University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Science

CS150 J. Warzynek and N. Weaver
Fall 2000 Revisions by R. Fearing, X. Zhang, B. Choi, and N. Zhou

CS150 Project Check Point #1
Wire-Wrap, SRAM, and FIFO design.

1 Objectives

For this lab, you will:

1. Learn to use Wire-Wrap.
2. Gain more practice in digital design
3. Learn some strategies to test circuits
4. Wire an SRAM to the Xilinx on your Design Demonstration Board and test it.

2 Prelab

Fllowing the directions in Section 3, use Wire-Wrap to connect a W24512AK, an 64K x 8 SRAM to the Xilinx chip on your Design Demonstration Board. Use the pinouts in Figure 2 and Wrap-ID from Page 5.

Design the FIFO and SRAM access logic for the circuit described in Section 5

3 Wire-Wrap

Wire-Wrap is a prototype construction technique where 30-gauge wire (``Wire-Wrap wire'') is twisted around a square post to make aconnection. These connections are physically strong, reliable, electrically sound, and quickly made.

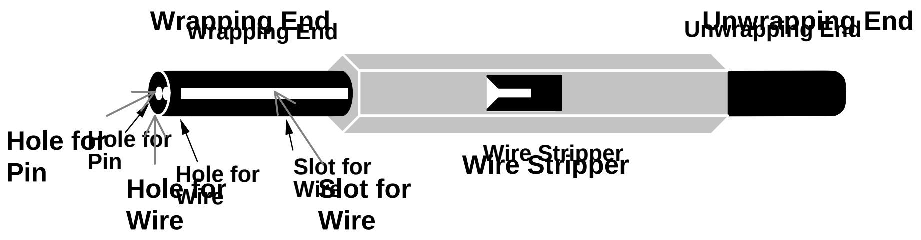
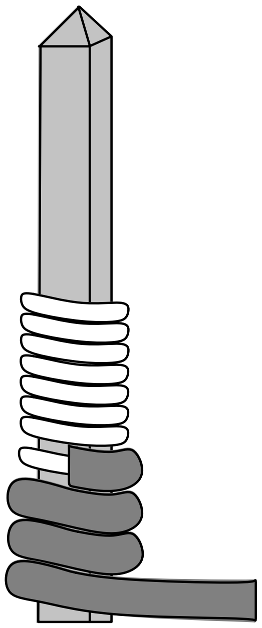


Figure1 Wire-Wrap tool (not to scale)

A Wire-Wrap tool consists of three parts:

- A central hexagonal rod with a wire stripper in the center (the little black notch).
- The wrapping end: a black circular rod with a hole in the center and an off-center hole leading to a side slot.
- The unwrapping end: a shorter circular rod with a hole in the center.

To make a Wire-Wrap connection



1. Plan the route of your wire. Leave little slack, perhaps a half inch. Wires should lie flat against the board.

2. Cut the wire~2.5 inches longer than the route you chose.

3. Strip an inch of insulation off each end of the wire using the Wire-Wrap tool's stripper: thread the wire through the circular hole in the hexagonal handle, push it into the stripper's notch, and pull.

4. Insert a stripped end of the wire into the off-center hole in the end of the wrapping end of the tool. The wire should be visible in the slot on the side, although it should not hang out of this slot.

5. Put the wrapping end of the tool on the pin to be connected. The pin should fit easily in the center hole.

6. Gently, without pushing down or lifting up, twist the tool clockwise about twelve revolutions. This should wrap a few turns of insulated wire around the bottom, followed by the stripped wire. There should be a single layer of wire, with no spaces between turns.

If you make a mistake connecting a wire, remove it completely and start again. Place the unwrapping end of the tool on the pin and turn counter-clockwise. The wire should slide off. An unwrapped wire should not be re-wrapped.

Use differently colored wire to group signals. For example, make the data lines in this lab green, the address lines orange, and the control signals red.

It is easy to forget that when you Wire-Wrap, the chip's pins appear as the *mirror image* of the usual pinout diagram. The best solution is to use a Wrap-ID, something that fits over the Wire-Wrap pins with the (correctly mirrored) pinout written on it. Such a Wrap-ID is printed on Page 5---cut it out, and carefully push it down on the pins before you start wiring your W24512AK.

A few hints on wire wrapping:

- Always use Wrap-IDs. We will always provide you with Wrap-IDs, but if you ever need to wire wrap something and you don't have one, make one.

- Be neat. Although it is difficult to produce perfectly pristine connections, you should keep the wires as flat and as neet as possible. Messy wirewrap is much harder to debug, and is less reliable.

- Check for wire fragments. It is quite possible to have a little piece of wire fall down and short out two pins. Look carefully at the wires to make sure this does not happen.

- Inspect each connection after you make it, to insure it is to the right place and that the connection is solid. If you have a digital multimeter(If you don't, inexpensive ones are available at Radio Shack. You probably should buy one, they are good things to have.), use it to check each connection after you make it.

4 SRAMs

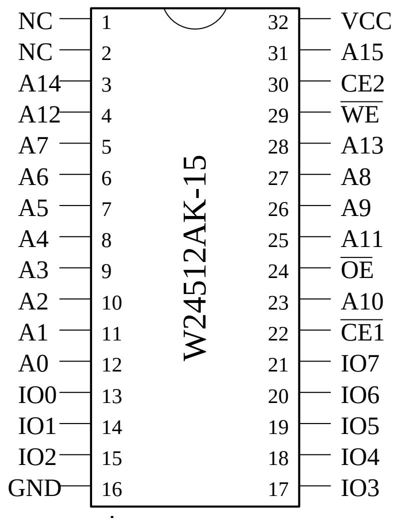
An SRAM (Static RAM) is a standard form of digital storage. It is generally faster, simpler, and more expensive then DRAM (Dynamic RAM).SRAMs are nice when one needs to store a moderate but not excessive amount of data, and needs to retrieve it quickly.

The W24512AK you use is a high speed SRAM. It takes only 15nanoseconds for an address to produce output data, and comes in a standard DIP package.

Their electrical interface is simple:

- eight data pins, for input and output
- address input pins. The~W24512AK, 2^{16} = 65536-byte part, has 16 address pins.
- an active-low and an active high chip enable inputs ($\overline{CE1}$ and CE2), and an active-low output enable input ($\overline{OE}$)
- an active-low write enable ($\overline{WE}$).
- power (V_{CC}) and ground (GND) pins

For this project $\overline{CE1}$ should be tied ground physically since we only need one chip enable signal. CE2 should be high to enable the chip for read and write operations. To read the chip, lower the $\overline{OE}$ input, the chip will respond by driving the data I/O to the content of the memory at the location specified by the address input, change the address and the output will change. To write to the chip, lower the $\overline{WE}$ input the data at the address will change to reflect the data I/O.



D0	77	A3	7
D1	78	A4	6
D2	60	A5	5
D3	59	A6	4
D4	58	A7	3
D5	57	A8	72
D6	66	A9	71
D7	65	A10	79
$\overline{WE}$	70	A11	80
$\overline{OE}$	67	A12	81
A0	10	A13	82
A1	9	A14	83
A2	8	A15	84

Figure2 W24512AK Pinout (top view) and pins on the Xilinx

Additional wirings: (CE2 14), ($\overline{CE1}$ GND), (NC GND)

5 testing the SRAM and the FIFO

There are generally two ways to test a memory, by producing specific patterns, or by a brute force approach. A set of specific patterns would test each wire a line at a time, to insure the output is correct. Brute force simply checks all possible values. For this lab, we will use brute force approach, as it is easier to implement. We will provide 2 bit files U:/cs150/Chk_pt1/SRAM test`(slow).bit and a …(fast).bit which tests the SRAM, but you should also build your own version.

The slow one runs at 128KHz and the fast one runs at 8MHz. Both circuits test writing and reading from all addresses of memory from 0000 to 7FFF. (note this does not test the A15 so you have to take extra care not to wire it wrong (and yes I did that for a reason)). The upper bits of the address is displayed on the number LEDs. If you wire-wrapped correctly, the circuit will continuously run through the memory and never stop. Other wise it will stop and the top 2 bits on the bar LED will light up.

Building you own circuit to test the SRAM:
You will need the following parts in your project so please design them well.

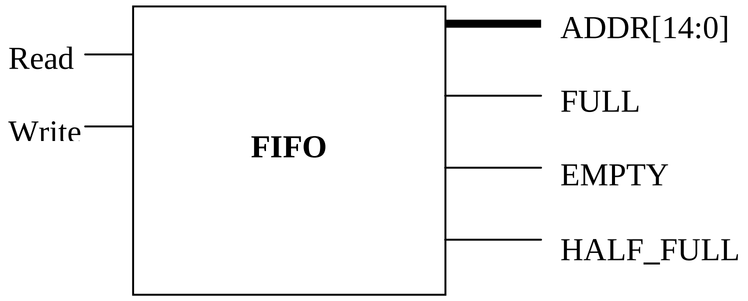
FIFO:

First we need something to generate the address for the SRAM. In this case we are going to use a FIFO (First In First Out) model. Abstractly a FIFO is a buffer that you can read and write to; elements are read out of it in the order that they were put it (think opposite of a stack).

In hardware this is implemented with 2 counters holding the “head” and “tail” address of the FIFO. For our purpose the head is the address for the next write and the tail is the address for the next element to be read. Each time you finish a read or write you simply increment the appropriate counter. In the final project you will need 2 FIFOs each taking ½ of the memory space so we will make a FIFO that only uses 15bits of address for this lab.

You will need to choose which address gets routed to the SRAM’s address input based on weather you are reading or writing to the FIFO. To do this a MUX or a Tri-state buffer could be used. (I recommend Tri-state, since we haven’t covered it in class ask your TA about it)

You will also need to generate signals (FULL, EMPTY, HALF_FULL) for your FIFO to guarantee that you never read when it is empty or write when it is full. To be clear the FIFO is empty when head_counter = tail_counter and it is full when head_counter – tail_counter = -1 or when the head_counter is at 7FFF and the tail_counter is at 0. I will leave it to you to design the FIFO. The HALF_FULL signal is optional for checkpoint 1.

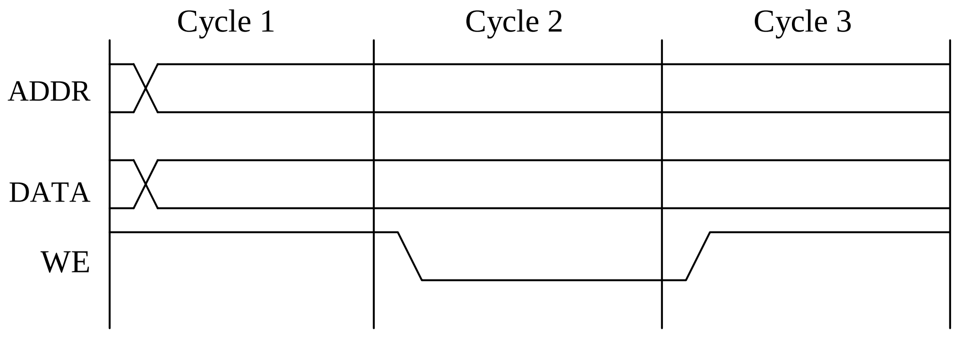


FYI. My reference design used 2 Counters, 30 Tri-states, one adder, one comparator, and quite a few logic gates. However it is not the optimal solution my any means. There are simpler and faster designs.

SRAM Controls:

You will also need to put together some FSMish circuit to generate the OE and WE for the SRAM. For this lab just tie CE to VCC in you schematic (not physically on the board!). You will also have to tie A15 to GND or V_{CC} as the FIFO will not generate it. Reading on the SRAM is simple as explained just assert OE and read the data line.

Writing however is a little bit tricky. The trickiness arises from the fact that a SRAM is not a clocked component (it’s Asynchronous). This means that when you assert the WE signal it is absolutely critical that the address and data line stay stable and not just on the clock edges. If the address line wiggles just a little after the WE was asserted or before it is de-asserted (possibly due to skews caused by wire length difference) you could get junk written to parts of the memory that you did not intend to write to. In practice we usually do 3 cycle writes.



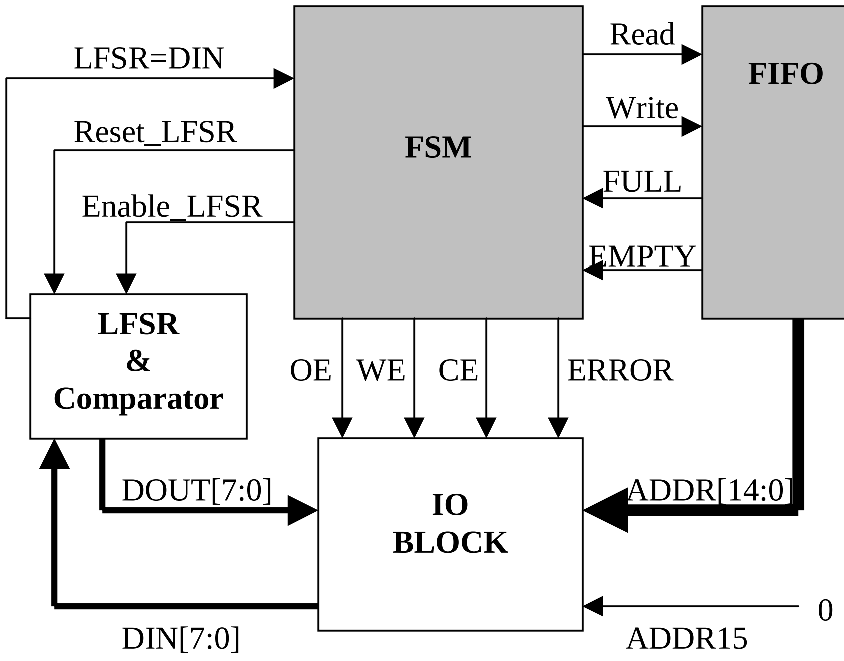
By Dividing the write to 3 cycles you guarantee that neither the address nor the dada changes when WE is asserted.

FYI. My reference design takes 4 cycles write and 2 for read. Again if you are clever you should be able to get 1 cycle read with 2 cycles for write.

Putting it all together:

Again you will have provided blocks to make your lives easier this time. (You're welcome ☺) As always the library is located in U:\cs150\chk_pt1\. You are going to design a FSM that writes data into the FIFO until it is FULL then read from the FIFO till it is empty while making sure that what you read is the same as what you wrote. If the data you get back is not the same then the FSM will stop and assert an error signal until it is reset. If no error is encountered the FSM will just loop back and forth between write and reed forever. We provide the IO block and one more block containing a LFSR and a comparator to generate the data to write into to SRAM and for verifying.

Here is a reference Block diagram. Design the shaded parts. (note I have inserted inverters in the proper lines in IO BLOCK so that OE, WE and CE are all active high for you)



Quick summary:

a) Wire-wrap everything. Test wire wrap with provided bit file.
b) Design FIFO. 15 address lines, 2 counters for head and tail, generate FULL EMPTY signal.
c) Design FSM to write to SRAM till FIFO is FULL, then read from FIFO and verify what you read is correct is correct. Assert error and stop if there is error else repeat. You FSM should also generate correct control signal for SRAM read and writes. (all OE, CS, and WE going into the IO BLOCK is *active-high*)

Advice:

Start this early. This design took me about 15hr to do. You don't have to do as much as I did (assigning pins on Xilinx, design lab from scratch, wrong SRAM parts) but it is still complicated enough that most of you will not be able to design and debug it in 3hrs of lab time, and it only gets worse from now on so you don't want to get left behind.

GOOD LUCK! and please for give me for any typos, it's late.

Name________________________________ Name________________________________
Lab Section (Check one)
M: ☒ AM T: ☒ AM W: ☒AM Th:
☒ PM ☒ PM ☒PM ☒PM

6 Checkoffs

1. At the START of the lab session, show your TA your wiring which you did before lab.
   SRAM wired to Xilinx (neatly for full credit)

   TA: _____________(20%)

2. Show your TA your FSM and FIFO circuit, which you entered before lab. At the START of the lab session. Show your TA simulated read and write cycle, showing ADDR, DATA, WE, OE, LFSR_EN and ERROR.

   TA: _____________(25%)

3. Verify the SRAM functionality with provided bit file.
   TA: _____________(20%)

4. Use your design to verify SRAM functionality **TA:** _____________(25%)

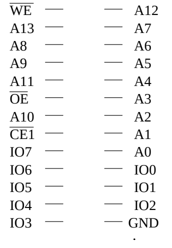
5. Full. Understanding of SRAM **TA:** _____________(10%)

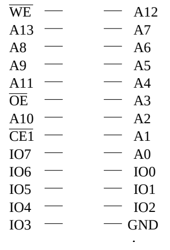
6. **Extra Credit:** Being more clever than Norm with your design. (impossible! J/K).
   Must fully explain your design for extra credit.
   a. Better FIFO with less hardware (pretty easy) 5%
   b. Sustained 1 cycle SRAM read (not that hard) 5%
   c. Sustained 2 cycle SRAM write (hmm... you need to be pretty clever for this one) 10%

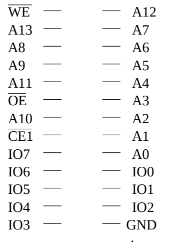
   TA: ___________(___%)(20%MAX)

7. Finished on time (phew…) **TA:** ___________(x100%)(full credit)

8. One week late (Doh!) **TA:** ___________(x50%)


W24512AK-15
VCC
A15
CE2
$\overline{WE}$
A13
A8
A9
A11
$\overline{OE}$
A10
$\overline{CE1}$
IO7
IO6
IO5
IO4
IO3
NC
NC
A14
A12
A7
A6
A5
A4
A3
A2
A1
A0
IO0
IO1
IO2
GND


W24512AK-15
VCC
A15
CE2
$\overline{WE}$
A13
A8
A9
A11
$\overline{OE}$
A10
$\overline{CE1}$
IO7
IO6
IO5
IO4
IO3
NC
NC
A14
A12
A7
A6
A5
A4
A3
A2
A1
A0
IO0
IO1
IO2
GND


W24512AK-15
VCC
A15
CE2
$\overline{WE}$
A13
A8
A9
A11
$\overline{OE}$
A10
$\overline{CE1}$
IO7
IO6
IO5
IO4
IO3
NC
NC
A14
A12
A7
A6
A5
A4
A3
A2
A1
A0
IO0
IO1
IO2
GND