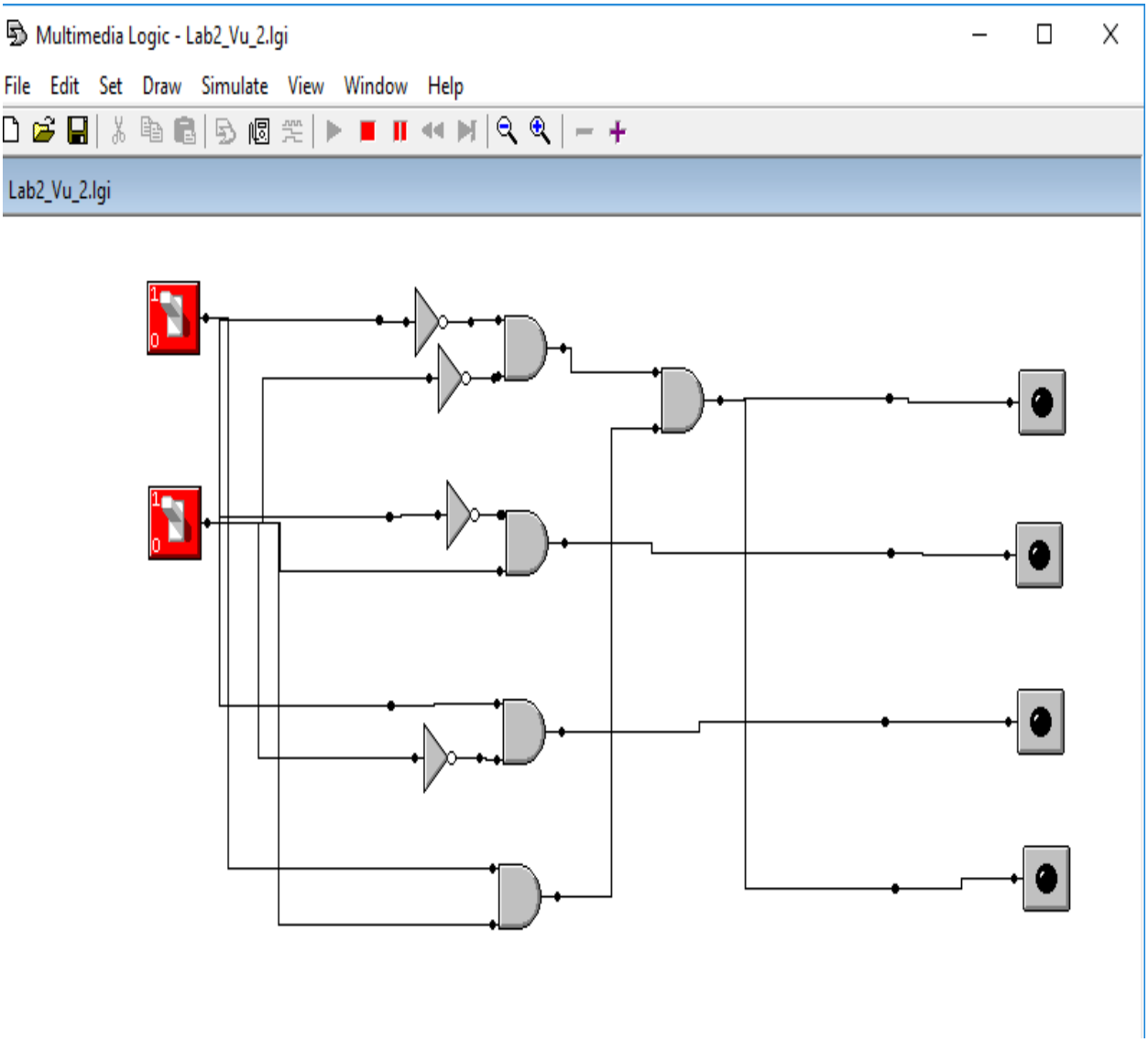
A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

A	B	XNOR
0	0	1
0	1	0
1	0	0
1	1	1



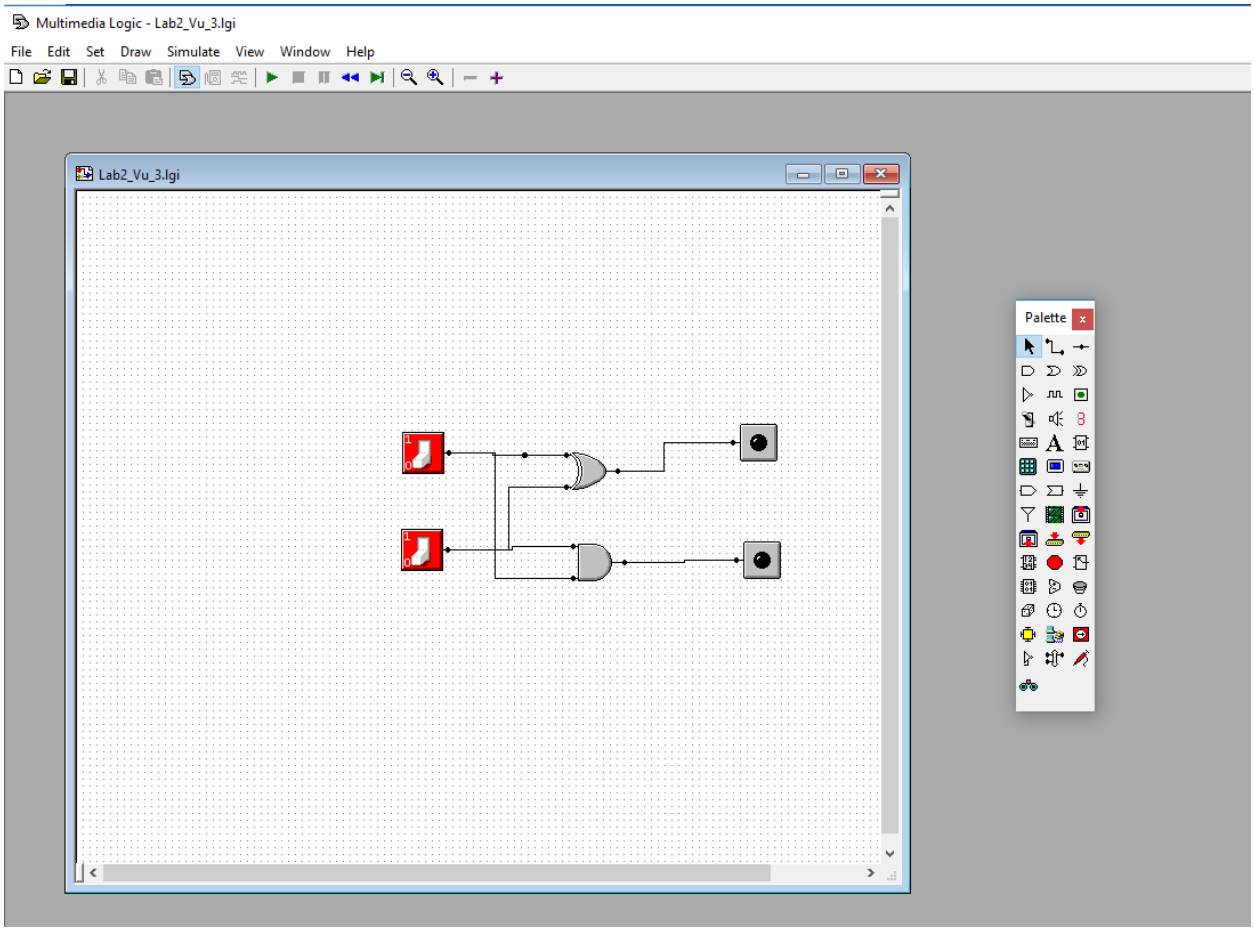
2. Use the decoder in Figure 3.11 (page 60 of the text book) to construct a circuit that performs the XOR gate function. You will need to connect appropriate outputs. (4 points)

Hint: The decoder has 4 outputs and each of them produces logic 1 for only one specific combination of input. For example, the first output is 1 because both A and B are 0. Write the truth table of the circuit and append the XOR truth table. You will find that you may achieve the XOR function by connecting appropriate outputs of the decoder with a gate. What gate should be used?

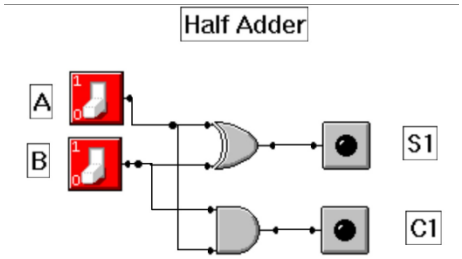


3. Half adder is a circuit that adds two one-bit binary numbers to produce the “sum” and “carry” outputs. Fill out the following truth table and then construct a circuit that realizes this truth table. (4 points)

A	B	Sum (s1)	Carry (c1)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



4. Draw and simulate the following circuit with the MMLogic and answer if it produces the same results for equivalent inputs (2 points)



Answer: The results are the same.

If your circuit is equivalent to the above one, what is the difference between them? If you are to design a half adder and the price of all gates were equal, which one do you prefer? Why? (2 point)

Answer:

- There was no different between them. I built the exact same haft adder circuit as the picture.
- If I was to design a half adder and the price of all gates were equal, I would prefer to go with the most inexpensive and easy to build one. In this case, I would prefer to go with AND, OR and NOT gates because it will cost less in production.

Part Two: (Arithmetic and Logic Unit)

Introduction

This lab is designed show you how a simple 8-bit ALU works and how the ALU is used in a data path (or Central Processor Unit). You will observe that:

1. The ALU behavior is controlled by the function selector S (3 bits in this case).
2. The data inputs include two 8-bit values, A and B, together with a single carry-in bit Cin.
3. The ALU output F is a 8-bit value.
4. Flags Cout, Z and V are single bit by-product information about the ALU computation just completed

The main activities to be performed are to configure an ALU by using inputs from 16-key keypads and outputs to LEDs and 7-segment displays, observe the outputs S, F, Cout, Z and V with Cin = 0 and 1, and record the results as you observe.

Background

Arithmetic Logic Unit (ALU) device

The MMLogic ALU is an 8-bit control unit shown schematically in Figure 1. It has two 8-bit operands labeled by A and B. The operations that it may perform are addition, subtraction, multiplication, division, shifting and comparing. They are selected by using a 3-bit selector S. An input bit Cin is used as the “carry-in” where appropriate. The 8-bit ALU output is labeled by F. Three 1-bit operational output flags, Cout, V and Z, are set when the result is “Carry-out”, Overflow and Zero, respectively. The bit labeled by N is not functional.

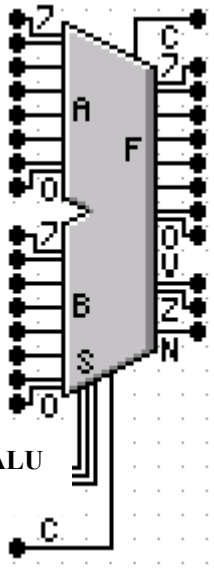


Figure 1 8-bit ALU

S2	S1	S0	Function
0	0	0	Add
0	0	1	Sub
0	1	0	Mult
0	1	1	Divide
1	0	0	Test A=B
1	0	1	Test A<B
1	1	0	Shift A Left
1	1	1	Shift A Right

Function	Output Flag Bit	Behavior (Common to all functions):
Indicate zero result	Z	IF the result is 0, THEN Z is 1. Otherwise Z is 0.
Indicate overflow	V	IF the result overflows or underflows, THEN V is 1. Otherwise V is 0
Indicate Negative	N	N is 0 (for future use)

Function	S	Output to F	Cout
ADD	**0**	A + (B + Cin)	IF the result overflows, THEN Cout is 1. Otherwise Cout is 0.
SUB	**1**	A - (B + Cin)	IF the result underflows, THEN Cout is 1. Otherwise Cout is 0.
MULT	**2**	A * B	Cout is 0.
DIV	**3**	A / B	Cout is 0.
Test for A=B	**4**	A = B (output is 0 for False and 1 for True)	Cout is 0.
Test for A<B	**5**	A < B (output is 0 for False and 1 for True)	Cout is 0.
Shift Left A by B	**6**	A << B (A left shifted by B) fill with Cin	Cout is the last (upper) bit shifted out of A.
Shift Right A by B	**7**	A >> B (A right shifted by B) fill with Cin	Cout is the last (lower) bit shifted out of A.

ALU Simulation using MMLogic

The first step to investigate the ALU operation is to configure an ALU with keypad controls and LED and 7-segment displays as shown in Figure 2. The top two keypads are connected to ALU input A and the values are displayed by two 7-segment displays in hexadecimal. (**15 points**)

The third and fourth keypads are connected to ALU input B and the values are displayed by two 7-segment displays in hexadecimal.

The fifth keypad is connected to Cin and the 3-bit selector input S. This keypad's value is displayed by a text box that indicates the ALU operation for the pressed function key. Figure 2 shows an addition scenario 0x12 + 0x48 + 1 = 0x5B, inputs A and B are 0x12 and 0x48, respectively and Cin=1. The result (F) is 0x5B. The selected operation is ADD with carry Cin.

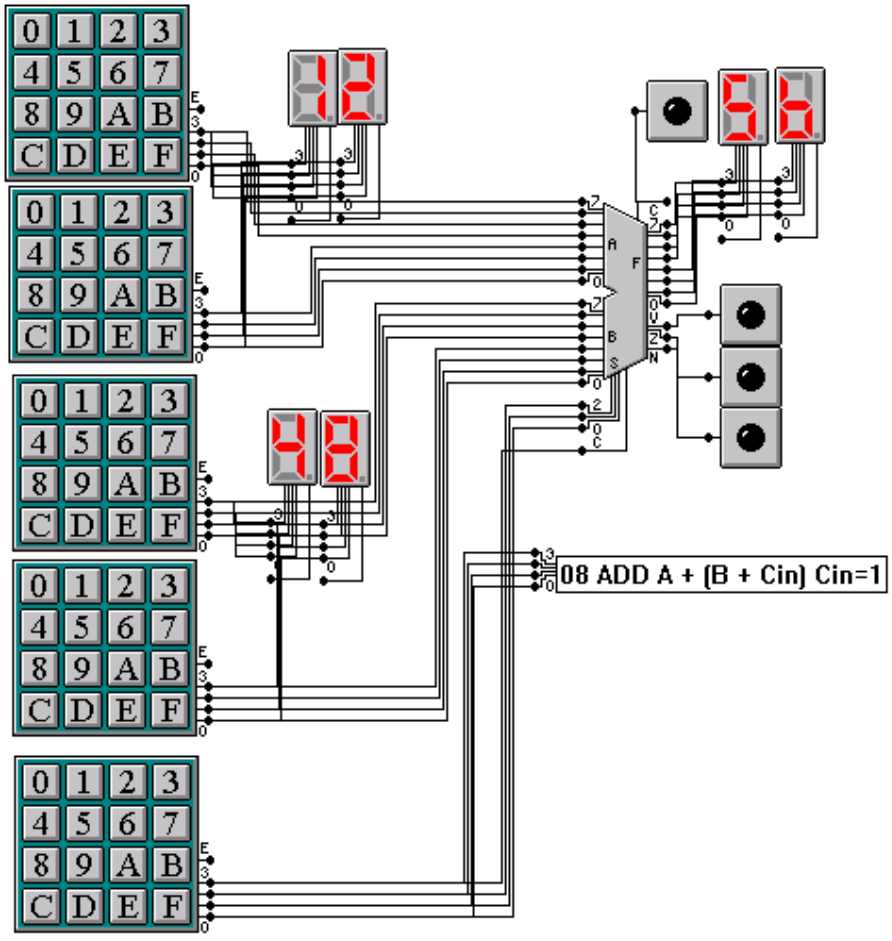


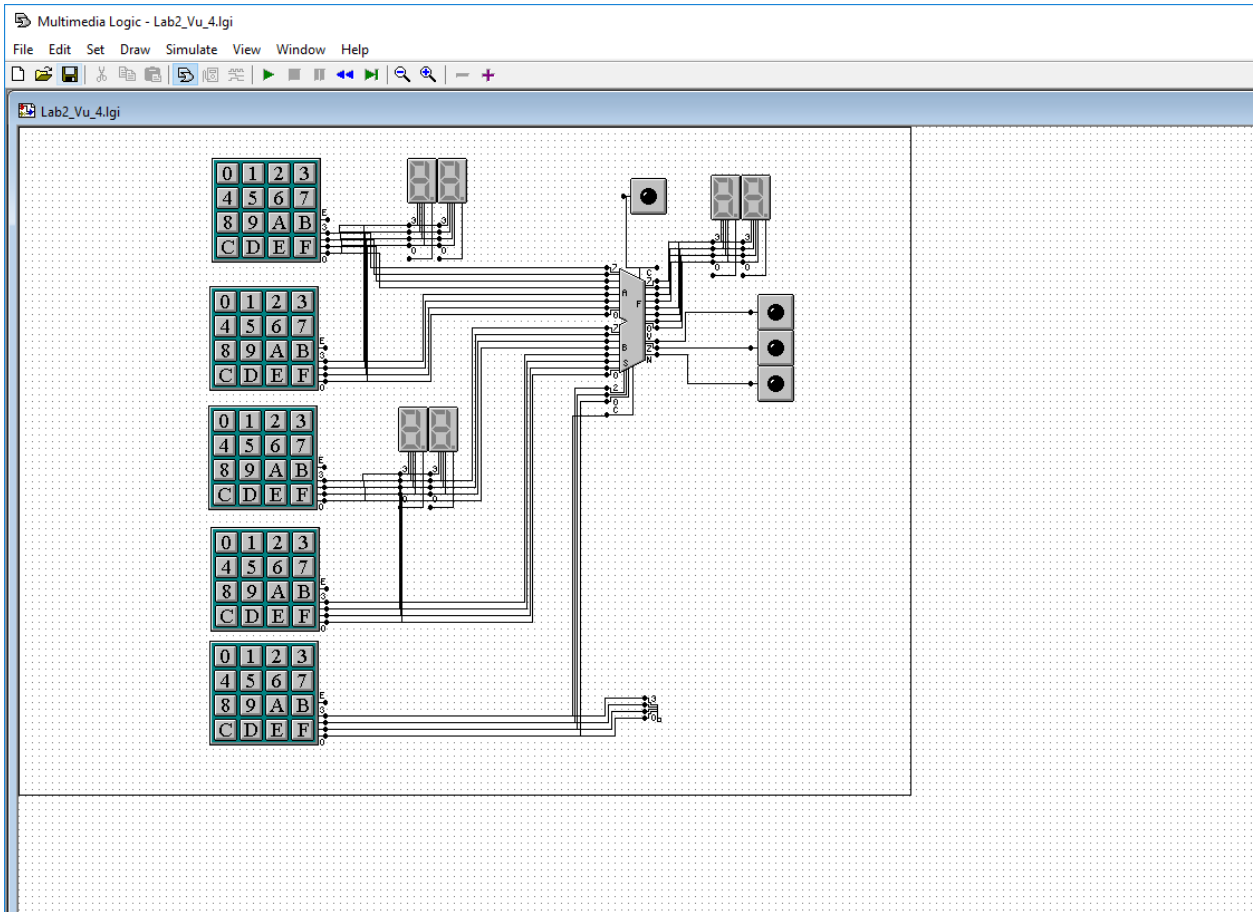
Figure 2 Instrumented ALU for simulation

Build a circuit as the same as Figure 2, run simulation, and check that all operations work. Record your results in the following table (with at least 16 rows): (15 points)

A	B	Cin	S	F	Cout	Z	V	OK
0x12	0x48	1	0x8	0x5b	0	0	0	yes

0x12	0x48	0	0x0	0x5a	0	0	0	yes
0x34	0x19	1	0x8	0x4e	0	0	0	yes
0x34	0x12	0	0x1	0x22	0	0	0	yes
0x34	0x15	0	0x1	0x1f	0	0	0	yes
0x6a	0x6a	0	0x1	0x00	0	1	0	yes
0x7	0x3b		0x2	0x9d	0	0	1	yes
0x07	0x10		0x2	0x70	0	0	0	yes
0x57	0x20		0x2	0xe0	0	0	1	yes
0x01	0x02		0x3	0x00	0	1	0	yes
0x08	0x02		0x3	0x04	0	0	0	yes
0x1a	0x10		0x3	0x0A	0	0	0	yes
0x10	0x10		0x4	0x01	0	0	0	yes
0x08	0x0A		0x4	0x00	0	1	0	yes
0xba	0xba		0x4	0x01	0	0	0	yes
0x01	0x55		0x5	0x01	0	0	0	yes
0x03	0x55		0x5	0x01	0	0	0	yes
0xa	0x02		0x5	0x00	0	1	0	yes
0x03	0x01		0x6	0x06	0	0	0	yes
0x09	0x01		0x6	0x12	0	0	0	yes
0x77	0x03		0x6	0xb8	1	0	0	yes
0x77	0x03		0x7	0x0e	1	0	0	Yes
0x45	0x02		0x7	0x11	0	0	0	yes

Table 1 ALU Simulation observations


Multimedia Logic - Lab2_Vu_4.lgi
File Edit Set Draw Simulate View Window Help
Lab2_Vu_4.lgi