Memory Components

- Register-transfer level design instantiates datapath components to create datapath, controlled by a controller
  - A few more components are often used outside the controller and datapath
- \( M \times N \) memory
  - \( M \) words, \( N \) bits wide each
- Several varieties of memory, which we now introduce

RAM Internal Structure

- Similar internal structure as register file
  - Decoder enables appropriate word based on address inputs
  - \( w\text{data} \) controls whether cell is written or read
  - Let's see what's inside each RAM cell

Random Access Memory (RAM)

- RAM – Readable and writable memory
  - "Random-access memory"
    - Strange name – Created several decades ago to contrast with sequentially-accessed storage like tape drives
    - Logically same as register file – Memory with address inputs, data inputs/outputs, and control
    - RAM usually just one port; register file usually two or more
  - RAM vs. register file
    - RAM typically larger than roughly 512 or 1024 words
    - RAM usually just one port; register file usually two or more
  - RAM typically implemented on a chip in a square rather than rectangular shape – keeps longest wires (hence delay) short

Static RAM (SRAM)

- "Static" RAM cell
  - 6 transistors (recall inverter is 2 transistors)
  - Writing this cell
    - \( w\text{data} \) controls whether cell is written or read
    - Let's see what's inside each RAM cell
• "Static" RAM cell
  - Reading this cell
    - Somewhat trickier
      - When the bit to read, the RAM logic sets both data and data!'
      - The stored bit of cell will either the left line or the
        right bit (down slightly below 1)
      - It may look as "neutral" because which side is slightly
        pulled down
    - The electrical description of SRAM is really
      beyond our scope – just general idea here, mainly to contrast with DRAM.
  - Writing
    - 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

SRAM cell

- The electrical description of SRAM is really
  beyond our scope – just general idea here, mainly to contrast with DRAM.

- Writing
  - Put address on en and data on data lines, set en=1, en=1
  - Data will appear on data lines
  - Don’t forget to obey setup and hold times
  - In short – keep inputs stable before and after a clock edge

- Reading
  - Set addr and en lines, but put nothing (Z) on data lines, set en=0
  - Data will appear on data lines

- SRAM is often a separate chip
- SRAM for large items, and DRAM for huge items
- Note: DRAM is big capacitor requires a
  special chip design process, so DRAM
  is often a separate chip
- SRAM is often a separate chip

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

- Dynamic RAM (DRAM)
  - 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

- DRAM cell
  - Capacitor slowly discharging

RAM Example: Digital Sound Recorder

- Behavior
  - Record: Digitize sound, store as series of 4096 12-bit digital values in RAM
    - Each use a 4096x16 RAM (12-bit wide RAM not common)
    - Play back later
    - Common behavior in telephone answering machine, toys, voice recorders
  - To record, processor should read a-to-d, store read values into successive
    RAM words
    - To play, processor should read successive RAM words and enable d-to-a

- RTL design of processor
  - Create high-level state machine
    - Begin with the record behavior
    - Keep local register
      - Stores current address, ranges from 0 to 4095 (thus need 12 bits)
    - Create state machine that
      counts from 0 to 4095 using a
        - For each a
          - Read analog to digital converter
            - ad, bd=1, ad, bd=1
          - Write to RAM at addresses a, b = 1, b=1
  - Local register x 12 bits
  - RAM Example: Digital Sound Recorder

- RTL design of processor
  - Create high-level state machine
    - Begin with the record behavior
    - Keep local register
      - Stores current address, ranges from 0 to 4095 (thus need 12 bits)
    - Create state machine that
      counts from 0 to 4095 using a
        - For each a
          - Read analog to digital converter
            - ad, bd=1, ad, bd=1
          - Write to RAM at addresses a, b = 1, b=1
  - Local register x 12 bits
Read-Only Memory – ROM

- Memory that can only be read from, not written to
  - Data lines are output only
  - No need for read input
- Advantages over RAM
  - Compact: May be smaller
  - Nonvolatility: Saves bits even if power supply is turned off
  - Speed: May be faster (especially than DRAM)
  - Low power: Doesn’t need power supply to save bits, so can extend battery life
- Choose ROM over RAM if stored data won’t change (or won’t change often)
  - For example, a table of Celsius to Fahrenheit conversions in a digital thermometer

ROM Types

- If a ROM can only be read, how are the stored bits stored in the first place?
  - Storing bits in a ROM known as programming
  - Several methods
- Mask-programmed ROM
  - Bits are hardened as 0s or 1s during chip manufacturing
  - 2-bit word on right stores "10" word enable (from decoder) simply passes the hardened value through transistor
  - Notice how compact, and fast, this memory would be

ROM Types

- Fuse-Based Programmable ROM
  - Each cell has a fuse
  - A special device, known as a programmer, blows certain fuses (using higher-than-normal voltage)
  - Those cells will be read as 0’s (involving some special electronics)
  - Cells with unblown fuses will be read as 1’s
  - 2-bit word on right stores "10"
  - Also known as One-Time Programmable (OTP) ROM

ROM Types

- Erasable Programmable ROM (EPROM)
  - Uses “floating-gate transistor” in each cell
  - Special programmer device 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 (without electrons trapped in gate) will be 1
  - 2-bit word on right stores "10"
  - Details beyond our scope – just general idea is necessary here
  - To erase, shine ultraviolet light onto chip
  - Gives trapped electrons energy to escape
  - Requires chip package to have window
ROM Types

- **Electronically-Erasable Programmable ROM (EEPROM)**
  - Similar to EPROM
  - Uses floating-gate transistor, electronic programming to trap electrons in certain cells
  - But erasing done electronically, not using UV light
  - Erasing done one word at a time
- **Flash memory**
  - Like EEPROM, but all words (or large blocks of words) can be erased simultaneously
  - Become common relatively recently (late 1990s)
- Both types are in-system programmable

ROM Example: Talking Doll

- **High-level state machine**
  - Create state machine that waits for \( \text{vib}=1 \), and then counts from 0 to 4095 using a local register \( a \)
  - For each \( a \), read ROM, write to digital-to-analog converter

ROM Example: Digital Telephone Answering Machine Using a Flash Memory

- **Blurring of Distinction Between ROM and RAM**
  - We said that
    - RAM is readable and writable
    - ROM is read-only
  - But some ROMs act almost like RAMs
    - EEPROM and Flash are in-system programmable
    - Essentially means that writes are slow
  - And because message will never change, use a mask-programmed ROM or OTP ROM
  - Processor should wait for vibration (\( \text{vib}=1 \)), then read words 0 to 4095 from the ROM, writing each to the d-to-a
  - Also need busy output to indicate that erasing is in progress
  - Requires bi-directional data lines, and write control input
  - Uses floating-gate transistor, electronic programming to trap electrons, not using UV light
  - Non-volatile RAMs: Can save their data without the power supply
  - Message must be stored without power supply
    - Use a ROM, not a RAM, because ROM is nonvolatile
    - And because message will never change, use a mask-programmed ROM or OTP ROM
  - \( \text{we're not home} \)
  - RAM is readable and writable
    - Processor should wait for vibration (\( \text{vib}=1 \)), then read words 0 to 4095 from the ROM, writing each to the d-to-a

ROM Example: Digital Telephone Answering Machine Using a Flash Memory

- **ROM Example: Talking Doll**
  - Doll plays prerecorded message, trigger by vibration
    - Message must be stored without power supply
    - Use a ROM, not a RAM, because ROM is nonvolatile
    - And because message will never change, use a mask-programmed ROM or OTP ROM
  - Processor should wait for vibration (\( \text{vib}=1 \)), then read words 0 to 4095 from the ROM, writing each to the d-to-a
Creating memory with more words - Put memories on top of one another until the number of desired words is achieved
- Use decoder to select among the memories
- Can use highest order address input(s) as decoder input
- Example: Compose 1024x8 memories into 2048x8 memory

Hierarchy and Composing Larger Components from Smaller Versions

Composing memory very common
- 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

Hierarchy and Abstraction

Abstraction - Hierarchy often involves not just grouping items into a new item, but also associating higher-level behavior with the new item, known as abstraction
- e.g., an 8-bit adder has an understandable high-level behavior - it adds two 8-bit binary numbers
- Frees designer from having to remember, or even from having to understand, the lower-level details

Hierarchy - A Key Design Concept

- Hierarchy
  - An organization with a few items at the top, with each item decomposed into other items
  - Common example: A country
    - 1 item at the top (the country)
    - Province item decomposed into state/province
    - Each state/province item decomposed into city items
- Hierarchy helps us manage complexity
  - To go from transistors to gates, muxes, decoders, registers, ALUs, controllers, datalatches, memories, queues, etc.
  - Imagine trying to comprehend a controller and datapath at the level of gates