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
- MxN 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
  - en 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 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
    - en controls whether cell is written or read
    - Let's see what's inside each RAM cell
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
- **Use register file for small items, SRAM for large items, and DRAM for huge items**
  - Note: DRAM's big capacitor requires a special chip design process, so DRAM is often a separate chip.

**Static RAM (SRAM)**

- "Static" RAM cell
  - Reading this cell
    - Somewhat trickier
    - When we set to read, the RAM logic sets both data and address
    - The stored bit at cell pull either left or the right bit down slightly below
    - The process is a little different 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.

**Dynamic RAM (DRAM)**

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

**Reading and Writing a RAM**

- **Writing**
  - Put address on address lines, data on data lines, set en=1, write=false
- **Reading**
  - Set address lines, but put nothing (Z) on data lines, set en=1, read=false
  - 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

**RAM Example: Digital Sound Recorder**

- **Behavior**
  - Record: Digitize sound, store as series of 4096 12-bit digital values in RAM
    - The 12-bit word RAM is 12-bit wide RAM
  - 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

**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
- **Use register file for small items, SRAM for large items, and DRAM for huge items**
  - Note: DRAM's big capacitor requires a special chip design process, so DRAM is often a separate chip.
**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 0s (involving some special electronics)
    - Cells with unblown fuses will be read as 1s
  - 2-bit word on right stores "10"
  - Also known as One-Time Programmable (OTP) ROM

- **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 1s
  - 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

**Read-Only Memory – ROM**

- Memory that can only be read from, not written to
  - Data lines are output only
  - No need for re-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

**Read-Only Memory – RAM**

- 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

- **High-level state machine**
  - Create state machine that waits for \( v = 1 \), 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: Talking Doll**

  - Doll plays prerecorded message, trigger by vibration
  - Message must be stored without power supply
  - Processor should wait for vibration (\( v = 1 \)), 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**

  - Want to record the outgoing announcement
  - When \( v = 1 \), record digitized sound in locations 0 to 4095
  - When \( v = 1 \), play those stored words to digital-to-analog converter
  - What type of memory?
    - Should store without power supply — ROM, not RAM
    - Should be in-system programmable — EPROM or Flash, not EEPROM, OTP ROM, or mask-programmed ROM
    - ROM always erase entire memory when reprogramming
    - Flash better than EEPROM

- **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
    - Also, number of writes may be limited (perhaps a few million times)
  - And, some RAMs act almost like ROMs
    - Non-volatile RAMs can save their data without the power supply
    - One type: Built-in battery, may work for up to 10 years
    - Another type: Includes ROM backup for RAM — controller writes RAM contents to ROM before turning off
  - New memory technologies evolving that merge RAM and ROM benefits
    - e.g., MRAM
  - Bottom line
    - Lot of choices available to designer, must find best fit with design goals
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)
    - Country item decomposed into state/province items
    - Each state/province item decomposed into city items
- Hierarchy helps us manage complexity
  - To go from transistors to gates, muxes, decoders, registers, ALUs, controllers, datapaths, memories, queues, etc.
  - Imagine trying to comprehend a controller and datapath
  - Frees designer from having to remember, or even from having to understand, the lower-level details
  - Frees designer from having to remember, or even from having to understand, the lower-level details

Hierarchy and Composing Larger Components from Smaller Versions

- A common task is to compose smaller components into a larger one
  - Gates: Suppose you have plenty of 3-input AND gates, but need a 9-input AND gate
    - Can simple compose the 9-input gate from several 3-input gates
  - Muxes: Suppose you have 4x1 and 2x1 muxes, but need an 8x1 mux
    - 2x1 selects either top or bottom 4
    - 4x1 selects particular 4-bit input
    - Implements 8x1 mux – 8 data inputs, 3 selects, one output
- Making memory words wider
  - Example: Compose 1024x8 ROMs into 1024x32 ROM
    - Share address/control lines, concatenate data lines
    - Easy – just place memories side-by-side until desired width obtained
  - Example: Compose 1024x8 ROMs into 1024x32 ROM

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 higher-order address inputs as decoder inputs
  - Example: Compose 1024x8 memories into 2048x8 memory

Hierarchy and Composing Larger Components from Smaller Versions

- Hierarchy and Abstraction
  - Abstraction 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 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 Composing Larger Components from Smaller Versions

- Creating memory with more words and wider words
  - Can first compose to enough words, then widen.
  - Use decoder to select among the memories
  - Can use higher-order address inputs as decoder inputs
  - Example: Compose 1024x8 memories into 2048x8 memory

Hierarchy and Composing Larger Components from Smaller Versions

- To create memory with more words and wider words, can first compose to enough words, then widen.