Memory Circuits & the CPU

CSC 103

September 20, 2005

Overview for Today

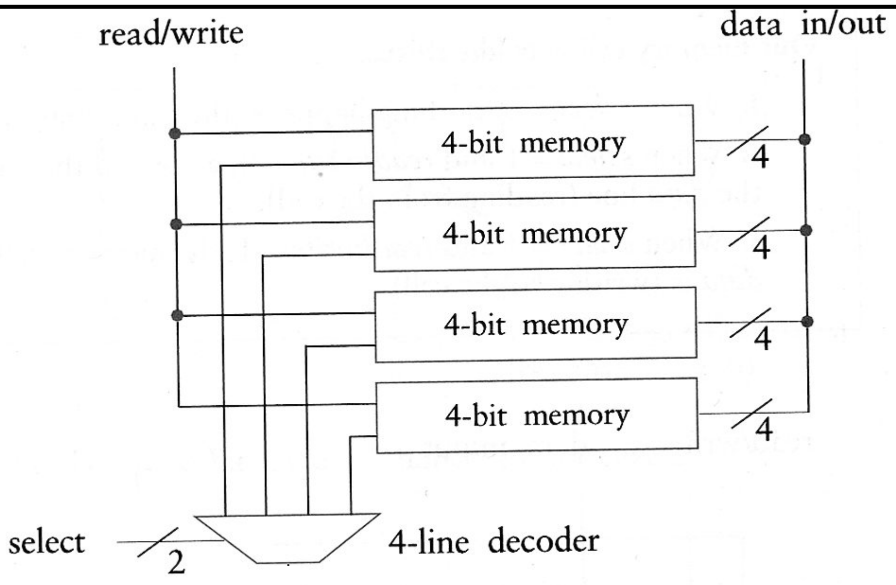
- Memory circuits
  - The importance of feedback
  - The organization of RAM
  - Control signals – R/W & Select
  - Data signals – a single shared data bus
- The CPU – central processing unit
  - Fetch-execute cycle
  - The Pippin simulator

Memory Circuits & Feedback

Physical Wiring of RAM

- Data bus
  - The single data bus is connected simultaneously to all locations in RAM
  - The data bus is for data input *and* data output
- Control
  - ‘Select’ specifies which address/location to read or write
  - ‘R/W’ specifies whether the location will be read or written

Memory Array



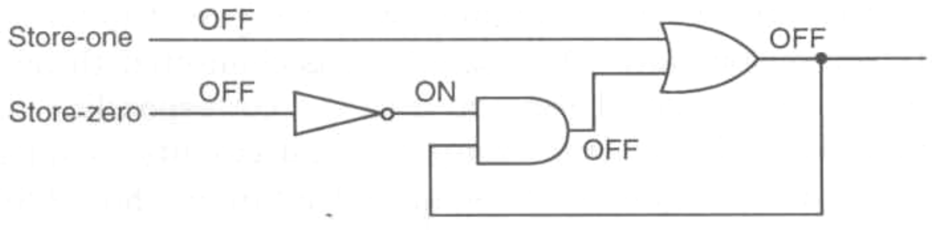
Memory Array Construction

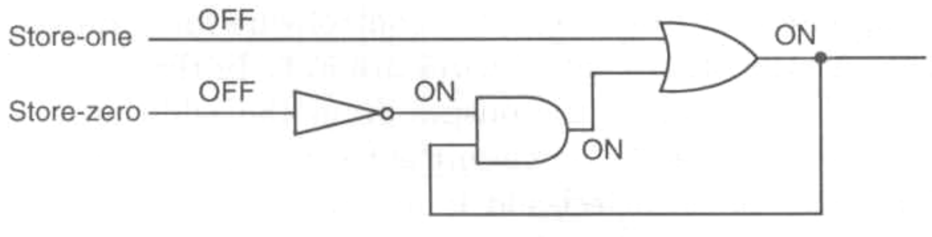
- Build a 1-bit memory cell
- Connect 1-bit cells to the ‘word’ size desired
  - Word = 4 bits in class examples
  - Word = 64 bits on new Pentiums
- Include control lines to each word
  - Select – the location
  - Read/write
- Include data bus connection to each word

1-Bit Memory Circuit

- Memory circuit
  - Two possible *states*, ‘1’ and ‘0’
  - Store or remember the state by maintaining the output at either ‘1’ or ‘0’
  - If we control which value is stored then we have a memory (RAM) circuit
- Trace the behavior of the circuit when...
  - If both inputs are ‘off’ and initial state is ‘off
  - If both inputs are ‘off’ and initial state is ‘on’

1-Bit Memory Circuit



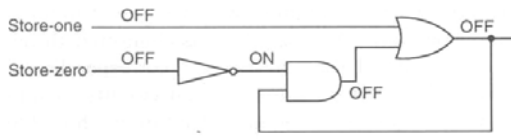


The Concept of Feedback

- The output of a circuit is fed-back to the input
- Important category of circuits for which looking at the inputs *does not* tell you the value of the output
- §2.3, page 58 onward

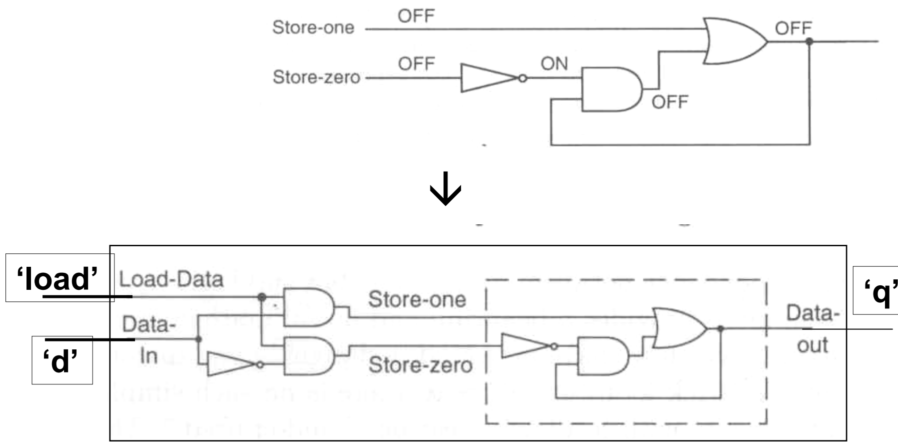
Problems With Simple Memory Circuit

- Building up to our initial memory ‘array’ figures...

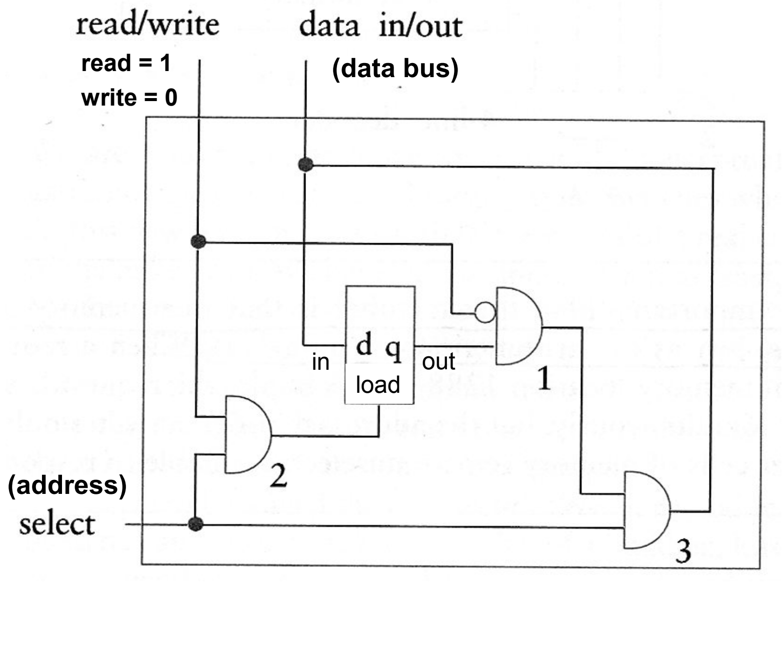


- We do not have control over *when* data is stored or read
  - Each cell must be read or written to *only when* its unique address is selected

1-Bit Memory: Load Control



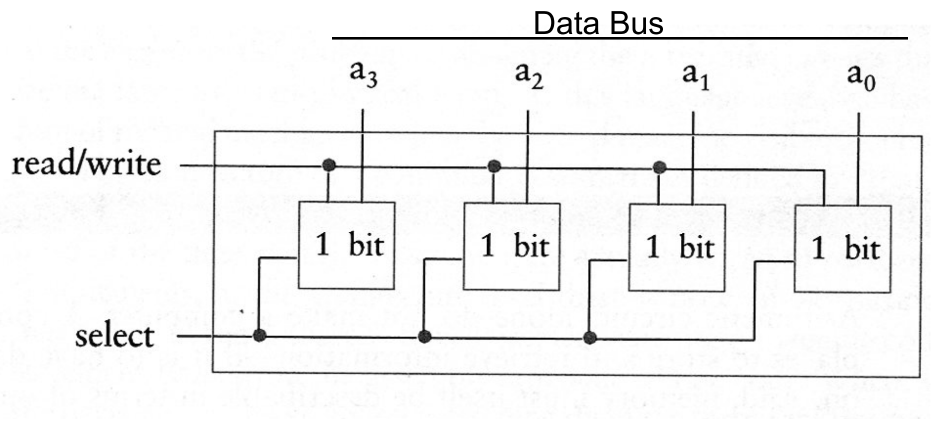
1-Bit Memory: R/W, Select, Data Bus



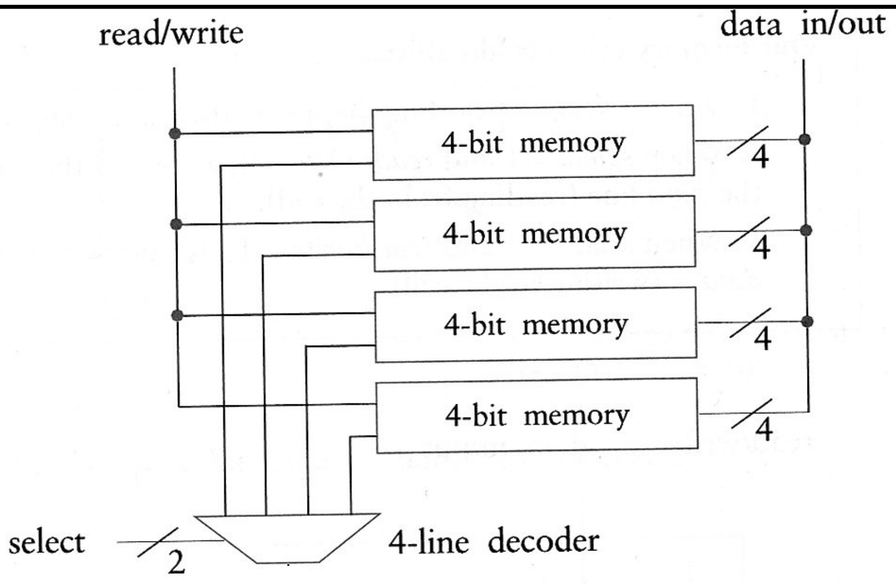
Behavior of 1-Bit Memory Cell

- When *select* = 0, nothing happens (the cell is inactive)
- When *select* = 1 **and** *read/write* = 0, a copy of the stored bit is put on the data bus (reading from the cell)
- When *select* = 1 **and** *read/write* = 1, the data on the data bus is stored into the cell (writing to the cell)

Cascaded 1-Bit Memory Cells



Memory Array



The CPU & ‘Fetch-Execute’

Computer Operation

- Computer *only* execute programs
- Programs are sequences of instructions
- To execute programs the computer
  - **Fetches** the next instruction, and any data needed
  - **Executes** the instruction
- This is the ‘fetch-execute’ cycle

Addition Example

To perform 2 + 3 = 5 :

1. Load ‘2’ into the CPU
2. Add ‘3’ to ‘2’ and temporarily store the result, ‘5’
3. (=) Store the result to main memory

‘Fetch’ each instruction and data, and then do what it directs

Addition Example Con’t

- Keeping everything organized
  - The CPU must know where to find everything
    - Data and instructions
  - The CPU must know where to store the result

Components of Pippin

- Pippin handout
  - ALU
  - Registers
    - PC, the program counter
    - Instruction register
    - Accumulator (the computer’s scratch pad)
  - Decoder
  - MUX
  - RAM

Addition Example Con’t

- Keeping everything organized
  - The CPU must know where to find everything
    - Data and instructions
  - The CPU must know where to store the result
- Performing the example in *binary*!
  - The operations (*e.g.,* ‘ADD’) need a binary number assigned to them
  - The operands (data) must be in binary

Assembly Language

- Our example must be written in simple steps
  - LOD #2 = load #2 into the CPU
  - ADD #3 = add #3 to whatever is there
  - STO Y = store the result to ‘Y’
  - HLT = stop!

TA Times

- TA: Allison Bellew
  - Meet in McConnell 104
  - Sunday 7-9 pm
  - Monday 6-7 pm

Summary

- Memory
  - 1-bit storage & feedback
  - Control lines: R/W and select (address)
  - Data bus
- The CPU
  - The fetch-execute cycle
  - The Pippin CPU simulator