CS250 Discussion 4
SRAMs
Spring 2016
Christopher Yarp
Lab 3 Preview

• In lab 2, you changed the memory controller to take advantage of multiple in flight requests
  • You also experimented with pipelining

• In lab 3 you will experiment with:
  • Replacing the message buffer with an SRAM
  • Using a multi-Vt flow
  • Writing a script to parse reports
Why use SRAM Arrays?

• SRAMs are typically denser than flip-flop arrays*
  • Not necessarily true for very small arrays
  • Not necessarily true for large numbers of ports

• *Static*, meaning they hold state as long as power is applied
  • DRAM requires periodic refreshing of charge on capacitor

• Faster than DRAM

• Standard CMOS (no need for special DRAM process)

A good reference is *CMOS VLSI Design: A Circuit and Systems Perspective (4th ed)* by Weste and Harris
Flip-Flop Arrays vs. SRAMs

Flip-Flop Arrays (Registers)
• Can access as many elements at a time as you want (may need multiplexers to select)
• Can read the previous value from an element while writing the new one

SRAMs
• Number of ports limits how many simultaneous reads/write you can do at once
  • Read and write ports are not always the same
• Simultaneously reading and writing the same elements may produce unexpected result
  • Unlikely to behave like register
The Current Solution

1. Send Requests to Memory
2. Place Responses Into the Buffer (only one response at a time but possibly out of order)
Absorb by XOR-ing each buffer entry with the old state (simultaneously)
The Current Solution

4 Start the data path
The SRAM Solution

1. Send Requests to Memory
2. Place Responses Into the Buffer (only one response at a time but possibly out of order)
Absorb by XOR-ing each buffer entry with the old state (one word at a time)
The SRAM Solution

Buffer (SRAM Single Port)

```
| msgChunk[0] |
| msgChunk[1] |
| msgChunk[2] |
| msgChunk[3] |
| ...          |
| msgChunk[14] |
| msgChunk[15] |
| msgChunk[16] |
```

State (Flip-Flops)

```
| workingState[0] |
| workingState[1] |
| workingState[2] |
| workingState[3] |
| ...             |
| workingState[14]|
| workingState[15]|
| workingState[16]|
```

4 Start the data path
Multi Vt Flow

• Until now, you have only been using standard cells with one Vt
• Changing the Vt effects the speed and power of cells
  • LVT – faster but more power hungry
  • HVT – slower but less power hungry
• Tool will place LVT cells on the critical path to speed it up and will place HVT cells outside of the critical path to save power
Parsing Reports

• There are several scripting languages that you can use
  • Python, ruby, perl, awk, sed, ...
  • Doesn’t really matter what language you use.

• Regex will probably be your friend
  • There is a tutorial for python at https://docs.python.org/2/howto/regex.html

• Take pride in your script ... it will probably help you in your project!
  • Good script writing: taking more time initially to save a bunch of time later