These circuits are just write operation occurring.

RAM - random access memory

Review: Timing

• All gates have delays
  – RC delay in driving the output
• Wires are distributed RCs
  – Delays goes with the square of the length
• Source circuits determines strength
  – Synchronous vs asynchronous
• Delays in combinational logic determine by
  – Input delay
  – Path length
  – Delay of each gate along the path
• Worst case over all possible input-output paths
• Setup and CLK-Q determined by the two latches in flipflop
• Clock cycle: \( T_{\text{cycle}} = T_{\text{CLK}} + T_{\text{setup}} + T_{\text{delay}} \)
  \( + \text{ worst case skew} \)
• Delays can introduce glitches in combinational logic

Outline

• Memory concepts
• Register Files
• SRAM
• SRAM Access
• Multiported Memories
  – FIFOs
• ROM, EPROM, FLASH
• Relationship to Comb. Logic

Memory Basics

• Uses:
  Whenever a large collection of state elements is required.
  – data & program storage
  – general purpose registers
  – buffering
  – table lookups
  – CL implementation
• Types:
  – RAM - random access memory
  – ROM - read only memory
  – EPROM, FLASH - electrically programmable read only memory

Examples of RAM: Register file from microprocessor

Definition:

Memory Interfaces for Accessing Data

• Asynchronous (unclocked):
  A change in the address results in data appearing
• Synchronous (clocked):
  A change in address, followed by an edge on CLK results in data appearing or write operation occurring.
  A common arrangement is to have synchronous write operations and asynchronous read operations.

• Volatile:
  Losses its state when the power goes off.
• Nonvolatile:
  Retains its state when power goes off.
**Regid (address) Decoding**

- The function of the address decoder is to generate a one-hot code word from the address.
  - Binary to unary
  - Simplified DEMUX
- The output is used for row selection.
- Many different circuits exist for this function. A simple one is shown.
- Where have you seen this before?

**Accessing Register Files**

- **Read:** Output is a combinational function of the address input
  - Change address, see data from a different word on the output
  - Regardless of clock
- **Write is synchronous:**
  - If enabled, input data is written to selected word on the clock edge
- **Often multi-ported (more on that later)**

**Basic Memory Subsystem Block Diagram**

- **Address Decoder**
- **Word Line**
- **Memory Cell**
- **m Bit Lines**
- **RAM/ROM naming convention:**
  - 32 X 8, “32 by 8” => 32 8-bit words
  - 1M X 1, “1 meg by 1” => 1M 1-bit words

**Memory Components Types:**

- **Volatile:**
  - Random Access Memory (RAM):
    - SRAM “static”
    - DRAM “dynamic”
- **Non-volatile:**
  - Read Only Memory (ROM):
    - Mask ROM “mask programmable”
    - EPROM “electrically programmable”
    - EEPROM “erasable electrically programmable”
    - FLASH memory - similar to EEPROM with programmer integrated on chip

**Read Only Memory (ROM)**

- Simplified form of memory. No write operation needed.
- Functional Equivalence:
  - Connections to Vdd used to store a logic 1, connections to GND for storing logic 0.
- Full tri-state buffers are not needed at each cell point.
- In practice, single transistors are used to implement zero cells.
- Logic one’s are derived through precharging or bit-line pullup transistor.

**Static RAM Cell**

- **Read:**
  - 1. Select row
  - 2. Cell pulls one line low and one high
  - 3. Sense output on bit and bit
- **Write:**
  - 1. Drive bit lines (e.g., bit=1, bit=0)
  - 2. Select row
- **Why does this work?**
- When one bit-line is low, it will force output high; that will set new state
Typical SRAM Organization: 16-word x 4-bit

What happens when # bits gets large

• Big slow decoder
• Bit lines very log
  – Large distributed RC load

• Treat output as differential signal, rather than rail-to-rail logic
  – Sense amps on puts
  – Can ‘precharge’ both bit lines high, so cell only has to pull one low
• => Make it shorter and wider

Inside a Tall-Thin RAM is a short-fat RAM

Column MUX

• Controls physical aspect ratio
  – Important for physical layout and to control delay on wires.
• In DRAM, allows time-multiplexing of chip address pins (later)

Administration and Announcements

• Reading 10.4.1, Xilinx Block RAM datasheet
• HW 6 (two problems) due Friday
• Midterm Results
  – Max: 97, Mean: 87, Median 69
  – about 10 pt low due to length
  – will weight later one a little more
  – You will see the later problems again
  – make sure you know it
• Project adjustments
**Revised: Schedule of checkpoints**

- CP1: N64 interface (this week)
- CP2: Digital video encoder (week 8)
- CP3: SDRAM controller (two parts, week 9-10)
- CP4: IEEE 802.15.4 (cc2420) interface (wk 11-12)
  - unless we bail out to ethernet
  - Overlaps with midterm II
- Project CP: game engine (wk 13-14)
  - 11/29 early checkout
  - 12/6 final checkout
  - 12/10 project report due
  - 12/15 midterm III

**Logic Diagram of a Typical SRAM**

- Write Enable is usually active low (WE_L)
- Din and Dout are combined to save pins:
  - A new control signal, Output Enable (OE_L)
  - WE_L is asserted (Low), OE_L is unasserted (High)
  - WE_L is unasserted (High), OE_L is asserted (Low)
  - We_L and OE_L are asserted
    - Chip is disconnected or chipSelect (CS) + WE
  - Never both asserted!

**Cascading Memory Modules (or chips)**

- Example: assemble of 256 x 8 ROM using 256 x 4 modules:
- Example: 1K x * ROM using 256 x 4 modules:
  - each module has tri-state outputs:

**Typical SRAM Timing**

- OE determines direction
- Hi = Write, Lo = Read
- Wires are dangerous! Be careful
  - Double signaling: OE Hi, WE Lo

**Memory Blocks in FPGAs**

- LUTs can double as small RAM blocks:
  - A-LUT is really a 16x1 memory. Normally we think of the contents being written from the configuration bit stream, but Virtex architecture allows bits of LUT to be written and read from the general interconnect structure.
  - achievable 16x density advantage over using CLB flip-flops.
  - Furthermore, the two LUTs within a slice can be combined to create a 16 x 2-bit or 32 x 1-bit synchronous RAM, or a 16x1-bit dual-port synchronous RAM.
  - The Virtex-LUT can also provide a 16-bit shift register of adjustable length.

- Newer FPGA families include larger on-chip RAM blocks (usually dual ported)
  - Called block multiplyRAM in Virtex series
  - 4k bits each

**Synchronous SRAM**

- Figure 3: Timing Diagram for Single-Port Block SelectSRAM+ Memory
Verilog for Virtex LUT RAM

```verilog
module ram6x1(q, a, d, we, clk);
output q;
input a;
input [3:0] d;
input clk, we;
reg [15:0];
always @(posedge clk) begin
  if (we) 
    mem[a] <= d;
  end
assign q = mem[a];
endmodule
```

Note: synchronous write and asynchronous read.

- Deeper and/or wider RAMs can be specified and the synthesis tool will do the job of wiring together multiple LUTs.
- How does the synthesis tool choose to implement your RAM as a collection of LUTs or as block RAMs?

Multi-ported Memory

- **Motivation:**
  - Consider CPU core register file:
    - 1 read or write per cycle limits processor performance.
    - Complicates pipelining. Difficult for different instructions to simultaneously read or write regfile.
    - Common arrangement in pipelined CPUs is 2 read ports and 1 write port.

**Diagram:**

- `data_a`
- `sel_a`
- `Regfile`
- `data_b`
- `sel_b`

What do we need in the project?

First-in-first-out (FIFO) Memory

- **Used to implement queues.**
  - These find common use in computers and communication circuits.
  - Generally, used for rate matching data producer and consumer:

**Diagram:**

- **Producer:**
  - Producer can perform many writes without consumer performing any reads (or vice versa). However, because of finite buffer size, on average, need equal number of reads and writes.
  - Typical uses:
    - Interfacing I/O devices. Example network interface. Data bursts from network, then processor bursts to memory buffer (or reads one word at a time from interface). Operations not synchronized.
    - Example: Audio output. Processor produces output samples in bursts (during process swap-in time). Audio DAC clocks it out at constant sample rate.

- **Address:**
  - Address pointers are used internally to keep next write position and next read position into a dual-port memory.

**Diagram:**

- `D_in`
- `RST` CLK
- WE
- FULL
- HALF FULL
- EMPTY
- `RE`
- `D_out`
- `write ptr`
- `read ptr`
FIFO Implementation

- Assume, dual-port memory with asynchronous read, synchronous write.
- Binary counter for each of read and write address. CEs controlled by WE and RE.
- Equal comparator to see when pointers match.
- Flip-flop each for FULL and EMPTY flags:

<table>
<thead>
<tr>
<th>WE</th>
<th>RE</th>
<th>EMPTY</th>
<th>FULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Control logic with truth-table “draft” — “equal” determined after clock.
- when is empty and full set?
- could play funny games with.
- See Xilinx application notes
- Can waste a slot to simplify

Non-volatile Memory

- Used to hold fixed code (ex: BIOS), tables of data (ex: FFS next state/output logic), slowly changing values that persist over power off (date/time)
- Mask ROM — Used with logic circuits for tables etc.
- Contents fixed at IC fab time (truly write once)
- EPROM (erasable programmable)
- & FLASH — requires special IC process (floating gate technology)
- writing is slower than RAM, EPROM uses special programming system to provide special voltages and timing.
- reading can be made fairly fast.
- rewriting is very slow.
- — erase is first required, EPROM — UV light exposure, EEPROM — electrically erasable

FLASH Memory

- Electrically erasable
- In system programmability and erasability (no special system or voltages needed)
- On-chip circuitry (FSM) and voltage generators to control erasure and programming (writing)
- Erasure happens in variable sized “sectors” in a flash (16K - 64K Bytes)

See: http://developer.intel.com/design/flash/

- Compact flash cards are based on this type of memory.
  - NAND flash
  - Configuration memory, microcontrollers usually NOR flash

Relationship between Memory and CL

- Memory blocks can be (and often are) used to implement combinational logic functions:
- Examples:
  - LUTs in FPGAs
  - 1Mbit x 8 EPROM can implement 8 independent functions each of log₂(1M)x8
- The decoder part of a memory block can be considered a “minterm generator”.
- The cell array part of a memory block can be considered an OR function over a subset of rows.
- The combination gives us a way to implement logic functions directly in sum of products form.
- Several variations on this theme exist in a set of devices called Programmable logic devices (PLDs)

Xilinx BlockRam Versions

- Our simple version has latency problems.
- FULL and EMPTY signals are asserted near the end of the clock period.
- Xilinx version solves this by “predicting” when full or empty will be asserted on next write/read operation. Also uses BlockRam with synchronous reads.
- Available on Xilinx website along with “app note” — application note. These are linked to our page.
- Two versions available. Easy to modify to change the width if necessary.
- Will use the “cc” (common clock) version.

A ROM as AND/OR Logic Device

- A ROM can be used to implement logic functions.
- The combination gives us a way to implement logic functions directly in sum of products form.
- Several variations on this theme exist in a set of devices called Programmable logic devices (PLDs)
PLD Summary

- Basic RAM structure:
  - Address decoder to select row of cell array
  - Bit, ~bit lines to read/write
  - Sense difference in each bit
  - Column mux

- Read/write protocols:
  - Synchronous (regfiles, FPGA block ram)
  - Asynchronous read, synchronous writes
  - Asynchronous

- Multiported RAMs:
  - reg files and fifos

- Non-volatile memory:
  - ROM, EPROM, EEPROM, FLASH

- Memory as combinational logic

- Relationship to programmable logic