**ECE 385**

Fall 2020

Experiment 2

**Data Storage**

**Zhicong Fan/Xin Jin**

**AB2/Online**

**Yucheng Liang**

**Introduction:**

In this lab, we designed a 2 bits 4 words shift register. Within the circuit, we implemented three operations: FETCH,STORE and LDSBR. We use switches to send signals and use LEDs to display the outputs.

**Memory Circuit Operation:**

* Addressing:

When LDSBR is high, data we want to store in DIN0 and DIN1 would be loaded into SBR0 and SBR1

* Write:

When we want to use “write” function, we have to first flip Din0 and Din1 for the data we want to write into the shift register. After that, we need to specify the address of the data, so we need to flip SAR0 and SAR1. Then a LDSBR function is required to put our data into SBR first. After that, we just need to flip STORE and then wait for the comparator to give us a signal if the counter matches with the address we want. It would take at most 4 cycles to load the data into shift register.

* Read:

SAR0 and SAR1 would need to be firstly flipped if we want to read data in a specific address. And then we should flip FETCH to tell the circuit we want to read something. Then when the counter matches with SAR0 and SAR1, the data would be loaded into SBR which is connected to the LEDs. If no data is in the shift register, 00(no light) would appear on LED.

**Description and Block Diagram:**

**High-level Description:**

A big switch is rudimentary if we want to control the circuit. From up to down, the functions of switches we assigned are: 1. SAR0, 2. SAR1, 3. DIN0, 4. DIN1, 5. FETCH, 6.STORE, 7.LDSBR. So when a operation takes place, the relative switches would be flipped first. Then a comparator and a counter is needed to see whether the current address in the shift register matches with the address we want. If the addresses are matched, the comparator would output 1 in the Control Unit. Besides these, we need two 3-to-1 muxes(We used 4-to-1 instead), two 2-to-1 muxes, two flip-flops, two shift registers, six LEDs and a few NAND or NOT gates are used in all.

The comparator takes SAR0, SAR1, C0,C1 as inputs and output 1 if SAR0 is the same as C0 as well as SAR1 and C1. Then the output combines either with STORE or with FETCH to control the 2-to-1 muxes that manage Shift Register 0 or Shift Register 1 to read either from Shift Register itself or from the SBR. Then the output of the Shift Registers are used as inputs as well as DIN and SBR in two 3-to-1 muxes. LDSBR and the coalition of FETCH and Comparator Output are used to control the 3-to-1 muxes. The output of the 3-to-1 muxes are the inputs of SBR.

A screenshot of a cell phone

Description automatically generated

**Control Unit:**

**Description:**

In the Control Logic we have a counter, a comparator, two NAND gates and two NOT gates.

The comparator takes SAR0, SAR1, C0,C1(C0, C1 are the outputs of the counter) as inputs and output 1 if SAR0 is the same as C0 as well as SAR1 and C1. Then the output combines with STORE and then go through a NAND gate as well as a NOT gate to control the 2-to-1 muxes that manage Shift Register 0 or Shift Register 1 to read either from Shift Register itself or from the SBR. There are two branches of the Comparator Output, the other one connects with FETCH signal and also the combined signal goes through a NAND gate successively a NOT gate to control the 3-to-1 muxes. But since the muxes have 4 input values, LDSBR is also used to control the muxes.

**Graph**

**Design steps:**

**a.**

**b.**

**c.**

**d.**

**Component Layout Sheet:**

**Bugs encountered:**

**Prelab Questions:**

***Q: The clock must run continuously – do not gate the clock (this is***

***bad practice in digital design, why?)***

A: If we gate the clock, the delay is prolonged which would cause a static hazard that we explored in the last lab.

***Q: Only the clock input needs to be de-bounced in order toto strep through your circuit (why?).***

A:

**Postlab Questions:**

***Q: What are the performance implications of your shift register memory as compared to a standard SRAM of the same size?***

A: An SRAM is static and thus we are able to directly read or store data from or in a SRAM and not to wait for a counter to match with the address we want to store at. The reason why a counter is needed when we are using shift register is that the voltage in the shift register would continuously drop so that the shift register has to loop around to make sure the voltage is high so that it could store and data and not to lose it. That’s why it is undetermined in the states at each time so that we need a counter and a comparator to keep track of it.

***Q: What are the implications of the different counters and shift register chips, what was your reasoning in choosing the parts you did?***

A: A ripple counter and a synchronous counter is mentioned during the lecture. We chose to use synchronous counter because it has a perplexed feature that could preclude glitches in its outputs, which means it could provide impervious monotonous signal alongside the clock which a ripple counter could not. So we chose to use a synchronous counter. For shift register, the shift register has the left serial input that we needed, so that we chose the 74194 chip.

**Conclusion:**