**ECE 385**

Fall 2020

Experiment 3

**A Logic Processor**

**Zhicong Fan/Xin Jin**

**AB2/Online**

**Yucheng Liang**

**Introduction:**

In this lab, we designed a bit serial logic operation processor. There are 9 inputs in this circuit, data input (D0-D3), control inputs (F0-F2, R0-R1). With these 9 inputs, the circuit could calculate 8 different functions, and store in 4 different ways.

**Prelab Questions:**

***Q: 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.***

A: We could achieve this by using an inverter and a 2-to-1 MUX. Here is the design:A close up of a screen

Description automatically generated

***Q: Explain how a modular design such as that presented above improves testability and cuts down development time.***

A: A modular design divides the entire circuit into several relative small parts. So that we could test units one by one. For example, in this lab, if the output from the Computation unit is correct, but the final output is wrong, then the error must be in the routing unit. If we design a modular design, we work on one part at a time which cuts down the development time comparing to implementing a random chip in a entire unit.

**Operation of the logic processor:**

**Load Values Into Register A and Register B:**

Let’s say we would store the value into A first and then B. Then, first, we need to flip the switches for the value we want to store in Register A. After that, the data would be loaded into Register A. Then we just need to do the same operation again except that instead of flipping Load A, we would flip Load B.

**Initiate Computation and Routing Unit:**

After loading data into Register A and B, we need to specify what function we need representing by flipping the switches from F0-F2. For example if we want an XOR operation, we would then flip F1 to 1. Then, to specify the routing, we need to flip R0-R1. For example if we want to store the data in B, we would then flip R0 to 1.

**Written Description and Diagrams:**

**Description:**

For the Register Unit:

D0-D3 are directly wired to the parallel inputs of the Shift Register to load data into the register. Then the output of the routing unit are wired in the Shift Right Serial Input pin to load the data routed by the routing unit. To control the mode of the register, S0 S1 calculated by the control unit are wired to pin S0 and pin S1 respectively.

For the Computation Unit:

In our circuit, NAND, NOR, XOR and several inverters are used to implement the 8 functions. From 00-11 in the 4-to-1 MUX controlled by F0-F1, we have: NAND, NOR, XNOR(XOR and a inverter) and a wire connected to the ground (0). F2 goes through an inverter and we use an XOR to control the final output of the Computation Unit. For example, if we want an XNOR result of a 0 and a 1, the 4-to-1 MUX would output 0. Then this 0 signal XOR with the output of F2 inverter 0, would give us the final output 0 which is correct.

For the Routing Unit:

R0 and R1 are used in the routing unit. From 00-01, we have: A B, A F, F B, B A, which means whether we should store the result F, or the same value into either Register A and Register B. For example, if we choose, 01(A F)mode, Register A would still store the original value while the Register B would store the result computed by the computation unit.

For the Control Unit:

Execute, Count0, Count 1 are the fundamental inputs of our logics. The counter is control by the output of a series of logic gates. Then the output of the counter Counter0 and Counter1 are combined with execute to generate the input of the flip flop, in which we would store Q the current state of the Mealy Machine. And this state Q is used with Execute, Count0 and Count1 again to compute S which acts as the control pin to control the mode of the Shift Register.

**Block Diagram: 记得改F2 和你的4-TO-1 MUX 11的input， 从1改成0。**

**State Machine Diagram:**

In this lab, we used Mealy machine instead of Moore Machine since it would contain less states.

Here is the diagram:

A close up of a logo

Description automatically generated

**Design Steps:**

**Breadboard View:**

**Bugs Encountered:**

1. We mistakenly apply the result of an XOR gate. The output of 00 is not 1 but a 0. After we change the logic, the computation unit gives us the correct output.
2. The control pin used to control the counter is hard to find. Instead of the output of Execute, Counter0 and Counter1 or the output of flip flop Q, the actual control pin is S the output of Q, execute, Counter0 and Counter1. We drew a truth table to help us decide the logic gates of S.
3. The inputs sequence of a 2-to-1 mux was reversed, instead of LOADA/B and then Ground, I connected the pin as Ground and then LOADA/B.

**Conclusion:**