CSE140: Components and Design Techniques for Digital Systems

Tajana Simunic Rosing
Announcements & Outline

- HW#7 due, HW#8 out
- Midterm back Thursday
- Today: Memory
  - ROM
  - RAM
  - FIFO
  - Queue
- Next: Register Transfer Level Design
CSE140: Components and Design Techniques for Digital Systems

Memory

Tajana Simunic Rosing
Memor: basic concepts

- Stores large number of bits
  - $m \times n$: $m$ words of $n$ bits each
  - $k = \log_2(m)$ address input signals
  - or $m = 2^k$ words
  - e.g., $4,096 \times 8$ memory:
    - 32,768 bits
    - 12 address input signals
    - 8 input/output data signals

- Memory access
  - r/w: selects read or write
  - enable: read or write only when asserted
  - multiport: multiple accesses to different locations simultaneously

Sources: TSR, Katz, Boriello, Vahid
Write ability/ storage permanence

- Traditional ROM/RAM
  - ROM
    - read only, bits stored without power
  - RAM
    - read and write, lose stored bits without power
- Distinctions blurred
  - Advanced ROMs can be written to
    - e.g., EEPROM
  - Advanced RAMs can hold bits without power
    - e.g., NVRAM

Write ability and storage permanence of memories, showing relative degrees along each axis (not to scale).

Sources: TSR, Katz, Boriello, Vahid
Random Access Memory (RAM)

- RAM – Readable and writable memory
  - Logically the same as register file
    - RAM just one port; register file two or more
  - RAM vs. register file
    - RAM is larger
    - RAM stores bits using a bit storage vs. FFs
    - RAM implemented on a chip in a square – keeps longest wires (hence delay) short

Sources: TSR, Katz, Boriello, Vahid
• Similar internal structure as register file
  – Decoder enables appropriate word based on address inputs
  – rw controls whether cell is written or read
  – Let’s see what’s inside each RAM cell

Sources: TSR, Katz, Boriello, Vahid
Static RAM (SRAM) - writing

- “Static” RAM cell
  - 6 transistors (recall inverter is 2 transistors)
  - Writing this cell
    - *word enable* input comes from decoder
    - When 0, value *d* loops around inverters
      - That loop is where a bit stays stored
    - When 1, the *data* bit value enters the loop
      - *data* is the bit to be stored in this cell
      - *data*’ enters on other side
      - Example shows a “1” being written into cell

Sources: TSR, Katz, Boriello, Vahid
• “Static” RAM cell - reading
  – When rw set to read, the RAM logic sets both *data* and *data'* to 1
  – The stored bit d will pull either the left line or the right bit down slightly below 1
  – “Sense amplifiers” detect which side is slightly pulled down
Dynamic RAM (DRAM)

- "Dynamic" RAM cell
  - 1 transistor (rather than 6)
  - Relies on large capacitor to store bit
    - Write: Transistor conducts, data voltage level gets stored on top plate of capacitor
    - Read: Just look at value of $d$
    - Problem: Capacitor discharges over time
      - Must "refresh" regularly, by reading $d$ and then writing it right back

Sources: TSR, Katz, Boriello, Vahid
Comparing Memory Types

- Register file
  - Fastest
  - But biggest size
- SRAM
  - Fast
  - More compact than register file
- DRAM
  - Slowest
    - And refreshing takes time
  - But very compact
  - Different technology for large caps.

Size comparison for the same number of bits (not to scale)

Sources: TSR, Katz, Boriello, Vahid
Generic SRAM timing

Sources: TSR, Katz, Boriello, Vahid
Generic DRAM timing

Sources: TSR, Katz, Boriello, Vahid
Page mode access

CE’
R/W’
RAS’
CAS’
Adrs
Data
time

Sources: TSR, Katz, Borriello, Vahid
Ram variations

• PSRAM: Pseudo-static RAM
  – DRAM with built-in memory refresh controller
  – Popular low-cost high-density alternative to SRAM

• NVRAM: Nonvolatile RAM
  – Holds data after external power removed
  – Battery-backed RAM
    • SRAM with own permanently connected battery
    • writes as fast as reads
    • no limit on number of writes unlike nonvolatile ROM-based memory
  – SRAM with EEPROM or flash
    • stores complete RAM contents on EEPROM or flash before power turned off

Sources: TSR, Katz, Boriello, Vahid
Extended data out DRAM

- Improvement of FPM (full page mode) DRAM
- Extra latch before output buffer
  - allows strobing of cas before data read operation completed
- Reduces read/write latency by additional cycle

Sources: TSR, Katz, Boriello, Vahid
(S)ynchronous and Enhanced Synchronous (ES) DRAM

- SDRAM latches data on active edge of clock
- Eliminates time to detect *ras/cas* and *rd/wr* signals
- A counter is initialized to column address then incremented on active edge of clock to access consecutive memory locations
- ESDRAM improves SDRAM
  - added buffers enable overlapping of column addressing
  - faster clocking and lower read/write latency possible
Rambus DRAM (RDRAM)

• More of a bus interface architecture than DRAM architecture
• Data is latched on both rising and falling edge of clock
• Broken into 4 banks each with own row decoder
  – can have 4 pages open at a time
• Capable of very high throughput

Sources: TSR, Katz, Borriello, Vahid
DRAM integration problem

- SRAM easily integrated with CPU
- DRAM more difficult
  - Different chip making process between DRAM and conventional logic
  - Goal of conventional logic (IC) designers:
    - minimize parasitic capacitance to reduce signal propagation delays and power consumption
  - Goal of DRAM designers:
    - create capacitor cells to retain stored information

Sources: TSR, Katz, Boriello, Vahid
• **Behavior**
  - **Record:** Digitize sound, store as series of 4096 12-bit digital values in RAM
    - We'll use a 4096x16 RAM (12-bit wide RAM not common)
  - Play back later from RAM

Sources: TSR, Katz, Boriello, Vahid
**RAM Example: Digital Sound Recorder**

*Record* behavior

- Local register: $a$ (12 bits)
- $a = 0$
- $a = a + 1$
- $a < 4095$
- $a = 4095$

Keep local register $a$
- Stores current address, ranges from 0 to 4095
- Create state machine that counts from 0 to 4095 using $a$
- For each $a$
  - Read analog-to-digital conv: $ad_{ld}=1$, $ad_{buf}=1$
  - Write to RAM at address $a$: $Ra=a$, $Rrw=1$, $Ren=1$

Sources: TSR, Katz, Boriello, Vahid
Create state machine that counts from 0 to 4095; for each a:
- Read RAM
- Write to digital-to-analog conv.
- Note: Must write d-to-a one cycle after reading RAM, when the read data is available on the data bus
Read-Only Memory – ROM

- Memory that can only be read from
  - Data lines are output only
- Advantages over RAM
  - Nonvolatile
  - Low power
  - Compact

Let $A = \log_2 M$

Sources: TSR, Katz, Boriello, Vahid
**ROM Types**

- **Mask-programmed ROM**
  - Bits are hardwired as 0s or 1s during chip manufacturing

- **Fuse-Based Programmable ROM**
  - Each cell has a fuse
  - Programmer blows certain fuses (using higher-than-normal voltage)
    - Those cells will be read as 0s (involving some special electronics)
    - Cells with unblown fuses will be read as 1s
  - Aka. **One-Time Programmable ROM**

Sources: TSR, Katz, Boriello, Vahid
ROM Types

- **Erasable Programmable ROM (EPROM)**
  - Uses “floating-gate transistor” in each cell
  - Programmer uses higher-than-normal voltage to cause electrons to *tunnel* into the gate
    - Electrons become trapped in the gate
    - Only done for cells that should store 0
    - Other cells will be 1
  - To erase, shine ultraviolet light onto chip
    - Gives trapped electrons energy to escape
    - Requires chip package to have window

- **Electronically-Erasable Programmable ROM (EEPROM)**
  - Programming similar to EPROM
  - Erasing one word at a time *electronically*

- **Flash memory**
  - Like EEPROM, but large blocks of words can be erased *simultaneously*

- EEPROM & FLASH are in-system programmable

Sources: TSR, Katz, Boriello, Vahid
ROM Example: Digital Telephone Answering Machine

- Record the outgoing announcement
  - When rec=1, record digitized sound in locations 0 to 4095
  - When play=1, play those stored sounds to digital-to-analog converter
High-level state machine

- Once $\text{rec}=1$, begin erasing flash by setting $\text{er}=1$
- Wait for flash to finish erasing by waiting for $\text{bu}=0$
- Execute loop that sets local register $a$ from 0 to 4095, reading analog-to-digital converter and writing to flash for each $a$
Making memory words wider

- Easy – just place memories side-by-side until desired width obtained
- Share address/control lines, concatenate data lines
- Example: Compose 1024x8 ROMs into 1024x32 ROM

Sources: TSR, Katz, Boriello, Vahid
Composing Memory – More Words

a10 just chooses which memory to access

- Creating memory with more words
  - Combine memories until the number of desired words is achieved
  - Use decoder to select
  - Example: Compose 1024x8 memories into 2048x8 memory
- More words and wider words – first make enough words, then widen

Sources: TSR, Katz, Boriello, Vahid
Queues

- **FIFO Queue** (first-in-first-out)
  - Write at the back: *push*, Read at the front: *pop*
  - Treat memory as a circle

- **Common uses:**
  - Computer keyboard
    - Pushes pressed keys onto queue; Meanwhile pop and send to computer
  - Digital video recorder
    - Pushes frames onto queue; Meanwhile pops frames, compresses them, and stores them
  - Routers
    - Pushes incoming packets onto queue; Meanwhile pops packets, processes destination information, and forwards each packet out over appropriate port

Sources: TSR, Katz, Boriello, Vahid
Queues

- Two conditions have front=rear need FSM to detect:
  - Full: No pushes until a pop
  - Empty: No pops until a push
- Use Register file for storage
- Implement Rear and front with up counters:
  - rear as RF’s write address, front as read address

Sources: TSR, Katz, Boriello, Vahid
Summary

• Memory
  – RAM
  – ROM
  – Combining Memory
  – Queue

Sources: TSR, Katz, Boriello, Vahid