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

CS 150 J. Wawrzynek and N. Weaver
Fall 1998 Later revisions by R. Fearing, X. Zhang, and B. Choi

Lab 7
Wire-Wrap and SRAMS

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

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

Design the control logic for the circuit described in Section 4.1.

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 a connection. These connections are physically strong, reliable, electrically sound, and quickly made.

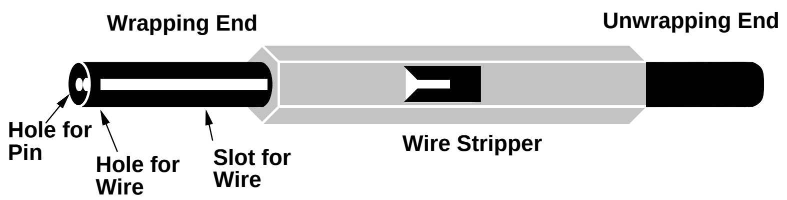
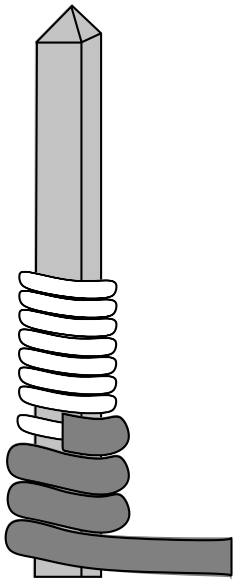


Figure 1: A Wire-Wrap tool (not to scale)

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 wirewrapping:

- 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[1], 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.

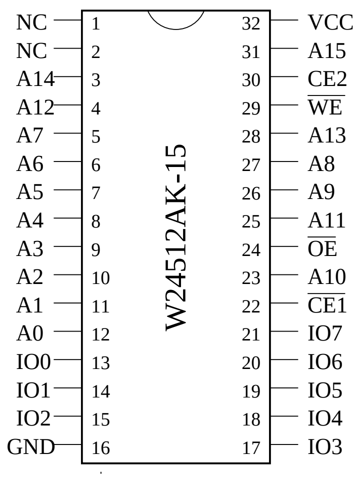
The W24512AK you use is a high speed SRAM. It takes only 15 nanoseconds for an address to produce output data, and comes in a standard DIP package.

Their electrical interface is simple:

[1]If you don't, inexpensive ones are available at Radio Shack. You probably should buy one, they are good things to have.

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

In normal operation CE2 should be tied high. The $\overline{\text{CE1}}$ input should be low to enable the chip. If $\overline{\text{WE}}$ is high and $\overline{\text{OE}}$ is low, the output lines always reflect the contents of the location specified by the address lines: change the address and the output will change. If $\overline{\text{WE}}$ is low, the data at the address will change to reflect the data input.



D0	72
D1	70
D2	69
D3	68
D4	67
D5	66
D6	65
D7	62
$\overline{\text{WE}}$	14
$\overline{\text{OE}}$	41
A0	10
A1	7
A2	5

A3	4
A4	3
A5	84
A6	83
A7	82
A8	81
A9	80
A10	79
A11	78
A12	77
A13	71
A14	60
A15	61

Figure 2: W24512AK Pinout(top view) and pins on the Xilinx

4.1 Testing the SRAM Wiring

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 recommend the brute force approach, as it is easier to implement. We provide a bit file `Wvlib/cs150/lab7.bit` which tests the SRAM , but you should also build your own version.

In addition to the wire wraps made to connect the SRAM, you need to connect the spare button to pin 58 and ground, so that you can use it in your design.

We provide a datapath for your test circuit `Wvlib/cs150/sch/lab7.1`, but no control logic. The datapath is fairly straghtforward. It consists of a 16 bit counter and an 8 bit LFSR. The lower 16 bits of the counter address the SRAM, and the LFSR data is output to the SRAM (write cycle) when the CE1.L, CE2.H, and WE.L signals are asserted. When the WE.L signal is deasserted and OE.L, CE1.L, CE2.H are asserted, the contents of the SRAM are read in (read cycle), and compared with the contents of the LFSR. These counters and flip flops have a synchronous reset, so reset must be high and the clock must come to reset them.

Your controller should write every memory location, and then read back all memory locations. It should clock the LFSR and the counter, to increment both the address and the LFSR. To write data, OE.L should first be deasserted (high), and after the address is stable, then on the NEXT clock cycle, assert the WE.L line (low). Once all the memory is full, both the counter and LFSR should be reset, and OE.L should be asserted (low), then the data in the SRAM should be read back out of memory, and compared with the LFSR value. If the data is different at any time, you should assert ERROR and stop operation. Otherwise, just repeat the test forever. The reset button should reset your controller.

Note that the TA pinout file `lab7.1` takes care of all of the pinouts for you. The WE and OE signals are inverted before being output, so your internal design can use OE.H and WE.H which are asserted high. (I.e. The internal signals OE=1, WE=0 when you want to read the RAM and WE=1, OE=0 when you want to write to the RAM.)

Name: ____________________ Name: ____________________

Lab Section (Check one)

M: ☐AM ☐PM T: ☐AM ☐PM W: ☐AM ☐PM Th: ☐PM

5 Checkoffs

1. At the START of 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 controller for the SRAM circuit, *which you entered before lab*. At the START of lab session, show your TA your simulation results for a write cycle and a read cycle, showing WE, OE, CNTR-CE (enable signal of the address counter), LFSR-CE(enable signal of the LFSR), and ERROR.

   SRAM test circuit TA: __________(25%)

3. Verify the SRAM is working by using the TA provided bit file.

   SRAM Tested TA: __________(20%)

4. Verify the SRAM is working by using your design.

   SRAM Tested TA: __________(25%)

5. Full. understanding of SRAM TA: __________(10%)

6. Turned in on time TA: __________(100%)(full credit)

7. Turned in one week late TA: __________(50%)(half credit)

.... Cut Here ....

W24512AK-15

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

W24512AK-15

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

W24512AK-15

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