EECS150 Fall 2000
Checkpoint #1
Wire-wrap, SRAM, and FIFO Design

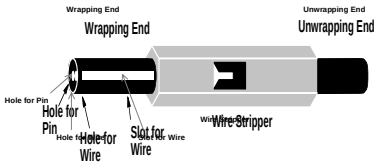
by Ramakrishna Gummadi
Norm Zhou

Goals of the Lab

- Learn to do wire-wrap
- Wire an SRAM to the Xilinx on the design demonstration board, test and debug it
- Learn another interesting application of LFSRs in generating and verifying test patterns
- Build the FIFO as a component that is going to be used later in the project

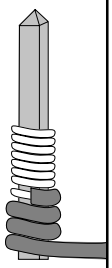
Using Wire-wrap

- You should have picked up a kit containing this tool from the TAs



Making a Wire-wrap Connection

- Plan the route of your wire
- Cut the wire ~ 2.5 inches longer than the route you choose
- Strip an inch of insulation off each end
- Insert a stripped end of the wire into the off-center hole in the end of the wrapping end of the tool
- Put the wrapping end of the tool on the pin to be connected
- Gently, without pushing down, or lifting up, twist the tool clockwise about 12 revolutions
- DO AS MUCH OF THE WIRE-WRAP WORK AS POSSIBLE BEFORE COMING TO THE LAB

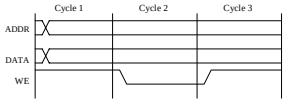


SRAMs

- Very fast, simple, more expensive than DRAM
- We use W24512AK that:
  - Has 8 data pins
  - 16 address pins
  - Can put out data in 15ns
- NOTE: the Xilinx pin assignment is different for this lab than the previous years' labs

Reading/writing From the SRAM

- Reading is easy:
  - Just assert OE and read the data line
- Writing, however, is a tricky:
  - Problem is, writes are asynchronous
  - Address and data line should stay stable, and not just on the clock edges



Testing SRAM

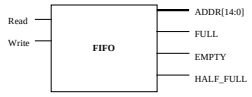
- How do we know whether the SRAM is working correctly?
  - Write test vectors into SRAM memory, read them back, and test if they match
- Two approaches to generating test vectors:
  - Use specific patterns
  - Use a brute-force approach
- We use the latter, except that we are a bit clever in the data we stick into the SRAM:
  - We use LFSRs to generate the data
  - Good, because we do not need to store the original patterns, and we can randomly test for all patterns

FIFOs

- Abstractly, fifos are like queues: you can enqueue elements into them, and dequeue them later, but only in the order in which they were enqueued (hence, the name)
- Can be implemented using two counters (head and tail), and a piece of SRAM
- In this class, however, the SRAM block is external to the FIFO

Better FIFOs

- Along with the "head" and "tail" pointers, we also have more control signals out of the fifo
- The fifo can use the control signals to indicate its state

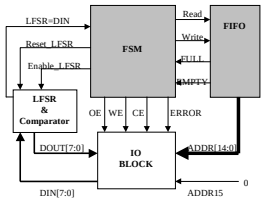


- This fifo is 32-K words deep

Notes on FIFO Design

- Since the SRAM is outside the FIFO, you need to route FIFO-generated addresses to SRAM address pins
- Recall that a FIFO has two counters, a head and a tail
  - Which counter's contents do we put out?
  - Answer: use a MUX or a tri-state buffer to determine whether a FIFO is being read or written

Putting It All Together



- The LFSR generates data for testing
- You must design the shaded parts
- NOTE: the OE, WE, and CE, are all active high externally