MEMORIES
Memory in Digital Systems

• Three primary components of digital systems
  – Datapath (does the work)
  – Control (manager)
  – Memory (storage)
Memories

• Use in general digital processors
  – Instructions
  – Data

• Uses more common in digital signal processors
  – Buffering input/intermediate/output data (e.g., rate matching)
  – Storing fixed numbers (e.g., coefficients)
  – Often relatively small (e.g., 8-64-256 words) and numerous (dozens are not unusual)
Memories

• Read-write memories
  - SRAM: static memory
    • Data is stored as the state of a bistable circuit
    • State is retained without refresh as long as power is supplied
  - DRAM: dynamic memory
    • Data is stored as a charge on a capacitor
    • State leaks away, refresh is required

• NVRWM: non-volatile read-write memory
  - Flash: ROM at low voltages, writable at high voltages
  - EPROM: ROM, but erasable with UV light

• ROM: read-only memory
  - Non-volatile—mask programmed at manufacture
  - Fuses—links blown during manufacture
  - Anti-fuses—links made during manufacture
Memories

- Memories generally contain several components:
  - Array of cells
  - Word decoder
  - Write circuitry
  - Read circuitry (sense amplifiers)
  - Wordlines
  - Bitlines

- Interface signals
  - Address (one for each port)
  - Data (one for each port)
  - Enable_write
  - Enable_read (likely)
  - Clock (sometimes)
Memories—Differential bitlines

- Differential bitlines (bitline and bitline_) require more area but dramatically increase robustness and speed.
Decoder Design

• Example: 256-word memory (8 addr bits)
  – Straightforward approach
    • Route 8 address wires plus inverted versions (16 wires) along array
    • Each word uses an 8-input AND gate
    • Each long wire has N/2=128 gate loads
**Critical path**

- INV + 8-input AND
- Neglect assertion level because we’ll have to add a number of inverters (buffers) anyway
- Building an 8-input AND:
  - AND-AND (inefficient)
  - NAND-NOR (better)
- Total critical path
  - INV + 4-input NAND + 2-input NOR, or
  - INV + 3-input NAND + 3-input NOR
Decoder Design
Predecoding

- Example: 256-word memory (8 addr bits)
  - Predecode
    - Ex: take groups of 2 address bits
      - Four possibilities; activate one of four wires (use 2 INVs, 4 ANDs)
    - Wires along array: 4 * (8/2) = 4 groups of 4 = 16 (same as non-predecoded)
    - Each word uses a 4-input AND gate (much faster)
    - Each long wire has N/4=64 gate loads (half of other approach!)
    - Works best with large memories
      - May have less toggling
      - May be faster
• **Critical path**
  
  - Predecoder + 4-input AND
  
  - Neglect assertion level because we’ll have to add a number of inverters (buffers) anyway

  - INV + 2-input AND
    
    + 4-input AND

    • May be able to use NANDs instead of ANDs
6-Transistor (6T) SRAM Cell

- Cross-coupled inverters: bistable element
- Density is important in memories
  - A lot of effort spent packing transistors and even pushing process design rules just for the 6T memory cell
Layout: 6-Transistor (6T) Cell

- Two 6T SRAM cells
- Wordlines horizontal, bitlines vertical
- Horizontal $V_{dd}$ and $G_{nd}$
6-Transistor (6T) Cells

- Micro-photographs of fabricated 6T SRAM memory cells
- The devices are partially built with diffusion and polysilicon only—metal interconnect is shown where it might be located

![Intel 65 nm 0.57 μm² 6T cell](http://www.electroiq.com/articles/sst/print/volume-47/issue-2/departments/tech-news/technology-news.html)
Layout: Memory Array

- 128 words x 36 bits
- Single-ported 6T SRAM
- Low-power hierarchical bitline structure
Layout: Memory array

- Layout issues
  - Cell density is critical
  - Decoder design should probably not use full $N$-word fanout; predecoding address bits often used
Multi-ported SRAM

• Frequently used in register files
  – Classic RISC computers have 1 write and 2 read ports
  – Modern multiple-instruction-issue computers can have many ports (22 (12 Rd, 10 Wr) in Itanium, ISSCC 05)

• More commonly use single-ended (non-differential) bitlines
Multi-ported SRAM

- Example: one write port, two read ports
- If feedback inverter is not tri-statable during writes, it must be made weak
Layout: Dual-ported memory cell

- 10T one-read, one-write port
- Operates at very low supply voltages
Layout: Dual-ported memory cell array

- Eight 10T dual-ported memory cells
Layout: Dual-ported memory array

- 10T one-read, one-write port
- 16 words x 40 bits

write decoder
read decoder
bitline drivers and sense amplifiers
DRAM (Dynamic RAM)

- Smaller cell size (1T cell)
  - One transistor to access cell
- Single bitline to read and write cell
- Must be periodically refreshed
- Requires a sophisticated sensing scheme
• Similar to DRAM except the source of the access transistor is tied to $Gnd$
• Strongly drives bitline low when transistor present
• For the other value (call it 1), the transistor is omitted (probably no diffusion) or disconnected (no contact)
• Sometimes better to use synthesized random logic instead of small ROMs
Memories for Custom-Designed Processors
(Also known as "ASIC")

- Memories for on-chip processors can be built in a number of ways:
  1) On-chip “macro” memory arrays
     - Think of as a single giant standard cell
     - FPGAs include them (block RAM)
  2) On-chip memory synthesized from standard cells
  3) Off-chip memories
     - Very large DRAM
     - Non-volatile memory such as flash memory
1) Memory Macro-cells

- Memory macro-cell generators are available for larger memories
- Typically a software tool generates a large variety of possible memories where a user may select options such as:
  - Number of words
  - Word-width (in bits)
  - Number of read ports
  - Number of write ports
  - Rd/wr or ROM
  - Built-in test circuits
- Tool produces models for verilog, place & route, and other CAD views
1) On-chip “macro” memory arrays

• Generally very area efficient due to dense memory cells (single-ported memories likely use 6-transistor (6T) memory cells)

• Generally good energy efficiency due to low-activity memory array architecture

[T. Nanya, et al., TITAC-2 processor]
2) Synthesized Memory

- Can synthesize memory from standard cells
  - Memory cells are now flip-flops
  - \( clk \) likely routed to all cells
  - Probably best for small memories only
  - Read bitline logic may be muxes
2) Synthesized Memory

- Standard cell layout is typically irregular
  - Wires not shown
  - Clocks routed to each “reg” (flip-flop)
Verilog Memories

- Declaring a 16-bit, 128-word memory
  - `reg [15:0] Mem [0:127];`

- Reading
  - `source1 = Mem[addr_rd];`

- Writing
  - Blocking or non-blocking depends on how you want to handle simultaneous reads and writes
  - `Mem[addr_wr] <= #1 c_datapath_out; // makes sense for FFs`
  - `Mem[addr_wr] = c_datapath_out;`
Verilog Memories

• Reading and writing
  – Cannot access a portion of a word without first reading the whole word in many simulators and CAD tools (though it is supported in some tools). Good practice to not do it!
    - source1 = Mem[addr_rd][5]; // won’t work
    - abc     = Mem[addr_rd];    // use these 2 lines instead
      source1 = abc[5];

• Multiple read ports
  – Simply make individual read statements for each read port
    - data1   = Mem[addr_rd1];
    - data2   = Mem[addr_rd2];
    - data3   = Mem[addr_rd3];
• Style 1: Registers are outside the memory array
• Memory macros typically register input addresses and data "outside" the memory array
For cases when the memory is a combinational block
- Add pipeline registers outside block
- Style 2: Register is in the middle of the memory array
- Standard cell memories store bits in the array in FFs
  - This forces a pipe stage across the middle of the memory array
Memory Pipelining/Timing
Timing Style 2

- Built-in pipeline stage in memory array
- May place logic in memory pipe stages to balance pipeline