ECE385 Experiment #3

Eric Meyers, Ryan Helsdingen

Section ABG; TAs: Ben Delay, Shuo Liu

February 10th, 2016

emeyer7, helsdin2

I. Introduction

THE purpose of this lab is to design and construct a four-bit serial logic processor that performs a total of eight logical operations in a bit-wise fashion. There are two four-bit words stored in two shift registers. WRITE MORE SHIT HERE

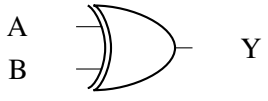
II. Pre-Lab

Part A) Describe the simplest (two-input one-output) circuit that can optionally invert a signal (i.e., one input determines if the output is equal to the other input or equal to the other input inverted). Sketch your circuit.

Answer: The logic needed is shown in the following truth table:

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Taking B to be the ”select” line in this example, when the input (A) is driven high and our select line (B) is low, the output will be high. The reverse case is true when the input (A) is low and the select line (B) is high, the output will also be high. Therefore, depending on the select line, the input will be inverted. This can be implemented with a single XOR gate as shown below.



Part B) Explain how a modular design such as that presented above improves testability and cuts down development time. Propose an approach that could be used to troubleshoot the modular circuit above if it appeared to be completing the computation cycle correctly but was not giving the correct output. (Be specific.)

Answer: A modular design allows the operator to unit-test each individual module. Once each module is successfully tested, the entire system can be integrated into a whole-system and tested. A modular design ideally will cut down the debugging time needed on the system as a whole.

III. Description of Circuit

The logic processor was designed to contain two storage registers, and allow the operator to specify the data loaded into each register (A and B), as well as the operation performed between the two register, where the output is stored and The team designed a total of four modular sub-systems in this lab. The following sub-systems were designed and constructed:

1) Data Loader Unit
2) Computation Unit
3) Function-Select and Output Router Unit
4) Control Logic Finite State Machine (FSM) Unit

Each sub-system was tested individually and system integration tests were performed at the end upon successfully unit-testing each component. This way the team could eliminate errors and debug with efficiency.

A total of eight operations can be performed on this unit. AND, OR, XOR, NAND, NOR, and NXOR are the six primary logical operations that can be performed, along with the last two operations simply being loading the specified registers with all 1s or 0s.

The Data Loader consists of a total of six switches. Two switches for loading either register A or B, and

four switches to specify the contents to store in the particular register.

The Computation Unit performed the selected operation specified through the three function-select switches. The output of the operation is funnelled into the proper register as given by the operator.

Finally, the Control Logic/Finite State Machine Unit is a sequential circuit that essentially determines the correct number of cycles to perform a full logical operation. It relies on one input (execute switch) and has one output that determines if the registers should be shifting right, doing nothing, or loading.

In the end, the functionality of the circuit is as follows: when a user selects a four-bit word and loads it into the register of their choosing, they must then select the logical operation they would like to perform, select where they would like the output of this logical operation to go via the routing selection, and flip the execute switch. This will begin shifting the outputs of the shift register and perform bitwise logical operations of the requestion function. The machine will halt after exactly four operations and the register requested through the routing switches will contain the output of the logical operation.

IV. State Diagram

The FSM created in this lab is a Mealy State Machine and this was chosen due to the simplicity of the relationship between the input and output on the state diagram.

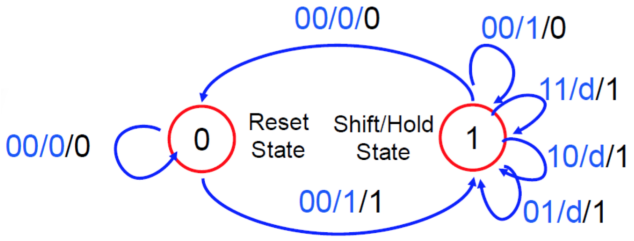


Fig. 1: Mealy FSM

V. Design

The first component designed in this lab was the FSM Sequential Logic Circuit. This is because the output of this state machine controls the actions on the shift register (i.e. this output controls the ”control bits”, S1 and S0, which determines whether the registers are shifting right, parallel loading, or doing nothing). This is important because it determines whether we are in a load/execute/do nothing state.

The truth table was given from the documentation in the Experiment 3 PDF. This is shown below in figure:.

Exec. Switch (‘E’)	Q	C1	C0	Reg. Shift (‘S’)	Q’	C1’	C0’
0	0	0	0	0	0	0	0
0	0	0	1	d	d	d	d
0	0	1	0	d	d	d	d
0	0	1	1	d	d	d	d
0	1	0	0	0	0	0	0
0	1	0	1	1	1	1	0
0	1	1	0	1	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	d	d	d	d
1	0	1	0	d	d	d	d
1	0	1	1	d	d	d	d
1	1	0	0	0	1	0	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	0

Fig. 2: FSM Truth Table

Next, given this truth table, the team constructed a K-Map for each of the output bits - S, Q , C1 and C0. Since the only output is ”S” and the remaining bits are only internal to the state machine, the

The first component that was designed was the Loader Unit. As stated before, this requires the operator to input the four bits of data via the DIN switches, and allows them to either load it into the A register or the B register. If either LoadA or LoadB are on, the Only when LoadA OR LoadB is on, must the contents specified by DIN have the ability to be parallel loaded into the shift register.

Next, the FSM Sequential Logic Circuit was designed

Exec. Switch (‘E’)	Q	C1	C0	Reg. Shift (‘S’)	Q’	C1’	C0’
0	0	0	0	0	0	0	0
0	0	0	1	d	d	d	d
0	0	1	0	d	d	d	d
0	0	1	1	d	d	d	d
0	1	0	0	0	0	0	0
0	1	0	1	1	1	1	0
0	1	1	0	1	1	1	1
0	1	1	1	1	1	0	0
1	0	0	0	1	1	0	1
1	0	0	1	d	d	d	d
1	0	1	0	d	d	d	d
1	0	1	1	d	d	d	d
1	1	0	0	0	1	0	0
1	1	0	1	1	1	1	0
1	1	1	0	1	1	1	1
1	1	1	1	1	1	0	0

Fig. 3: FSM Truth Table

S	LA	LB	S1A	S0A	S1B	S0B
0	0	0	0	0	0	0
0	0	1	0	0	1	1
0	1	0	1	1	0	0
0	1	1	1	1	1	1
1	0	0	0	1	0	1
1	0	1	0	0	1	1
1	1	0	1	1	0	0
1	1	1	1	1	1	1

VI. Block Diagram

Please refer to Figure ____ in ”Section XI: Figures” of this document to view the Block Diagram created in this lab.

VII. Circuit/Logic Diagrams

As stated before, to allow for easier construction and debugging, this system was broken down into modular components. All circuit diagrams are shown in ”Section XI: Figures” of this document.

The FSM Control Logic is shown in Figure ___. The Loader/Computation/Router Unit are shown in Figure ___.

VIII. Component Layout Sheet

Please refer to Figure ____ in ”Section XI: Figures” of this document to view the Component Layout Sheet created in this lab.

IX. Documentation from Experiment

RYAN SECTION

X. Conclusion

RYAN SECTION

XI. Figures