Chapter 2

Selected Elements of Digital Logic

[Reading Only]

KHAN WAHID
2012-13 (Term 1)

UNIVERSITY OF SASKATCHEWAN
College of Engineering

Chapter Objectives

- This chapter covers:
  - Quick review of digit logic components (adder, decoder, MUX, FF, registers, counters, etc.)
    - Most parts of this section are self-study from EE232 or other online Digital logic resources
    - Only the key materials will be reviewed
  - Memory Architecture
    - RAM, Address multiplexing, SRAM, DRAM
    - ROM types, programming

Self Study

- Number system and conversion
  - Binary, Decimal, Hexadecimal, Octal, BCD, code conversion (BCD to Gray, Hex to BCD, etc.

- Hexadecimal Number System
  - Base 16
    - 16 possible symbols
    - 0-9 and A-F (A=10, B=11,…, F=15)
  - Allows for convenient handling of long binary strings

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

Hexadecimal Number System

- Convert from hex to decimal by multiplying each hex digit by its positional weight

  Example: $163_{16} = (?)_{10}$

$$\begin{aligned} 163_{16} &= 1\times(16^2)+6\times(16^1)+3\times(16^0) \\ &= 1\times 256+6\times 16+3\times 1 \\ &= 355_{10} \end{aligned}$$

- Convert from decimal to hex by using the repeated division method used for decimal to binary conversion
  - Divide the decimal number by 16
  - The first remainder is the LSB and the last is the MSB
- How about HEX – BIN?

Hexadecimal Number System

- In Hex, each digit ranges from 0-15, i.e. 4 binary bits are required to represent it

Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Binary	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111

- Example of hex to binary conversion:
  - $(9F2)_{16} = (?)_2 \Longrightarrow (1001\_1111\_0010)_2$
- Convert from binary to hex :
  - By grouping bits in four starting with the LSB
  - Each group is then converted to the hex equivalent
  - Zeros are added to the left of the MSB to fill out the last group
- Example: $(1110100110)_2 = (?)_{16} \Longrightarrow (11\_1010\_0110)_2 = (3A6)_{16}$

Examples: Binary Representation

- 6-bit representation

	+16 Sign Magnitude	-16 Sign Magnitude
Un-signed Magnitude	010000	010000
Signed Magnitude	010000	110000
1's Complement	010000	101111
2's Complement	010000	110000

Q. How to extend the representation for 8-bit?

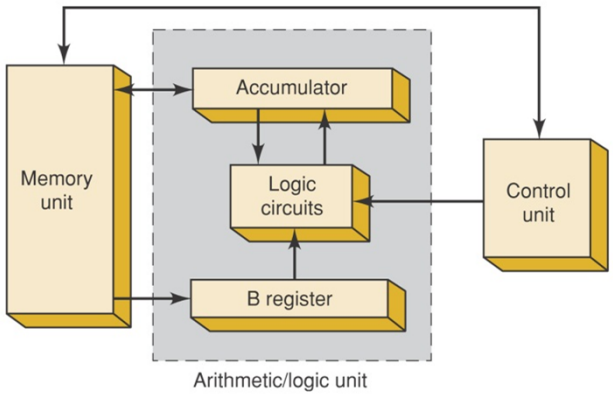
Examples: Binary Representation

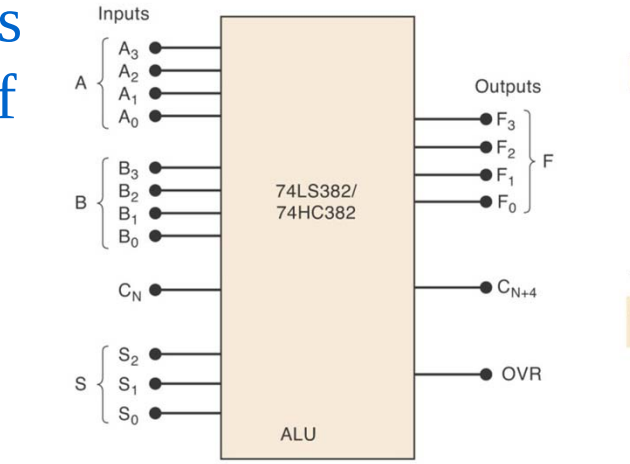
- 8-bit representation

	+115	-115
Un-signed Magnitude	01110011	01110011
Signed Magnitude	01110011	11110011
1's Complement	01110011	10001100
2's Complement	01110011	10001101

Arithmetic/Logic Unit (ALU)

- All arithmetic operations take place in the ALU of a computer





A = 4-bit input number
B = 4-bit input number
C_N = carry into LSB position
S = 3-bit operation select inputs
F = 4-bit output number
C_{N+4} = carry out of MSB position
OVR = overflow indicator

(a)

Function Table

S_2	S_1	S_0	Operation	Comments
0	0	0	CLEAR	$F_3F_2F_1F_0 = 0000$
0	0	1	B minus A	Needs $C_N = 1$
0	1	0	A minus B	Needs $C_N = 1$
0	1	1	A plus B	Needs $C_N = 0$
1	0	0	A ⊕ B	Exclusive-OR
1	0	1	A + B	OR
1	1	0	AB	AND
1	1	1	PRESET	$F_3F_2F_1F_0 = 1111$

Notes: S inputs select operation.
OVR = 1 for signed-number overflow.

(b)

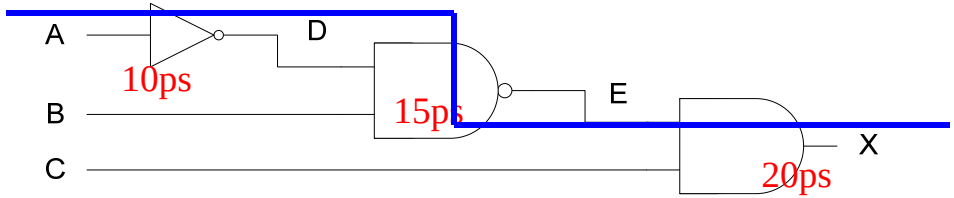
Example: Determine the output for the following sequence: (a) $S_2S_1S_0 = 010$, $A_3A_2A_1A_0 = 0100$, $B_3B_2B_1B_0 = 0001$, $C_N = 1$

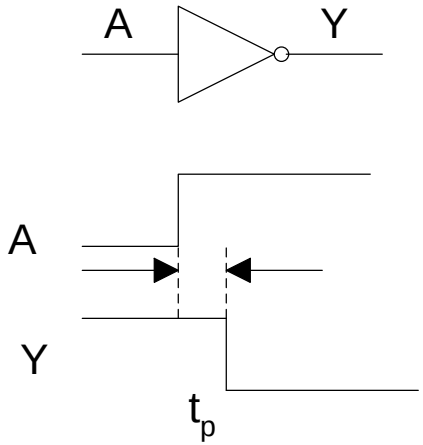
(b) Change select code to 011, and redo

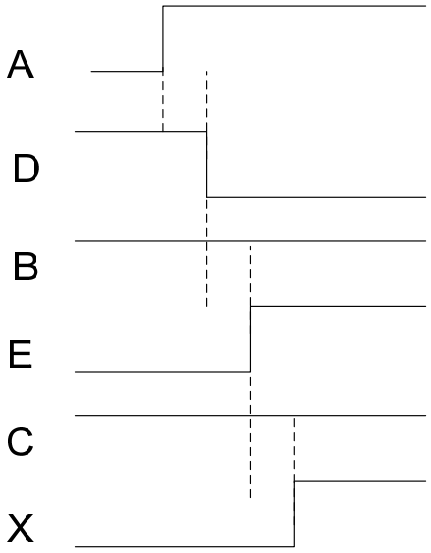
Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

Propagation Delay / Critical Path

- The maximum time when an input changes until the outputs reach their final values - t_p
- Typical values are in the picoseconds (10^{-12}) range







Critical Path: ***The longest delay from input to final output (10 + 15 + 20 = 45ps)***

The significance is that X reacts to the change in A after 45ps

University of Saskatchewan
College of Engineering

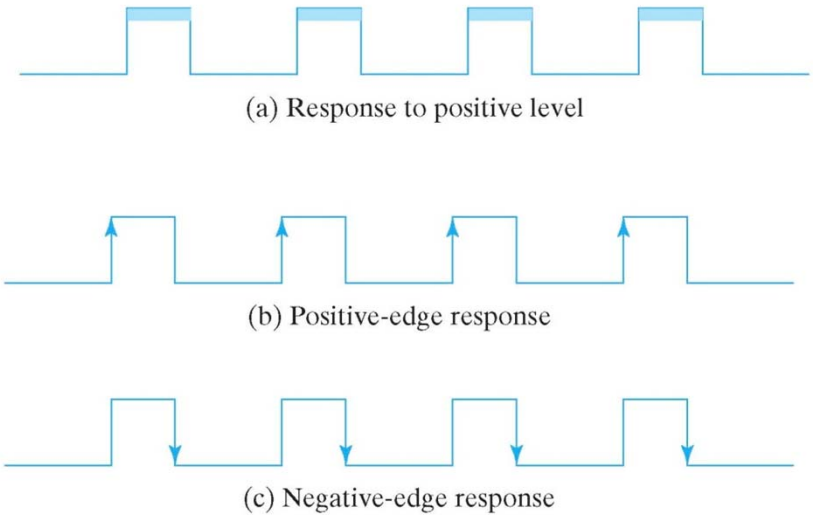
Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

Flip-Flop (or sometimes Latch)

- The flip-flop, abbreviated as FF, is a key memory element (stores 1 bit =>1 FF = 1 bit)
- Sequential Elements:
  - Everything changes in accordance with clock signal level or transitions

- Latch: Storage elements that operate (change output levels) with signal levels

- Flip-Flop: Storage elements that operate with clock (or signal) transitions

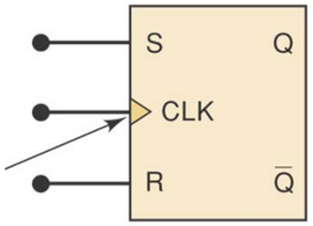


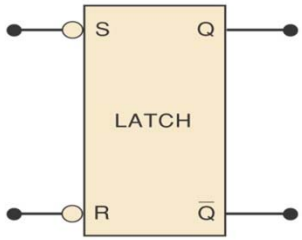
Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

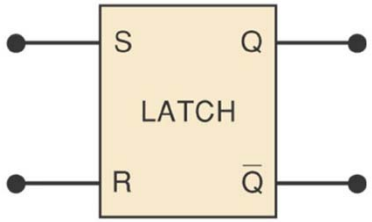
Latch Structure

SET = 1 => Q = 1

RESET = 1 => Q = 0



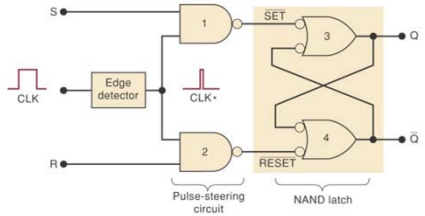




a) Clocked S-R latch

Inputs			Output
S	R	CLK	Q
0	0	↑	Q_0 (no change)
1	0	↑	1
0	1	↑	0
1	1	↑	Ambiguous

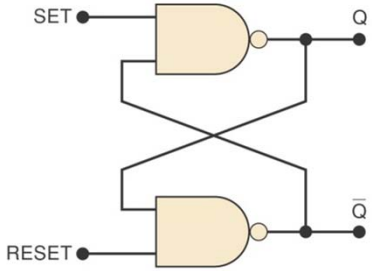
Q_0 is output level prior to ↑ of CLK.
↓ of CLK produces no change in Q.



b) Basic S-R Latch

Set	Reset	Output
1	1	No change
0	1	Q = 1
1	0	Q = 0
0	0	Invalid*

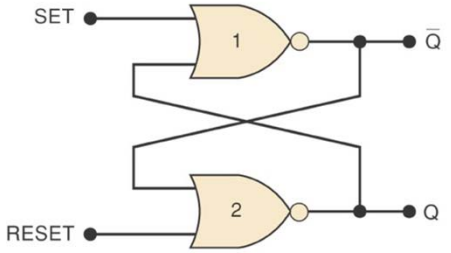
*Produces $Q = \overline{Q} = 1$.



c) Basic S-R Latch

Set	Reset	Output
0	0	No change
1	0	Q = 1
0	1	Q = 0
1	1	Invalid*

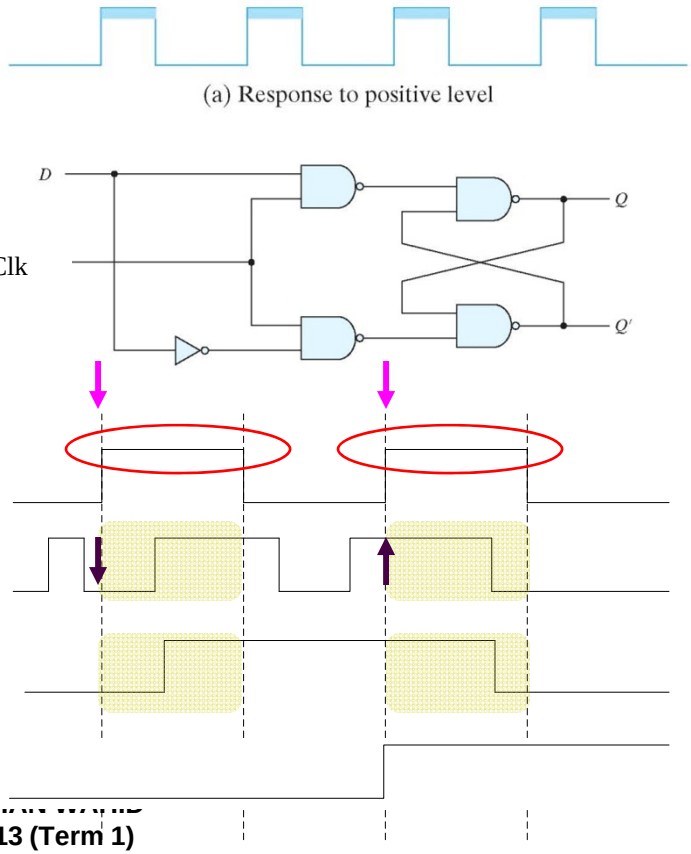
*Produces $Q = \overline{Q} = 0$.

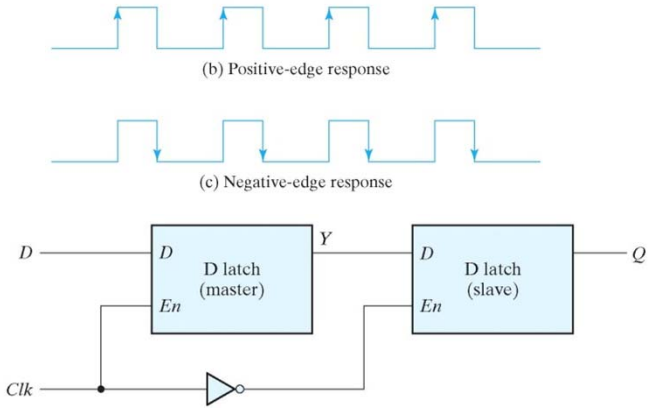


Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

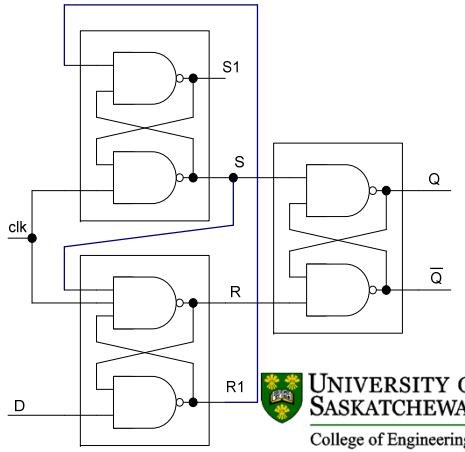
Latch vs. Flip-Flop

- Latch: Level sensitive
- Flip-Flop: Edge sensitive



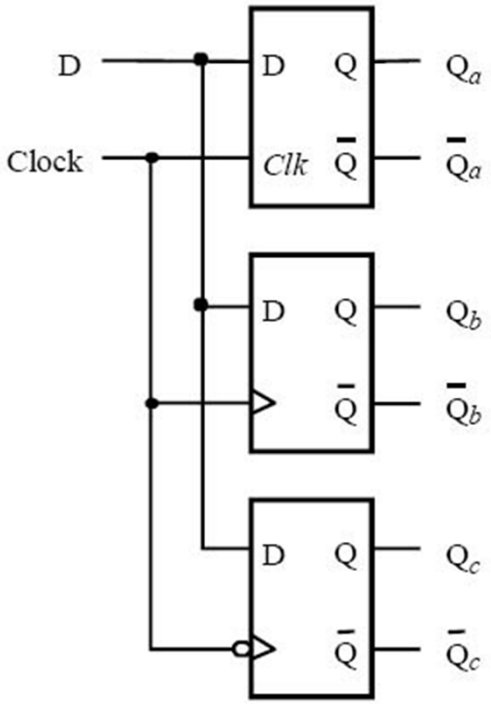


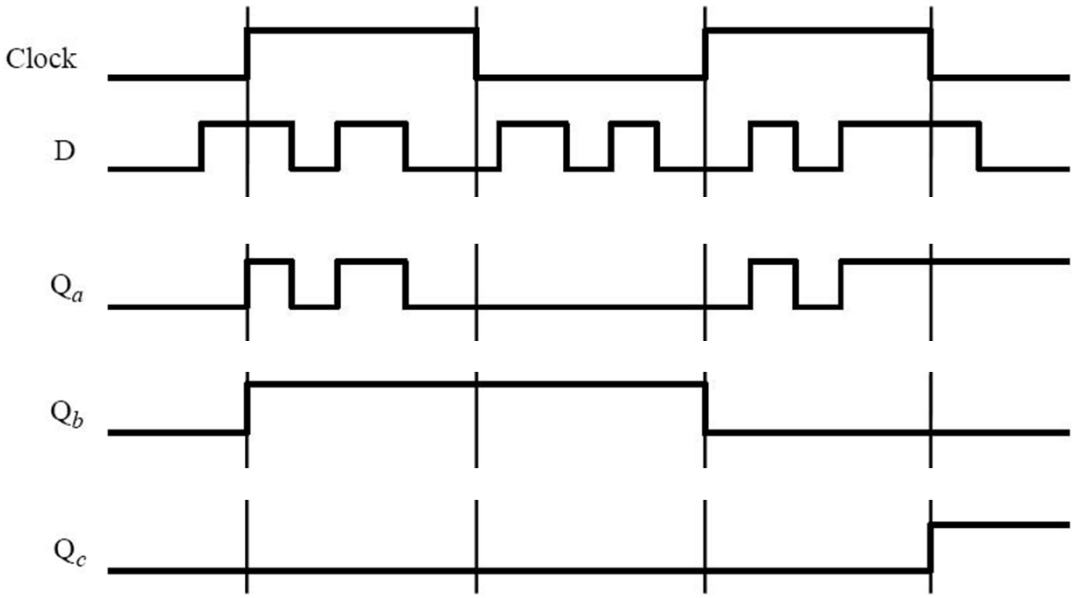
Or



UNIVERSITY OF SASKATCHEWAN
College of Engineering

Compare: Level vs. Edge

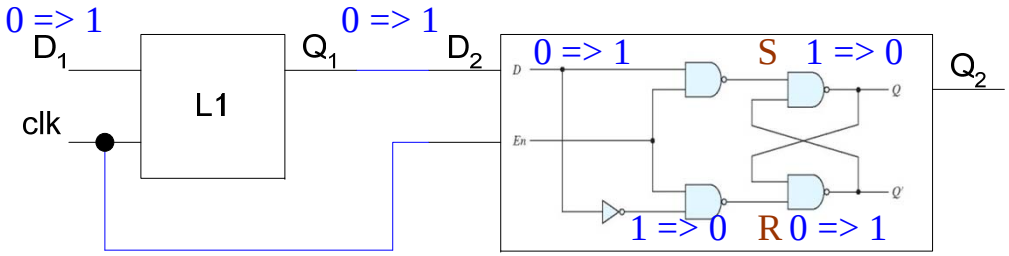


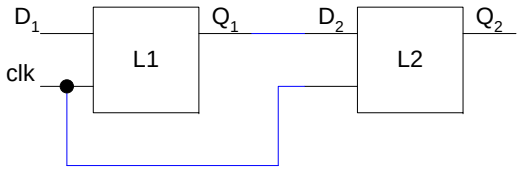


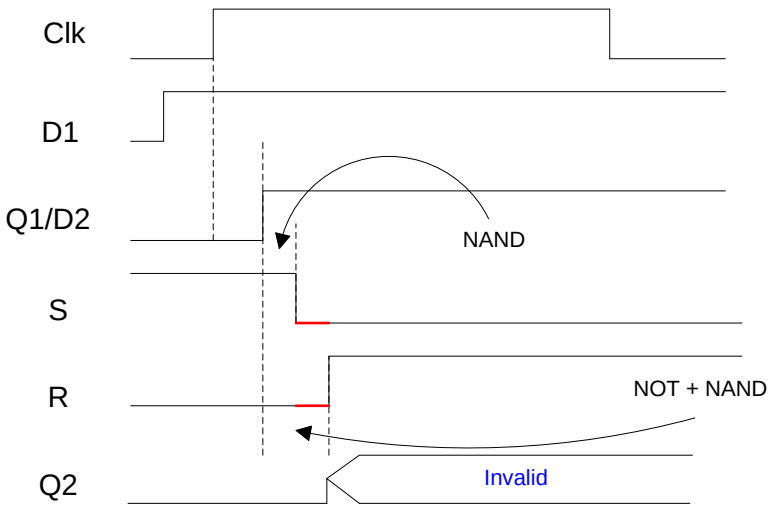
Source: Fundamentals of Digital Logic with Verilog Design, Brown and Vranesic, 2E, ISBN: 9780073380339

Problems with Latches (level sensitive)

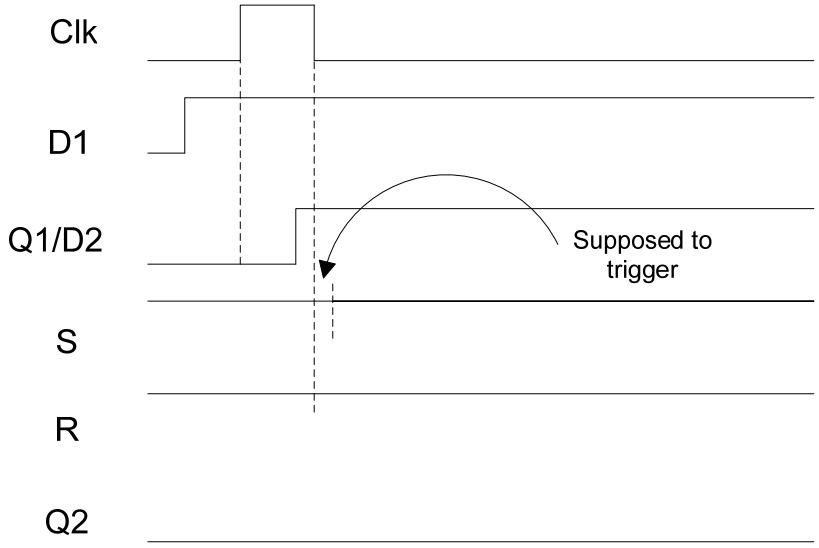
- Example: Two D-Latches are in series





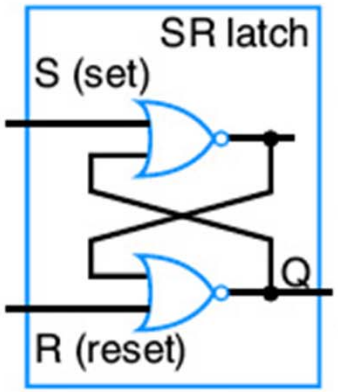


Clk pulse is too wide



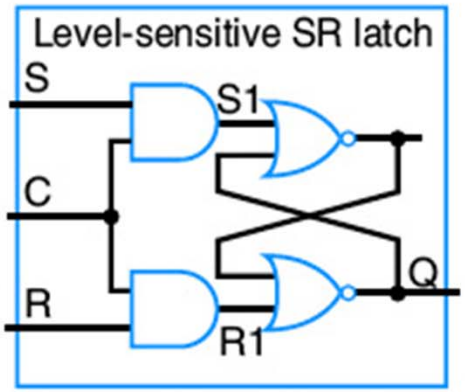
Clk pulse is too narrow

Increasingly better bit storage: SR Latch to Edge-triggered DFF



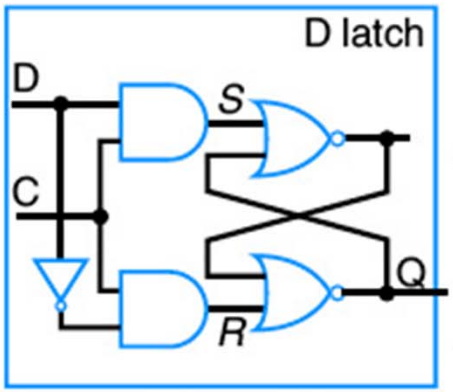
Feature: S=1 sets Q to 1, R=1 resets Q to 0.

Problem: SR=11 yield undefined Q.



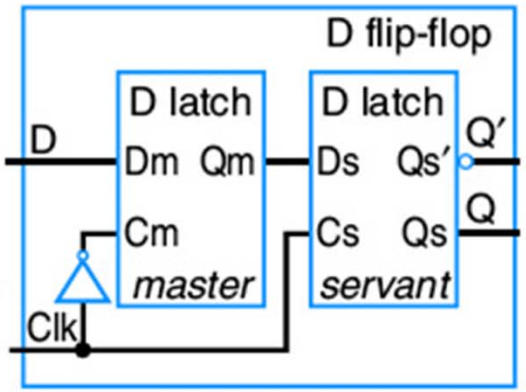
Feature: S and R only have effect when C=1. We can design outside circuit so SR=11 never happens when C=1.

Problem: avoiding SR=11 can be a burden.



Feature: SR can't be 11 if D is stable before and while C=1, and will be 11 for only a brief glitch even if D changes while C=1.

Problem: C=1 too long propagates new values through too many latches; too short may not enable a store.



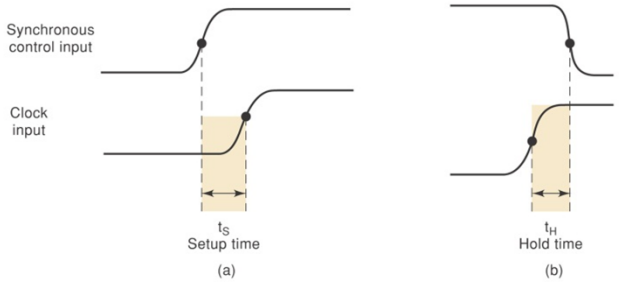
Feature: Only loads D value present at rising clock edge, so values can't propagate to other flip-flops during same clock cycle. *Tradeoff*: uses more gates internally than D latch, and requires more external gates than SR– but gate count is less of an issue today.

Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

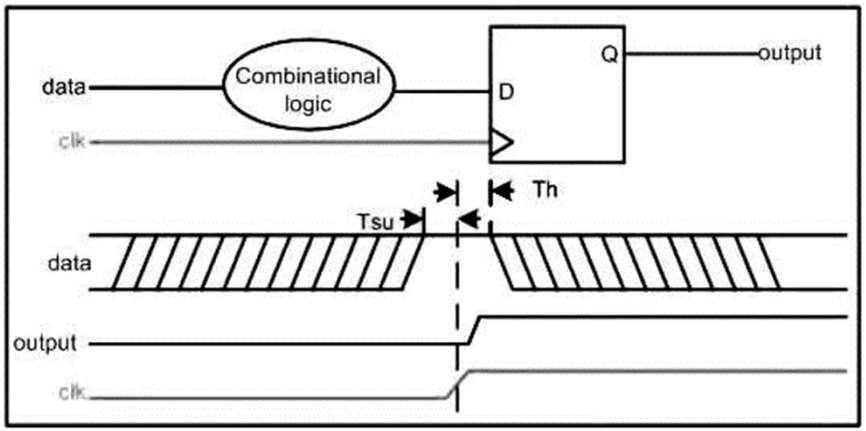
Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

FF Timing Considerations

- FF are built from gates and wires – they will have delays. Thus a real FF imposes some restrictions on when the FF's input can change relative to CLK so that correct operation is ensured



- Setup Time: D input must be stable before the clock edge (typ: 3ns) – wait for D input
- Hold Time: D input must be held stable after the clock edge (typ: 2ns) – wait for CLK

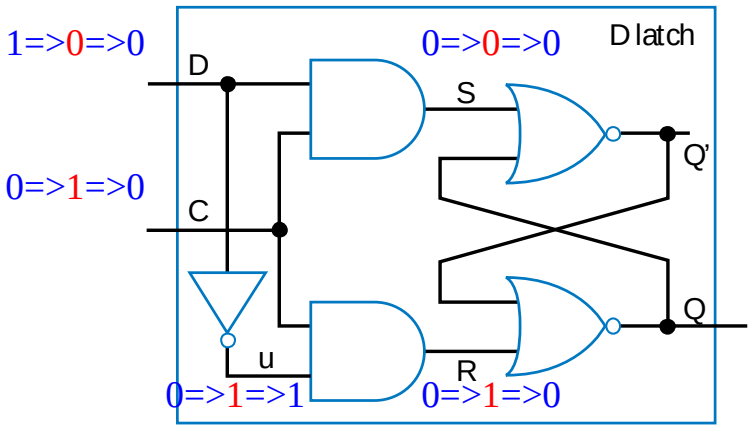


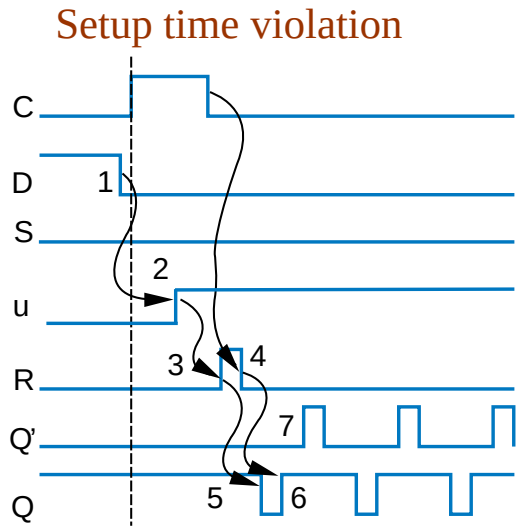
Critical Window = T_{su} + T_h

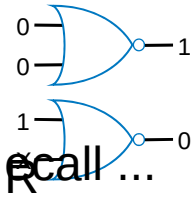
Metastability – How?

Set	Reset	Output
0	0	No change
1	0	Q = 1
0	1	Q = 0
1	1	Invalid*

*Produces Q = $\bar{Q}$ = 0.







Leads to oscillation!

Figure 3.61 Setup time violation: D changed to 0 (1) too close to the rising clock. u changed to 1 after the inverter delay (2), and then R changed to 1 after the AND gate delay (3). But then the clock pulse was over, causing R to change back to 0 (4) before a stable feedback situation with Q=0 occurred in the cross-coupled NOR gates. R's change to 1 did cause Q to change to 0 after the NOR gate delay (5), but R's change back to 0 caused Q to change right back to 1 (6). The glitch of a 0 on Q fed back into the top NOR gate, causing Q' to glitch to 1 (7). That glitch of a 1 fed back to the bottom NOR gate, causing another glitch of a 0 on Q. That glitch runs around the cross-coupled NOR gate circuit (oscillation)—a race condition would eventually cause Q to settle to 1 or 0, or possibly enter a metastable state (to be discussed).

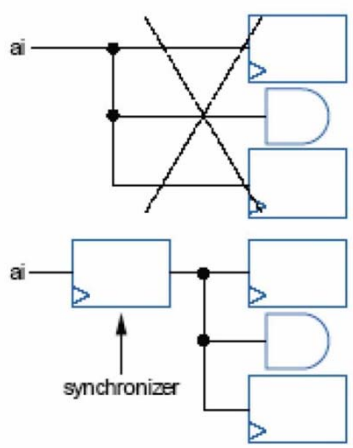
Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

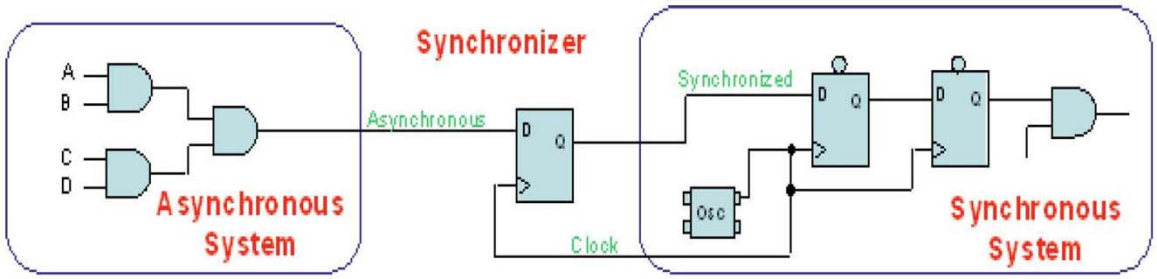
Metastability – When?

- If the circuit violets setup and hold times of a FF, the result is that the FF enters in a metastable state (other than stable 1 or stable 0 – sometimes referred as quasi-stable state)
- Cases when metastability occurs:
  - When a FF has an input that is not synchronized with the clock – asynchronous input
  - When the clock's rise and fall time are too long (i.e. more than the tolerable values)
  - When interfacing two domains are operating at two different frequencies or at the same frequency but with different phase
  - When the combinational delay causes the flip-flop's data input to change inside the critical window

Metastability – How to Prevent?

- Make sure that the clock period is long enough to allow for the resolution of quasi-stable states and for the delay of whatever logic may be in the path to the next flip-flop
- Add one or more successive synchronizing flip-flops as synchronizer which allows for an entire clock period (except for the setup time of the second flip-flop) for metastable events in the first synchronizing flip-flop to resolve themselves





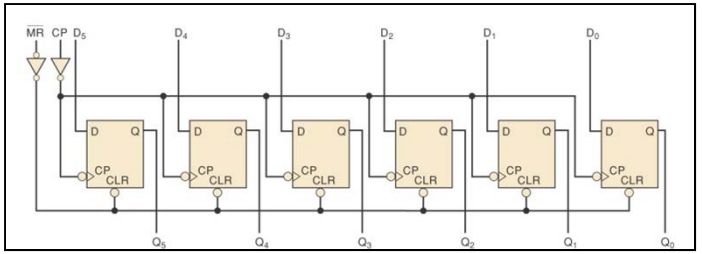
e.g., Other digital device

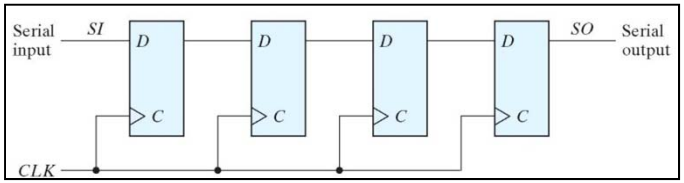
e.g., Microprocessor

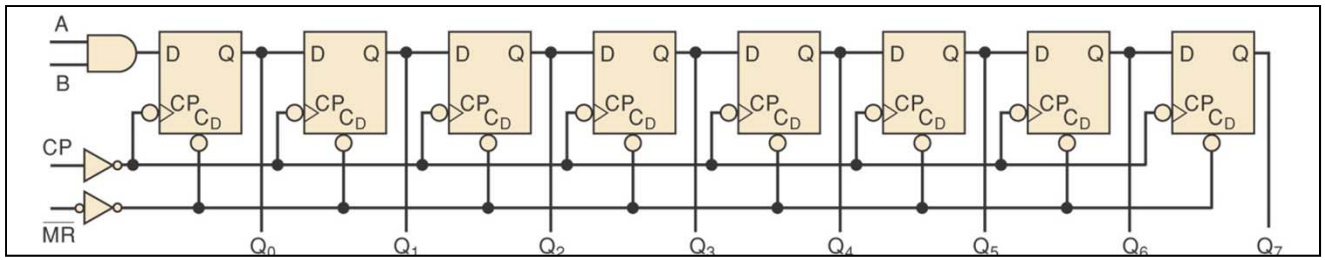
University of Saskatchewan
College of Engineering

Registers

- Registers are sets of flip-flops
  - Classified by the way data are entered and outputted
    - Parallel in/parallel out (PIPO)
    - Serial in/serial out (SISO)
    - Parallel in/serial out (PISO)
    - Serial in/parallel out (SIPO)



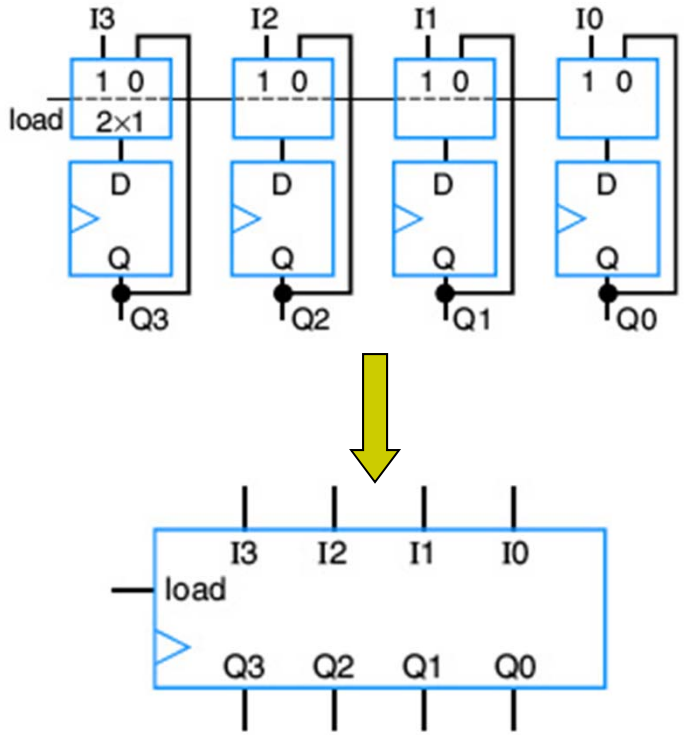




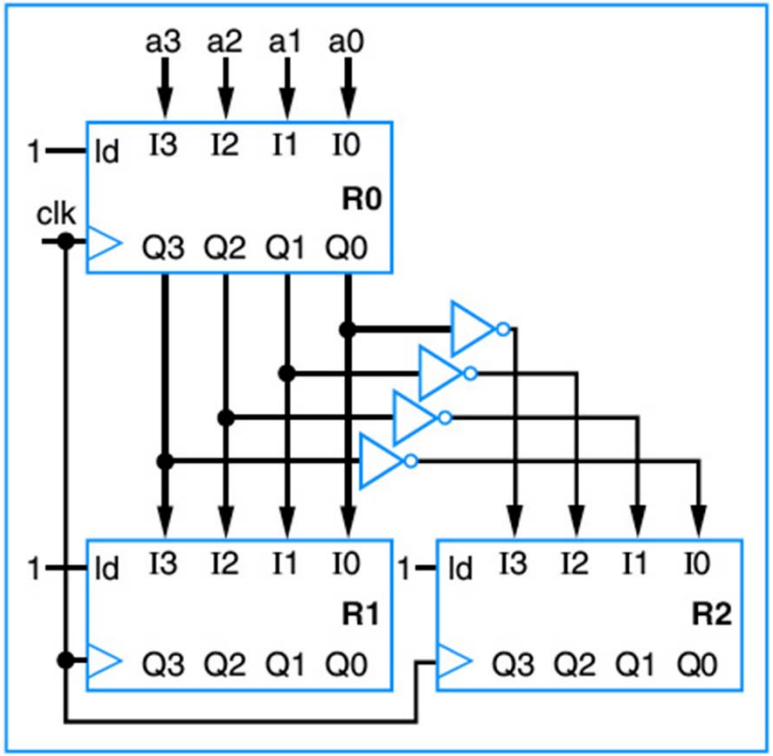
Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

Register with Parallel Load



load = 1 => parallel load @ CP
load = 0 => hold current (no CP)

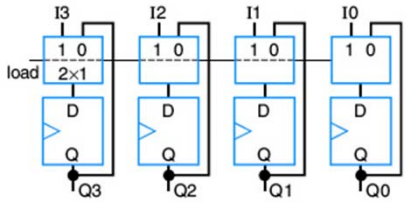


Example: Suppose the initial values of {a3,a2,a1,a0} are given. Find out the contents of R0, R1, and R3.

University of Saskatchewan
College of Engineering

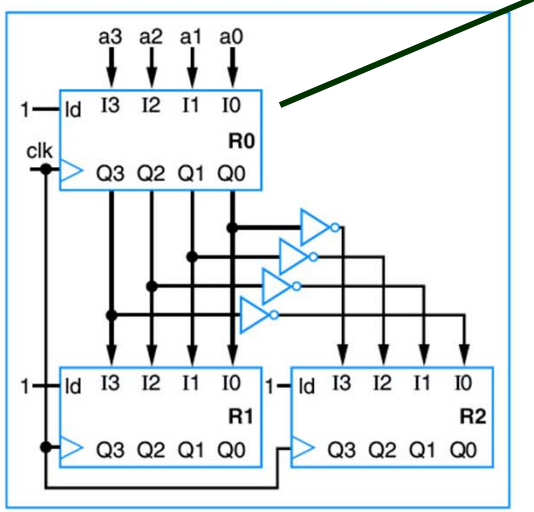
Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

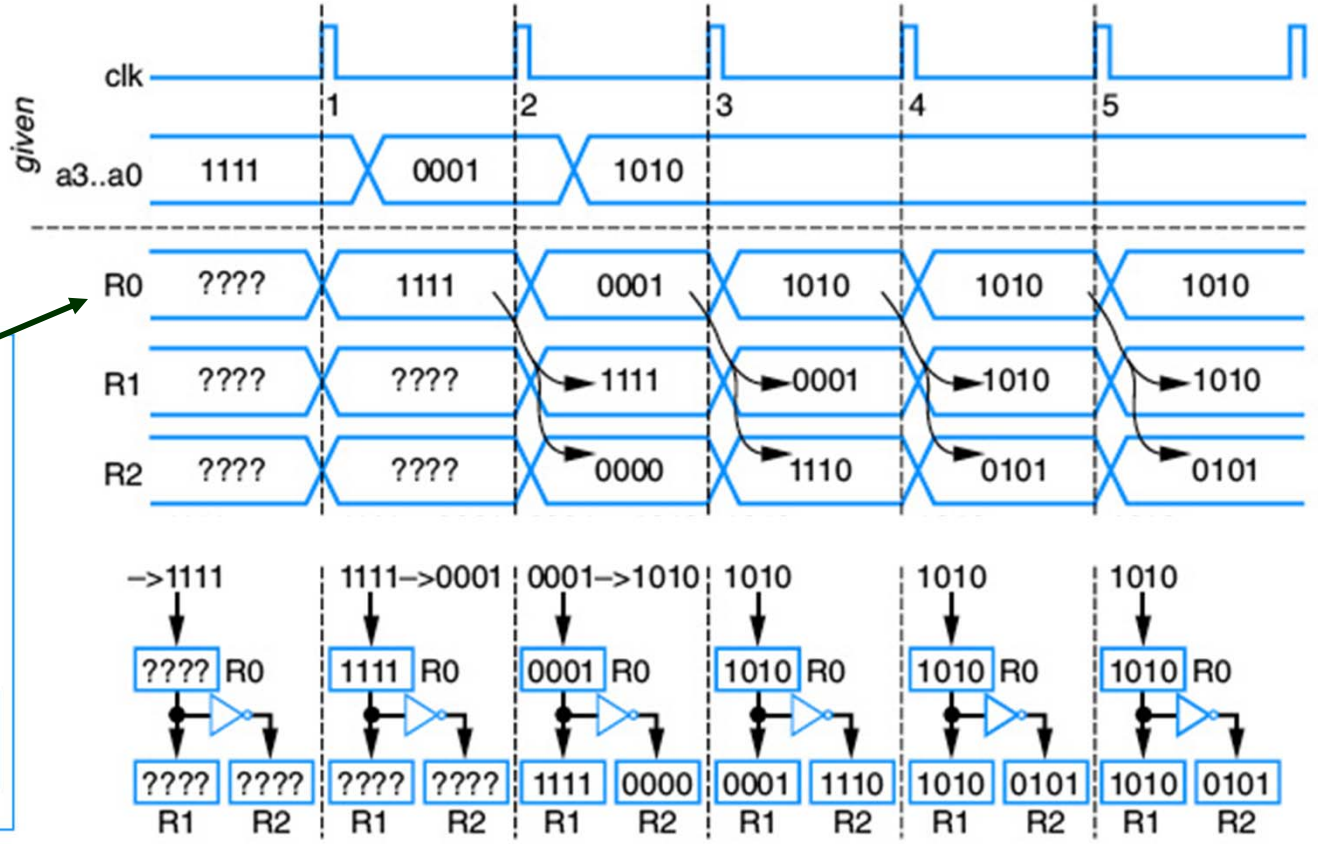
Register with Parallel Load



Note that:

1. ld = 1 (for all)
2. R2 = R1'
3. Pay attention to R0 and R1





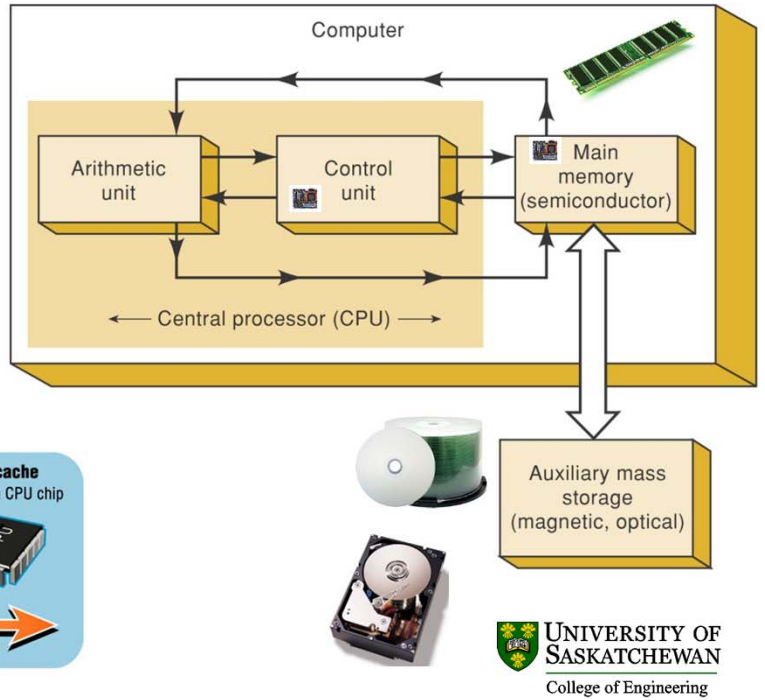
Important to note how the data are **latched** into registers at different clock edges – useful later in "**pipelining**"

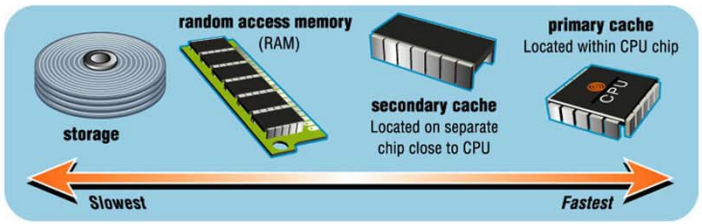
University of Saskatchewan
College of Engineering

Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

Memory Devices

- Memory: A device to which binary information is transferred for temporary storage and from which information is retrieved when needed for processing
- Commonly used memory devices:
  - Flip flops
  - Registers
  - RAM, ROM, etc.
- Cache memory:
  - Level 1 (L1), Level 2 (L2)
  - High-speed, small size





University of Saskatchewan
College of Engineering

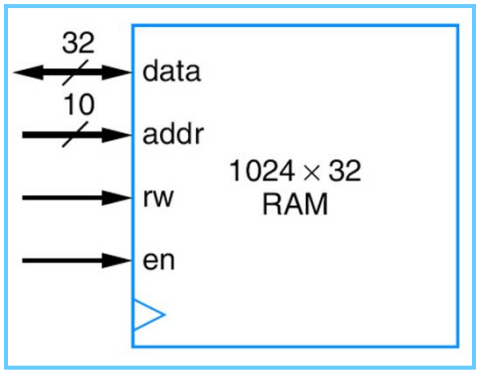
Memory Terminology

- Memory cell: An electrical circuit used to store a single bit
- Memory word: A group of bits
  - 1 byte = 8 bits; 1 nibble = 4 bits
- Capacity: Size of the memory cell
  - **16 x 4** => **16 words**, each word of **4 bits** => 64 bits = 8 bytes
  - 32 x 4 vs. 16 x 8
  - Density ⇔ Capacity
- Address: Each word in memory is assigned an identification number, called an address (for 64x4, how many address bits?)
- Read: Transfers binary information from a specific address out of memory
- Write: Transfers binary information into a specific address of memory

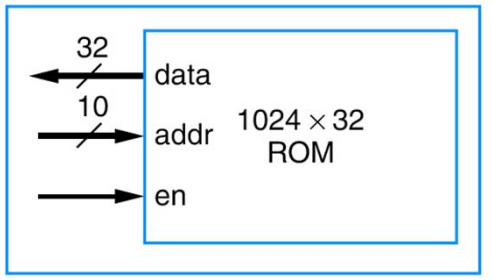
Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

RAM vs. ROM

- Random Access Memory (RAM):
  - The architecture of the memory is such that information can be retrieved from any locations (or *addresses*) with the same time
  - Performs both *read* and *write*



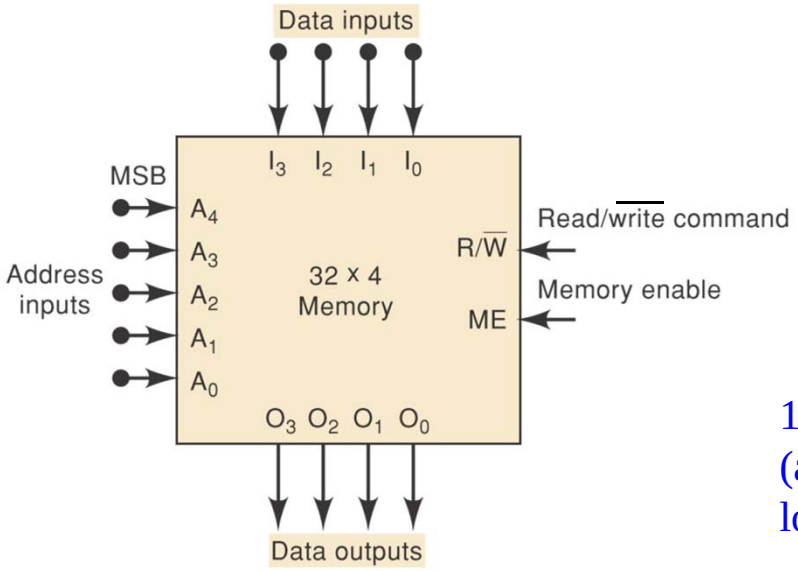
- Read Only Memory (ROM):
  - A memory device where permanent binary information is stored (used for storing program)
  - Once written, information cannot be altered by *write* operation
  - Performs only *read* operation



Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

A Typical Memory Cell



Control Inputs to Memory Chip

Memory Enable	Read/$\overline{\text{Write}}$	Memory Operation
0	X	None
1	0	Write to selected word
1	1	Read from selected word

Each bit is one register

Memory address		Memory content
Binary	Decimal	
0000000000	0	1011010101011101
0000000001	1	1010101110001001
0000000010	2	0000110101000110
⋮	⋮	⋮
1111111101	1021	1001110100010100
1111111110	1022	0000110100011110
1111111111	1023	1101111000100101

1024 words (addresses or locations)

16 bits data (contents)

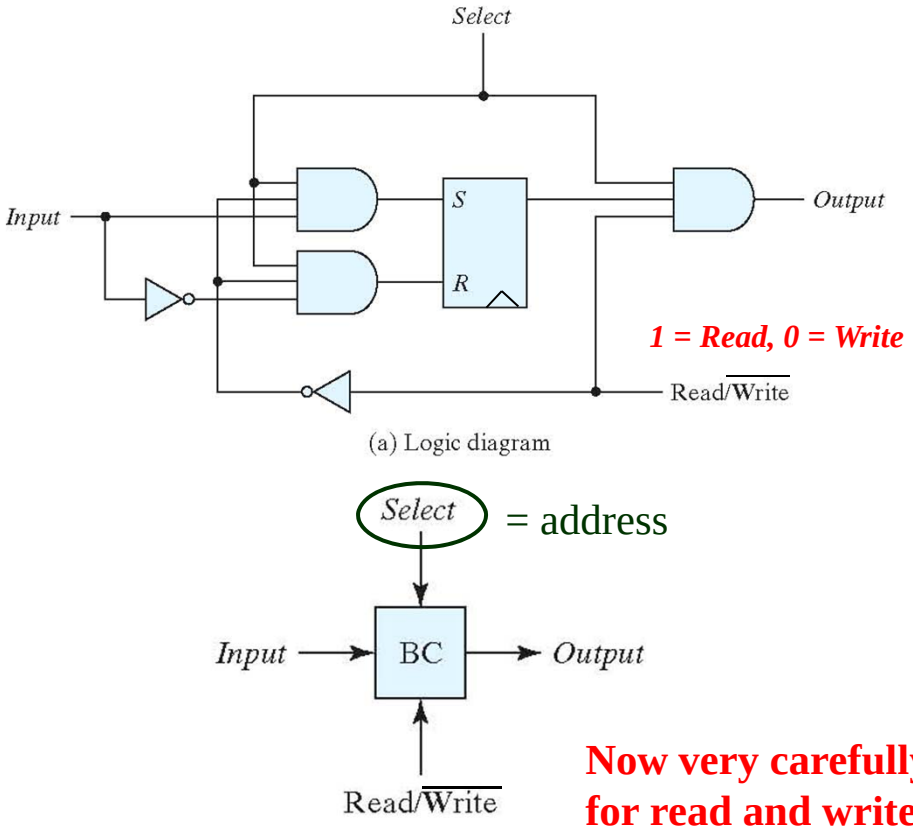
1024 x 16 Memory

University of Saskatchewan
College of Engineering

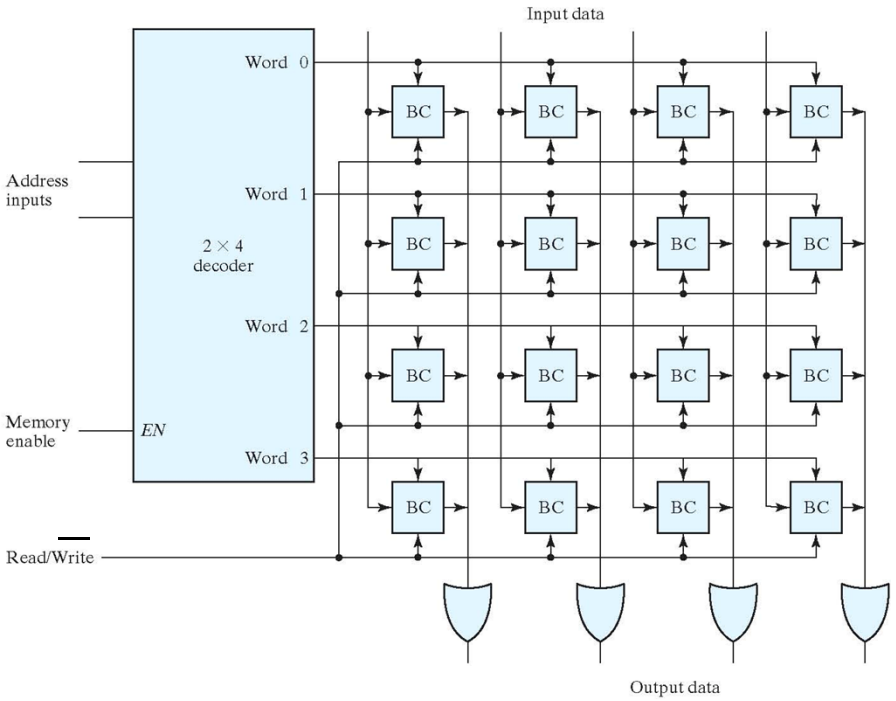
Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

A Typical Memory Cell

- 1 cell = 1 bit = 1 register = 1 FF (S-R or D)



1 = Read, 0 = Write



Now very carefully note the operation for read and write – w.r.t. clock pulse??

4x4 RAM

University of Saskatchewan
College of Engineering

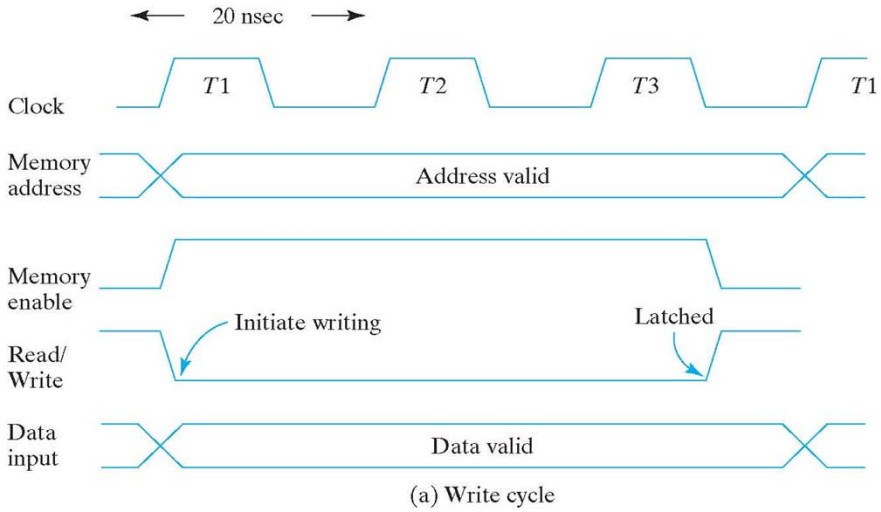
Memory Timing Waveforms

- The operation of the memory unit is controlled by the CPU (synchronized by CPU's own internal clock)
  - No internal clock in memory – read and write operations are specified by control inputs (from the CPU)
  - The CPU provides these control signals in such a way to synchronize its internal clocked operations with the read and write memory operations
  - Access time:
    - Time required to **select a word and READ it**
  - Cycle time:
    - Time required to complete a **WRITE operation**

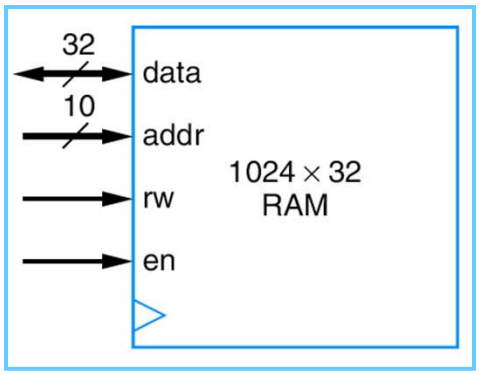
Access time (READ) < Cycle time (WRITE)

Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

Memory Timing Waveforms



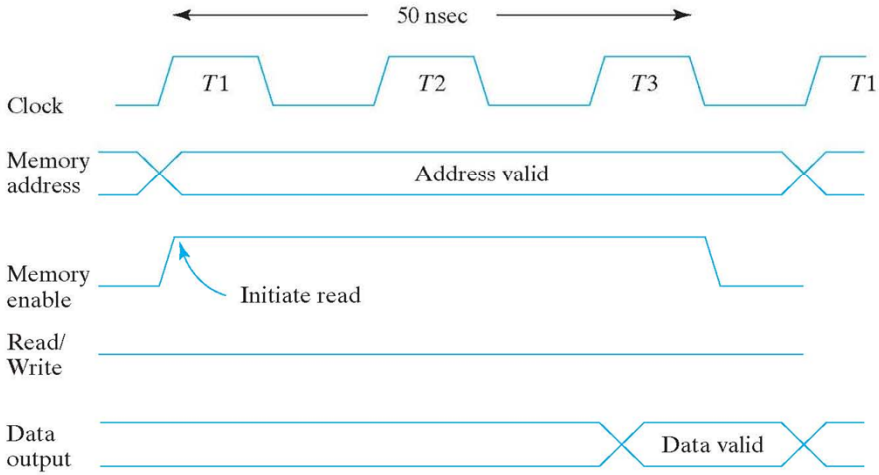
(a) Write cycle



- Address and Data must be held constant until Data is latched

- Write operation needs CP to write data

- Read operation does not need CP to read data



(b) Read cycle

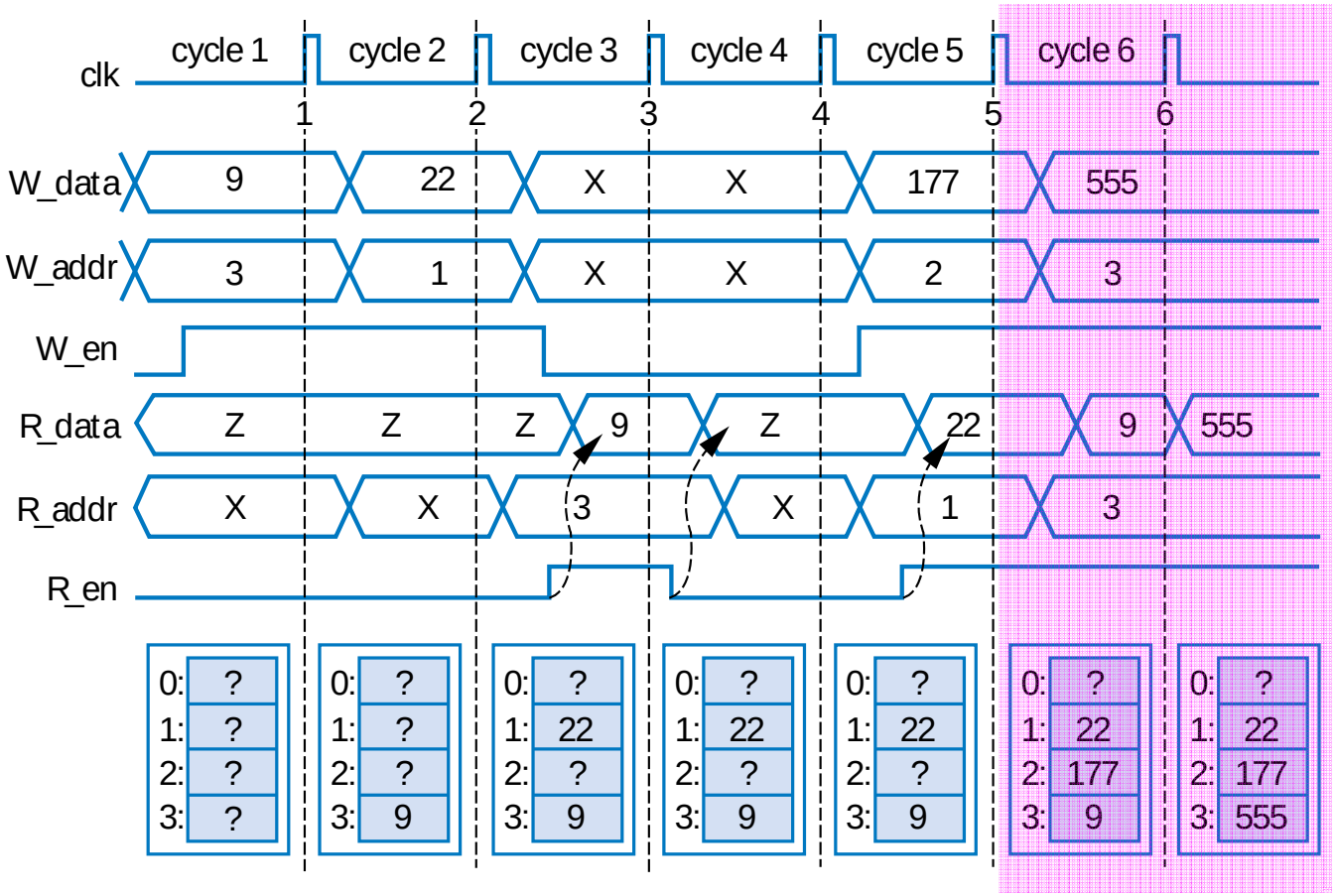
Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

Memory: Reading and Writing

Write operation needs CP to write data

Read operation does not need CP to read data

****Dual-port RAM** => **can read and write simultaneously**



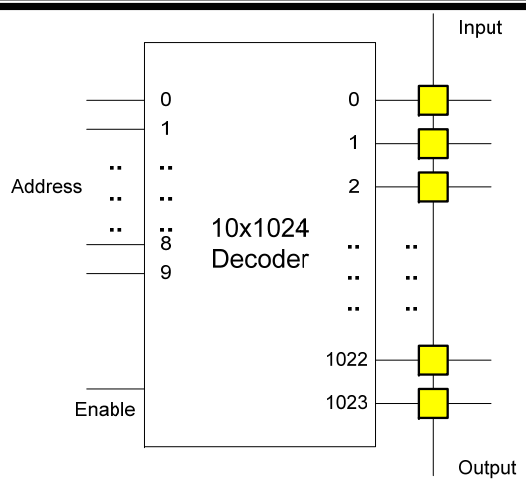
Reading may be done in the middle of a CP (instant)

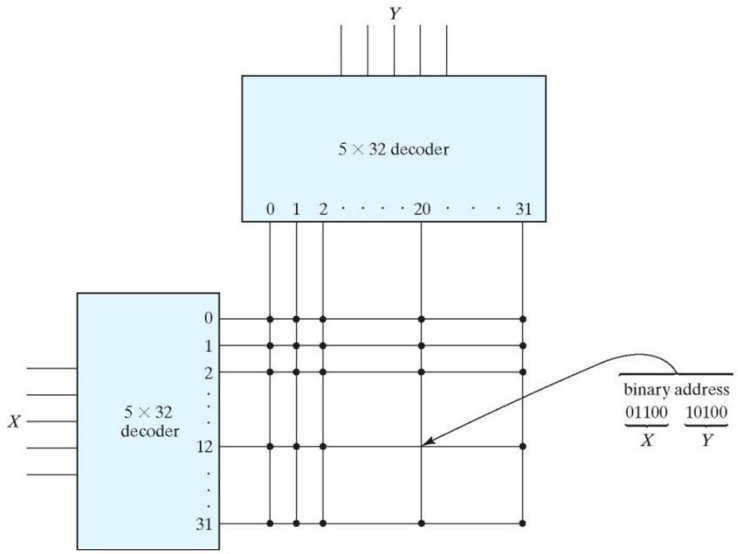
University of Saskatchewan
College of Engineering

Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

Address Multiplexing

- What about 1K x 1 (1024x1) memory?
- To decode the addresses, we require 10 bits
  - One 10x1024 decoder?
  - 1024 = 32 x 32
    - Rearrange 1024 memory cells as 32 x 32 2-D array
  - Need just Two 5x32 decoders?
- Address multiplexing
  - Two-dimensional decoding

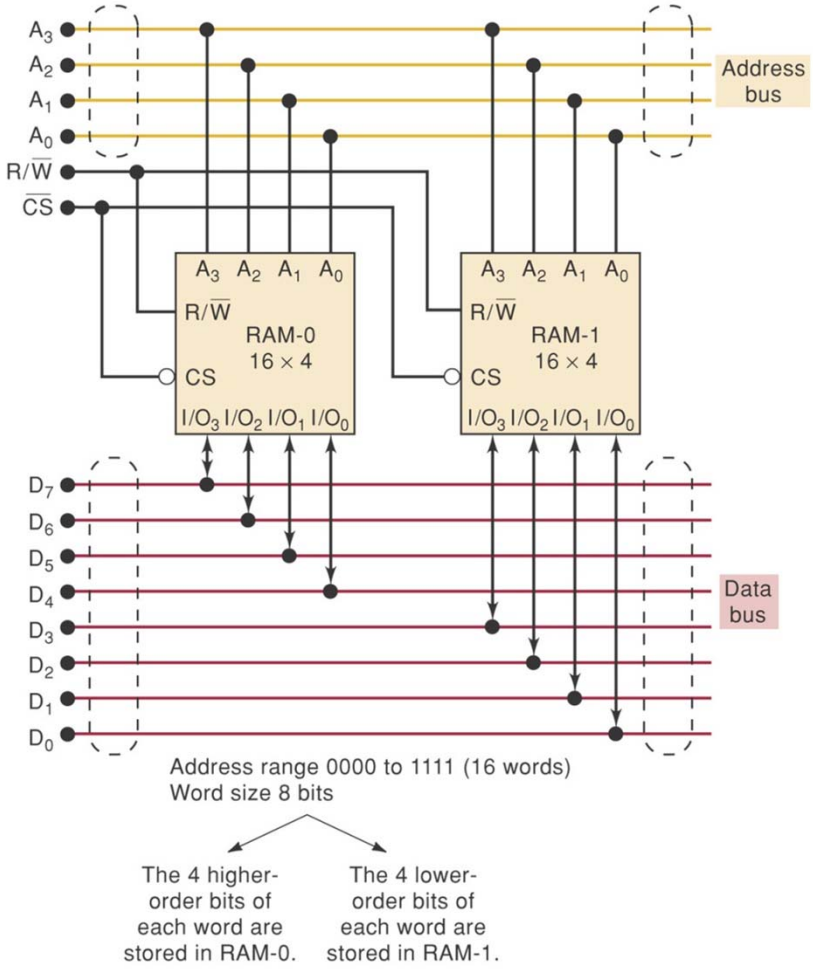




Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

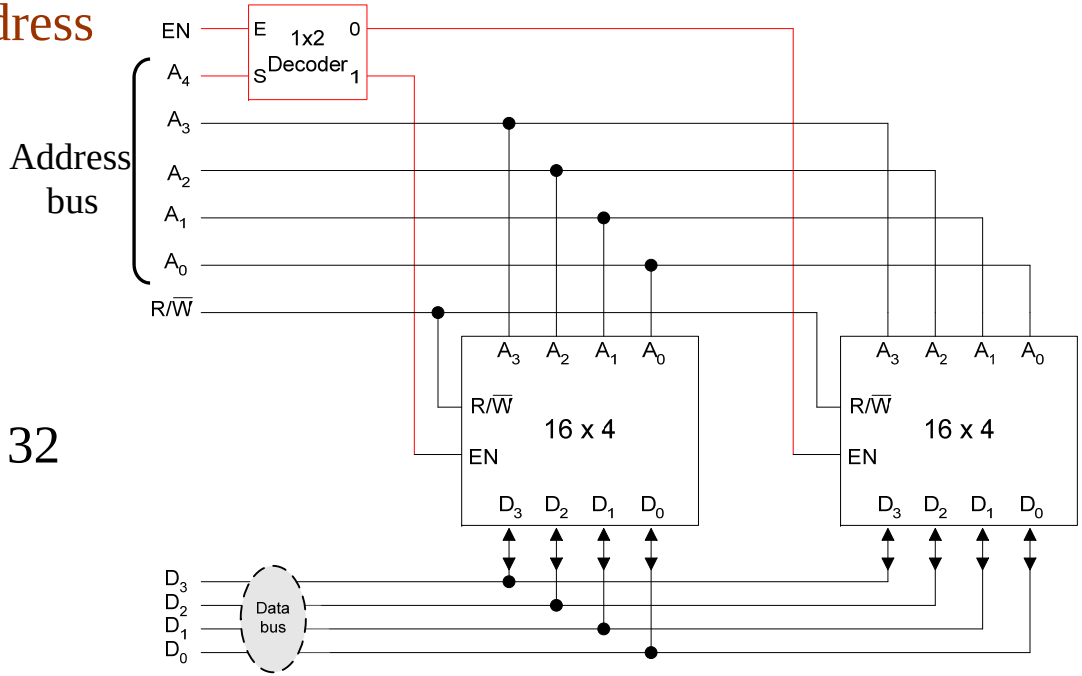
Expanding Capacity – by Word Size

- Expanding word size
- Combining RAMs
  - Two 16x4 RAM modules are combined for one 16x8 module (bit concatenation)
- What about one 32x4 module from 16x4 modules?



Expanding Capacity – by Address

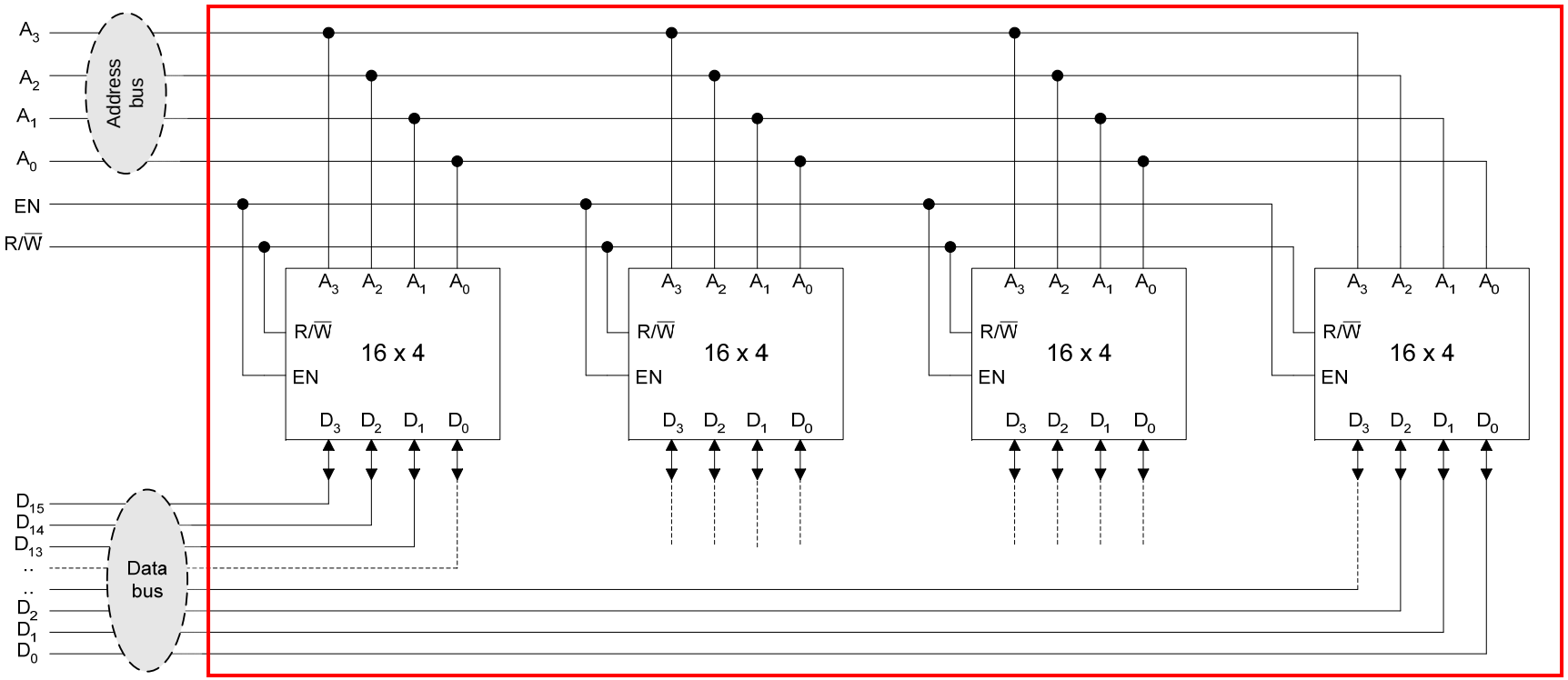
- Expanding addresses (by address multiplexing)
- Combining RAMs
  - Two 16x4 RAMs are combined for one 32x4 module
  - 5 address lines => 2^5 => 32 addresses
  - Here EN = CS
- Now design one **64x16** module using **16x4** RAM modules



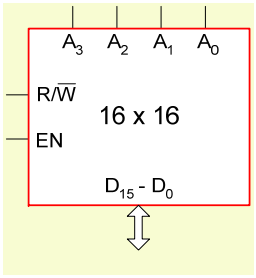
**Note that one 32x4 module has the same capacity as one 16x8 module (as size of both: 128 kbits = 16 bytes). However, they have entirely different functionality and are used for different purposes

University of Saskatchewan
College of Engineering

64x16 Module using 16x4 Modules

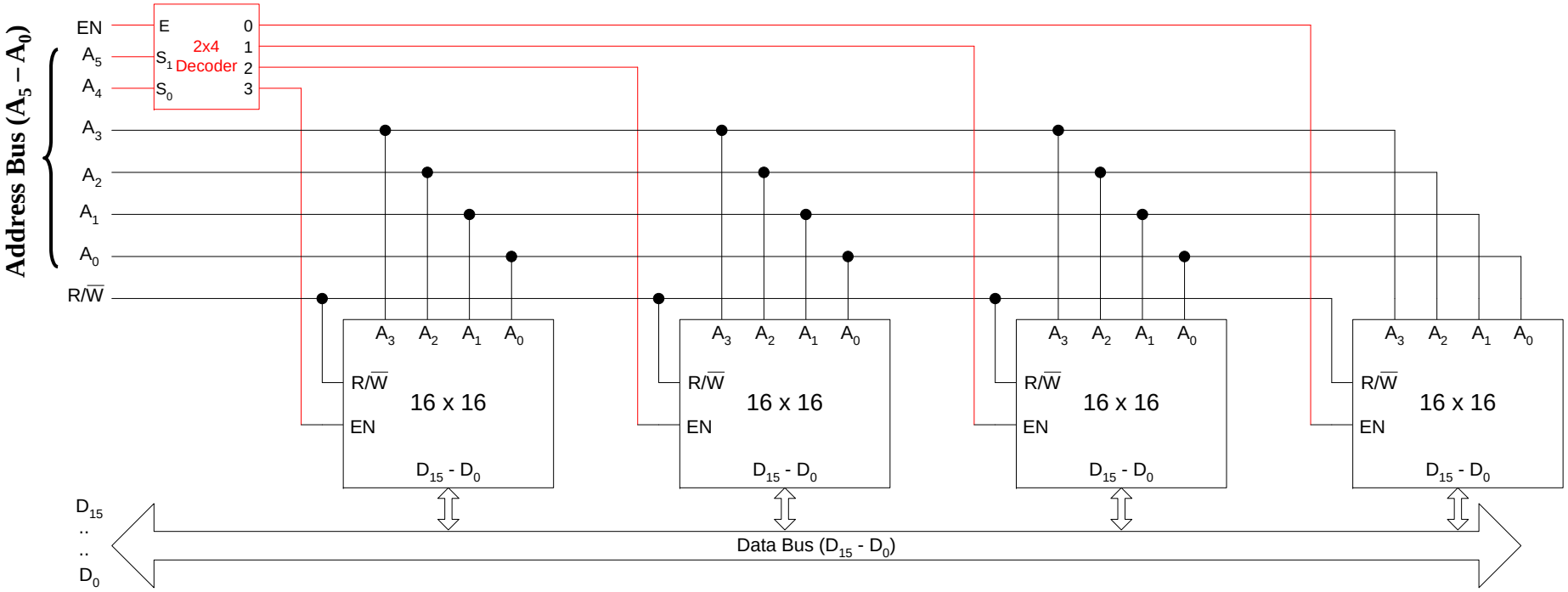


First, design a 16x16 module by increasing the word size



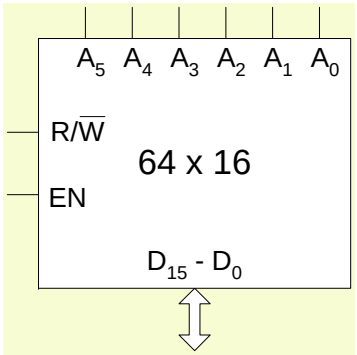
University of Saskatchewan
College of Engineering

64x16 Module using 16x4 Modules



Now, multiplex the previously designed "16x16 module" to increase the address lines using one 2x4 decoder

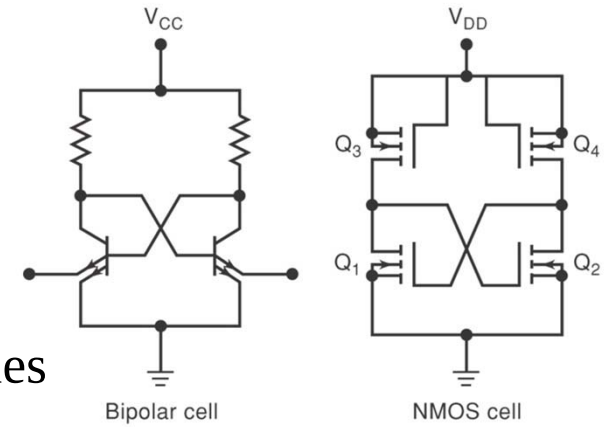
So we have 6 address lines => 2^6 => 64 addresses



Source: Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793

S-RAM

- Static Memory Devices:
  - Semiconductor memory in which the stored data will remain permanently stored as long as power is applied
  - No refreshing (periodically rewriting the data into the memory)
  - Requires more transistors
  - BJT
    - High speed but very complex design
  - MOS
    - Low power consumption, higher capacities



Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

S-RAM – Operation

- S-RAM Operation - Writing
  - 6 transistors are required
  - *word enable* input comes from decoder
  - When *we* = 0, value *d* loops around inverters; that loop is where a bit stays stored
  - When *we* = 1, the *data* bit ("1" or "0") 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

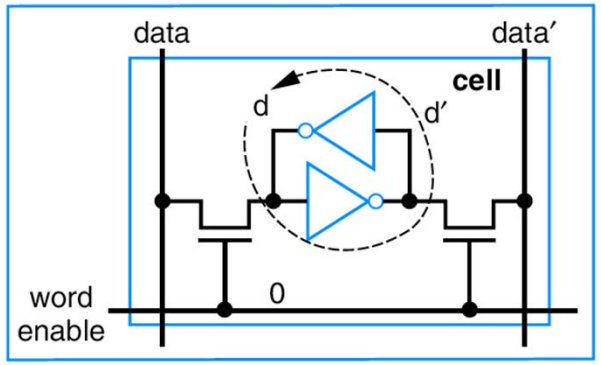


Figure 5.55 SRAM cell.

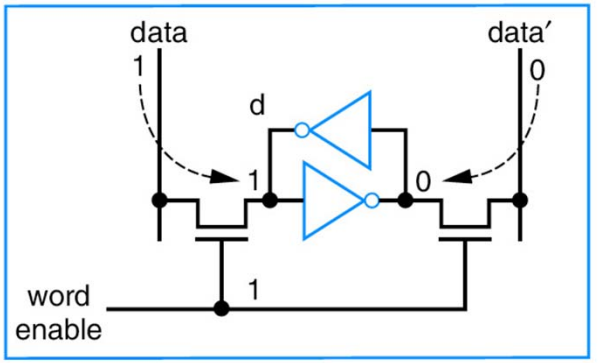


Figure 5.56 Writing a 1 to an SRAM cell.

Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

S-RAM – Operation

- S-RAM Operation - Reading
  - Somewhat complicated and tricky
  - When *r/w* 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 line down slightly below 1 depending on what is stored (here, right line is pulled down)
  - "Sense amplifiers" detect which side is slightly pulled down
    - The electrical description of SRAM is somewhat complex

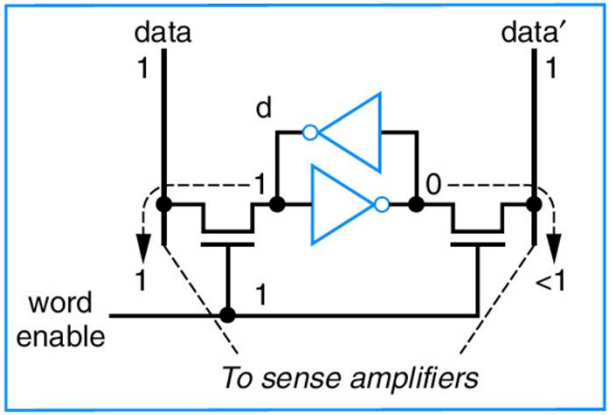
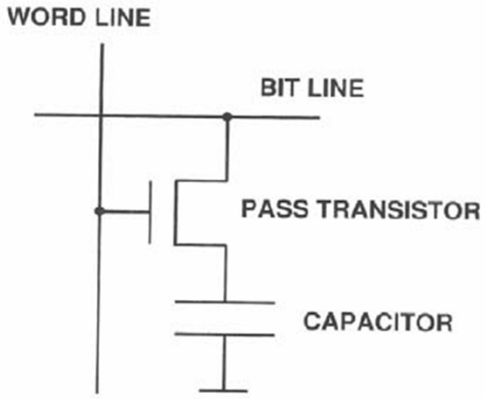


Figure 5.57 Reading an SRAM.

D-RAM

- Dynamic Memory Devices:
  - Semiconductor memory in which the stored data will not remain permanently stored, even with power applied
  - Needs refreshing (periodically rewritten into memory)
  - In modern chips, common refresh rate: 7.8 µsec, 15.6 µsec, 31.2 µsec, 64 µsec, 128 µsec, or Auto
    - Low power consumption, higher capacities
    - Requires less transistors
    - Moderate speed
    - Needs refreshing
    - DRAM is slower than SRAM



Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

D-RAM – Operation

- D-RAM – Writing / Reading
  - 1 transistor (rather than 6 in SRAM)
  - Relies on *large* capacitor to store bit
    - Write: Transistor conducts, data voltage level gets stored on top plate of capacitor
    - Read: Just read the value of *d*
    - Problem:
      - Capacitor discharges over time
      - Must "refresh" regularly, by reading *d* and then writing it right back

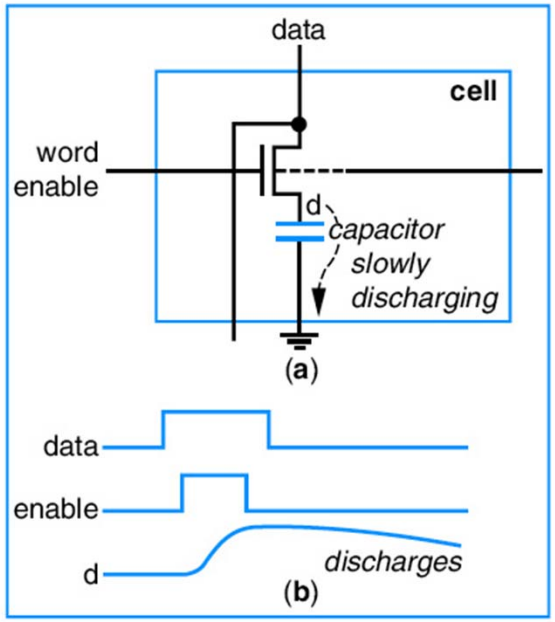


Figure 5.58 DRAM bit storage (a) bit storage block, (b) discharge.

More info: http://en.wikipedia.org/wiki/Dynamic_random_access_memory

D-RAM Types

- Memory modules
- FPM (Fast Page Mode) DRAM
- EDO (Extended Data Output) DRAM
- SDRAM (Synchronous DRAM)
- DDRSDRAM (Double Data Rate SDRAM)
- SLDRAM (Synchronous Link DRAM)
- DRDRAM (Direct Rambus DRAM)

Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

RAM Comparisons

- Register file
  - Fastest
  - But biggest size
- SRAM
  - Fast
  - More compact than register file
- DRAM
  - Slowest
    - And refreshing takes time
  - But very compact, widely used

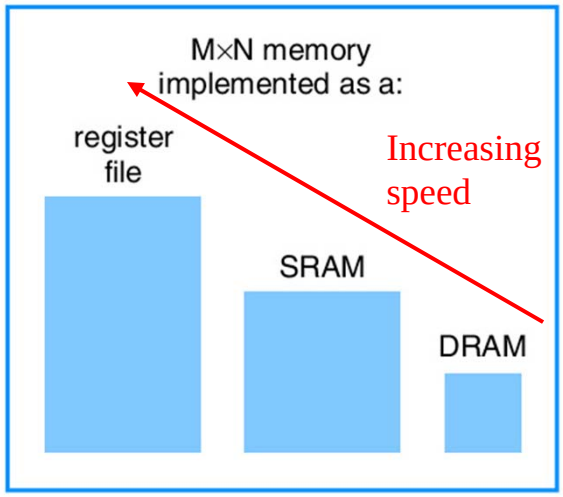


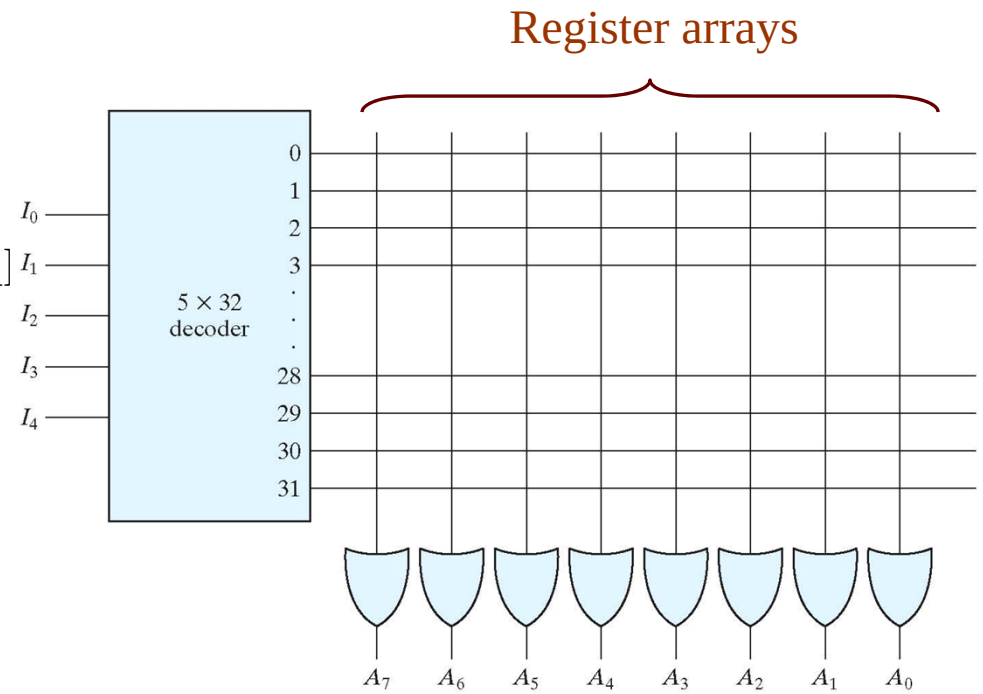
Figure 5.59 Depiction of compactness benefits of SRAM and DRAM (not to scale).

- 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

Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

ROM Architecture

- Three basic parts
  - Register array – stores data programmed into the ROM
  - Address decoders – determines which register will be enabled by row and column
  - Output buffers - pass data to the external data outputs
  - Internal logical structure similar to RAM, without the data input lines
    - No Input buffers



This ROM is not programmed yet

University of Saskatchewan
College of Engineering

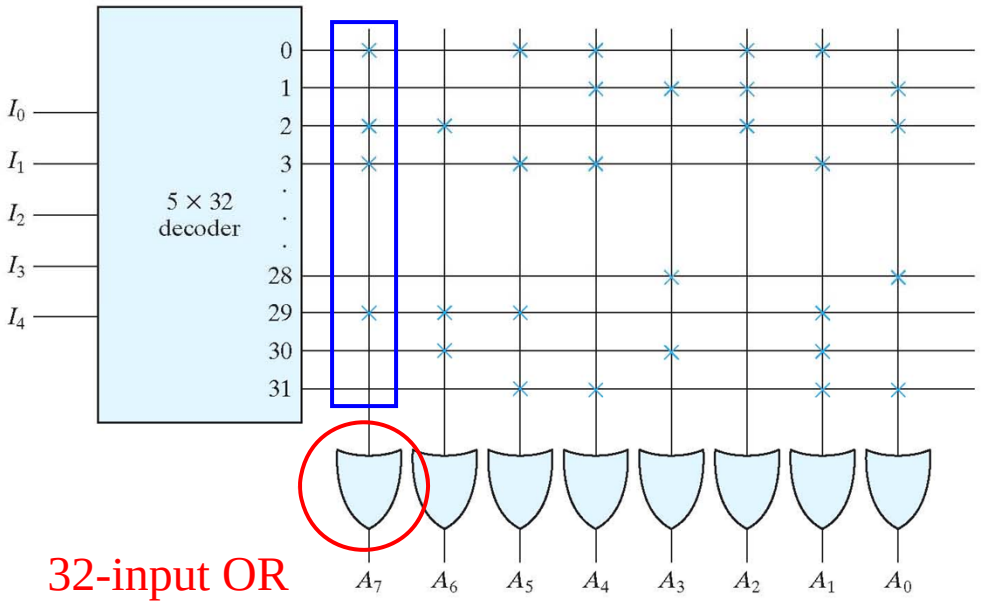
Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

Programming a ROM

ROM Truth Table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
		⋮							⋮			
1	1	1	0	0	0	0	0	0	1	0	0	1
1	1	1	0	1	1	1	1	0	0	0	1	0
1	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	1	1	0	0	1	1	0	0	1	1

Storing bits in a ROM known as programming



32-input OR

Several programming methods

- Mask-programmed ROM (MROM)
- One-Time Programmable ROM (OTPROM)
- Erasable PROM (EPROM)
- Electrically EPROM (EEPROM)

ROM Types

Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

Mask-programmed ROM

- Bits are hardwired as 0s or 1s during chip manufacturing
- **Once programmed, can never be altered**
  - 2-bit word on the right stores "10"
- Word enable (from decoder) simply passes the hardwired value through transistor
- The most compact of any ROM type
- Best suited for high-volume well-established products

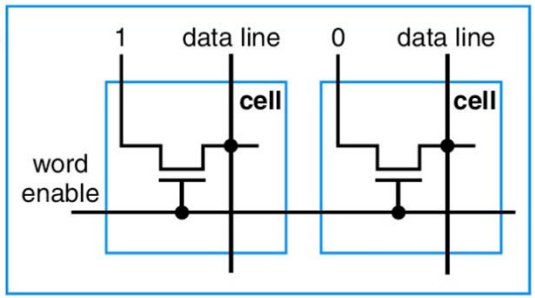
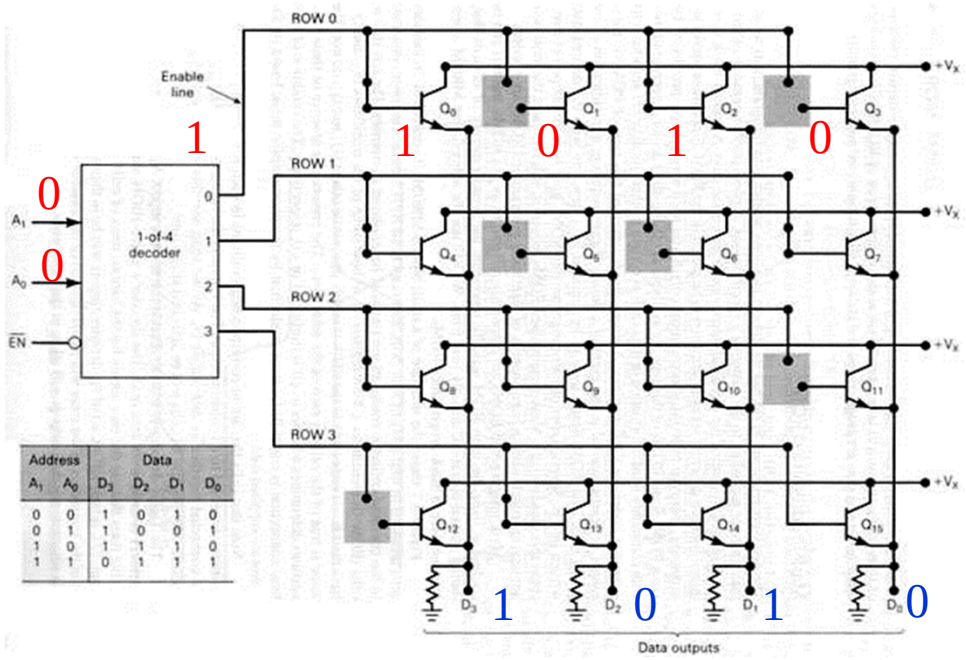


Figure 5.66 Mask-programmed ROM cells: left cell programmed with 1, right cell with 0.



Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

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)
    - Blown cells will be read as 0s (involves some special electronics)
    - Unblown cells will be read as 1s
    - 2-bit word on the right stores "10"
  - Also known as **One-Time Programmable (OTP) ROM**
    - **Once programmed, can never be altered**

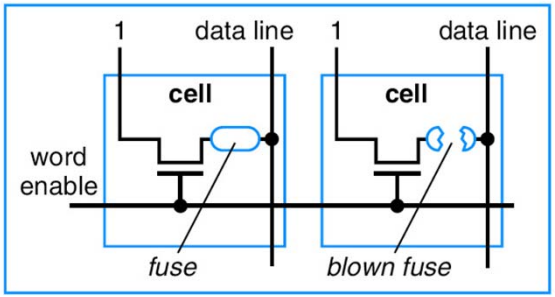


Figure 5.67 Fuse-based ROM cells: left cell programmed with 1, right cell with 0.

Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

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
    - Trapped cell stores 0
    - Non-trapped cell stores 1
      - 2-bit word on the right stores "10"
  - To erase, shine ultraviolet light onto chip
    - Gives trapped electrons energy to escape
    - Requires chip package to have a **window**

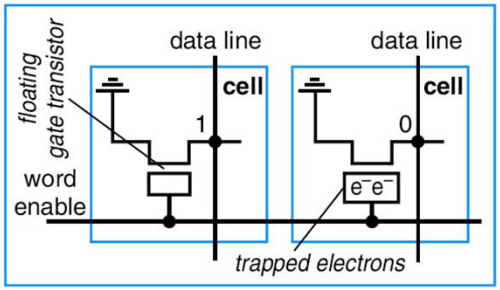


Figure 5.68 EPROM cells: left cell programmed with 1, right cell with 0.

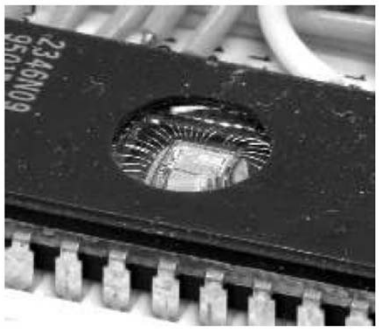


Figure 5.69 The "window" in the package of a microprocessor that uses an EPROM to store programs.

Source: Digital Design, Frank Vahid, 1st ed, 2006
http://www.cs.ucr.edu/~vahid/dd

ROM Types

- **Electronically-Erasable Programmable ROM (EEPROM)**
  - Similar to EPROM – uses floating-gate transistor, electronic programming to trap electrons in certain cells
  - But erasing is done electronically (no UV)
    - Erasing done one word at a time

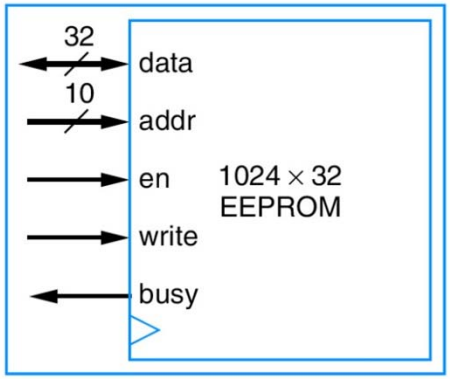


Figure 5.70 1024x32 EEPROM block symbol.

- **Flash memory**
  - Similar to EEPROM, but all words (or large blocks of words) can be erased simultaneously
  - Became very popular (after late 1990)
- Both types are in-system programmable
  - Requires bi-directional data lines, and write control input
  - Also need busy output to indicate that erasing is in progress – erasing takes some time

Source: Digital Design, Mano and Ciletti, 4E, ISBN: 0131989243

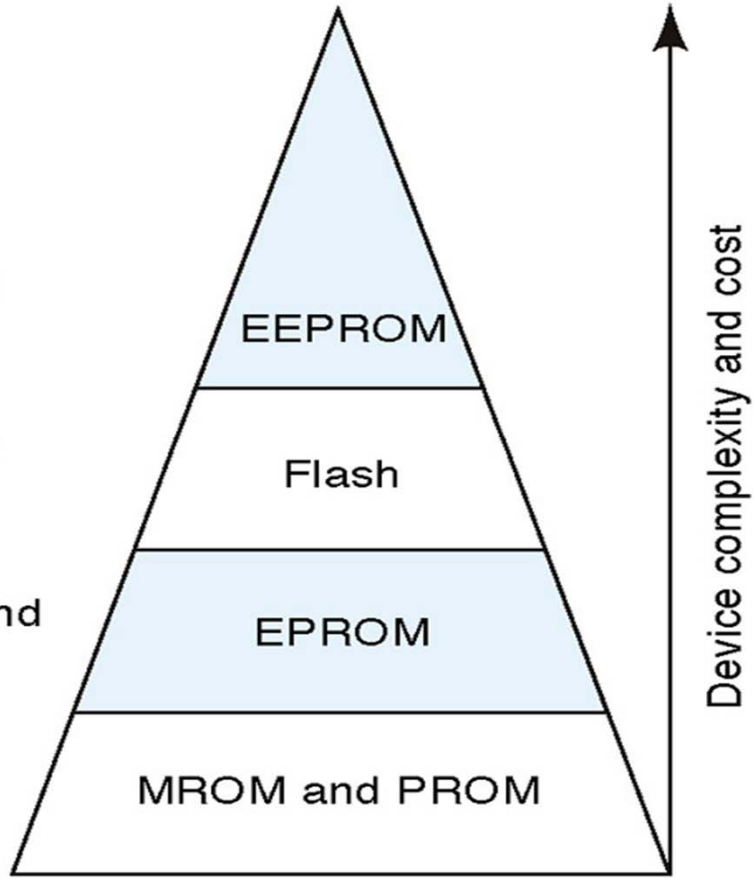
ROM Comparisons

In-circuit, electrically erasable byte-by-byte

In-circuit, electrically erasable by sector or in bulk (all cells)

UV erasable in bulk; erased and reprogrammed out of circuit

Cannot be erased and reprogrammed



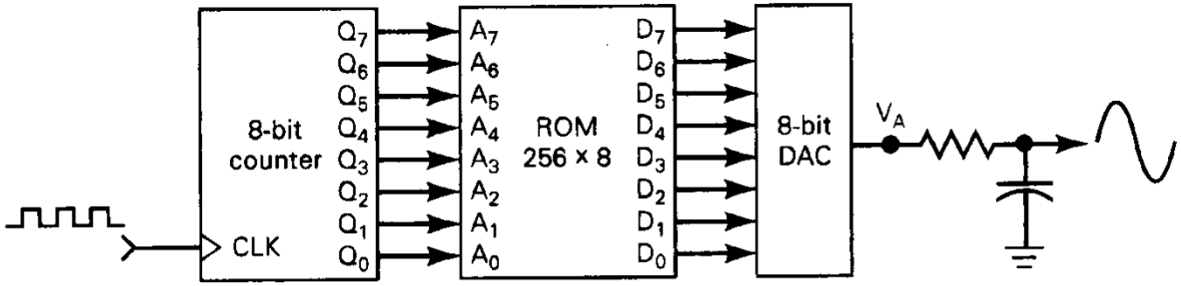
ROM Applications

- Firmware
  - Is in the storage of data and program codes that must be available immediately on power-up of computer/MCU systems
  - MCU systems require this so that MPU can execute appropriate instructions to initiate the system and invoke an OS from auxiliary memory (bootstrap or booting) - BIOS
  - EEPROM is often used to allow the firmware to be upgraded if necessary

- Stored Data Tables
  - To store tables of constant look-up data, e.g. trigonometric tables or code-conversion tables, etc.

ROM Applications

- Auxiliary Storage
  - Low cost flash memory is starting to become increasingly popular as an alternative form of auxiliary storage due to its high speed, low power requirements and lack of moving parts e.g. SDCARD, Compact flash cards, flash drive, etc.
- Function Generator
  - By connecting the outputs of a ROM to a DAC and the address lines to a counter, pre-set voltage functions can be generated (a low pass filter is usually employed to smooth the analog output), e.g. speech synthesizer, etc.



Acknowledgments

- These slides have been prepared by Khan Wahid and may contain material copyrighted by:
  - **Digital Systems: Principles and Applications, Tocci, Widmer and Moss, 10E, ISBN: 0131725793**
  - **Digital Design**, Frank Vahid, 1st ed, 2006
  - **Fundamentals of Digital Logic with Verilog Design**, Brown and Vranesic, 2E, ISBN: 9780073380339
  - **Digital Design**, Mano and Ciletti, 4E, ISBN: 0131989243