Topic 8

Memory & Memory Interface

Peter Cheung
Department of Electrical & Electronic Engineering
Imperial College London

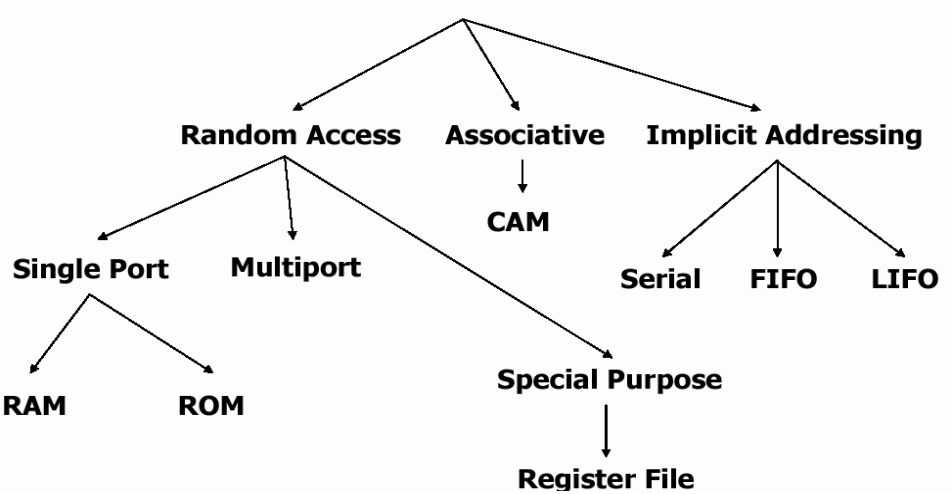
URL: www.ee.imperial.ac.uk/pcheung/
E-mail: p.cheung@imperial.ac.uk

Key Points addressed in this Topic

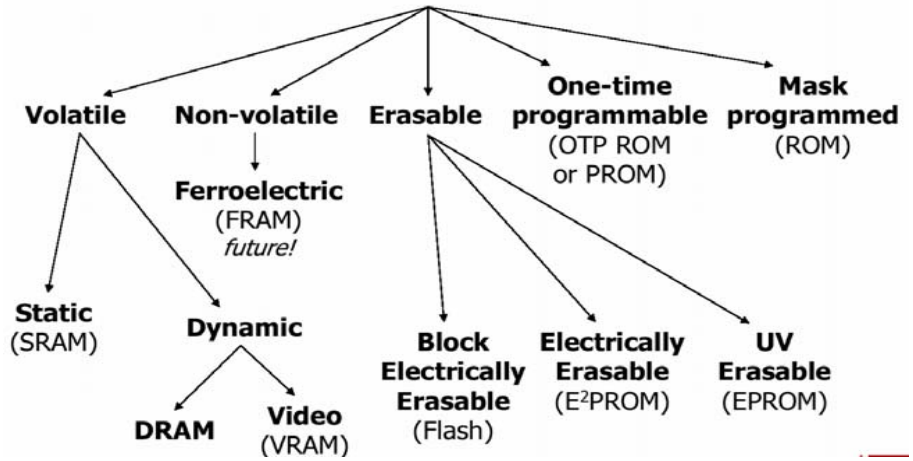
- Taxonomy of Memories
- Memory organisation
- Static memory timing
- Embedded RAM in Virtex FPGA – dual port RAM
- SRAM application - FIFO
- Dynamic memory
- Synchronous DRAM (SDRAM)
- SDRAM Timing

Slides based on notes from Paolo Ienne, EPFL & David Cullen, Berkeley.

Taxonomy of Memories

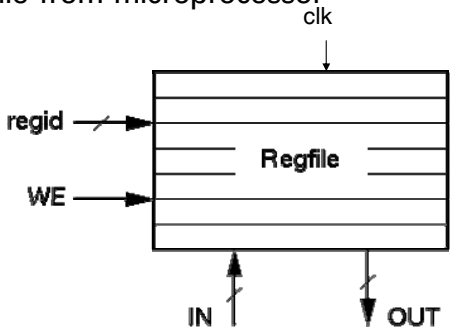


Taxonomy of Random Access Memory



Register File

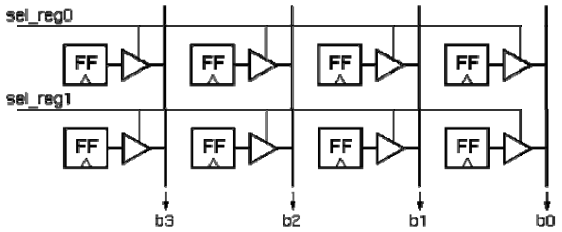
- Register file from microprocessor



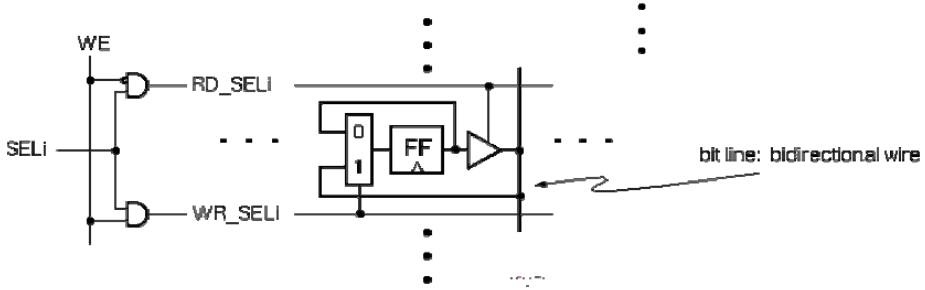
regid = register identifier (address of word in memory)
sizeof(regid) = log2(# of reg)
WE = write enable

Register File Internals

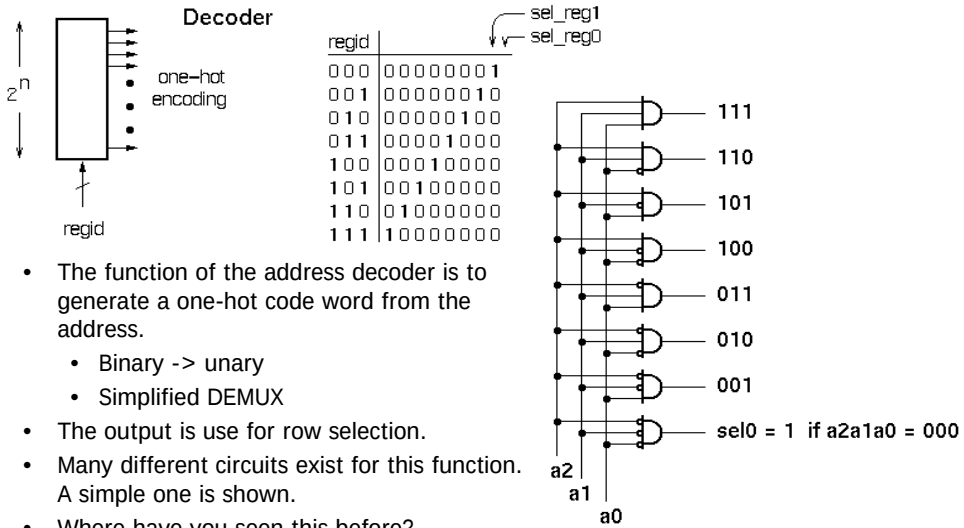
- For read operations, functionally the regfile is equivalent to a 2-D array of flip-flops with tristate outputs on each
  - MUX, but distributed
  - Unary control



- Cell with added write logic:



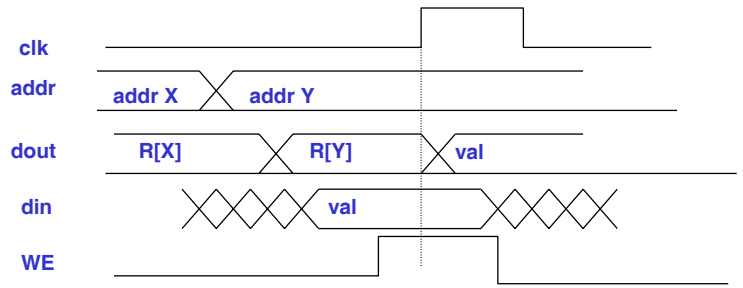
Regid (address) Decoding



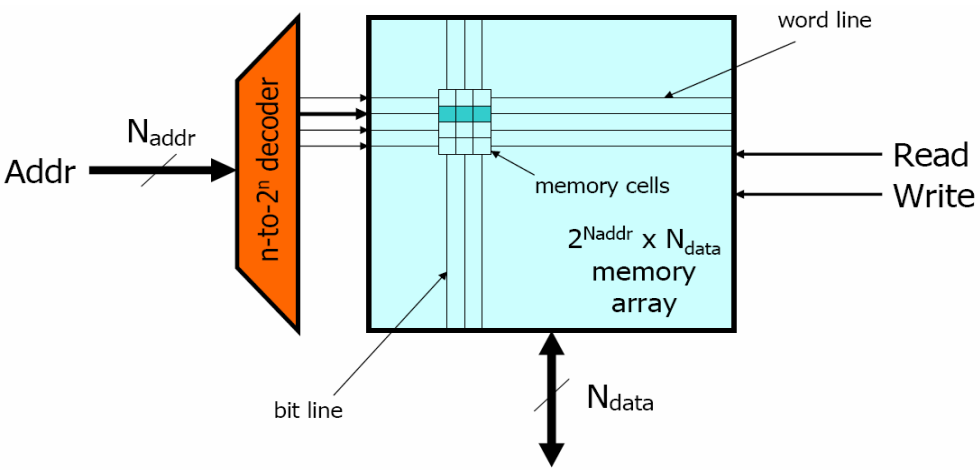
- The function of the address decoder is to generate a one-hot code word from the address.
  - Binary -> unary
  - Simplified DEMUX
- The output is use for row selection.
- Many different circuits exist for this function. A simple one is shown.
- Where have you seen this before?

Accessing Register Files

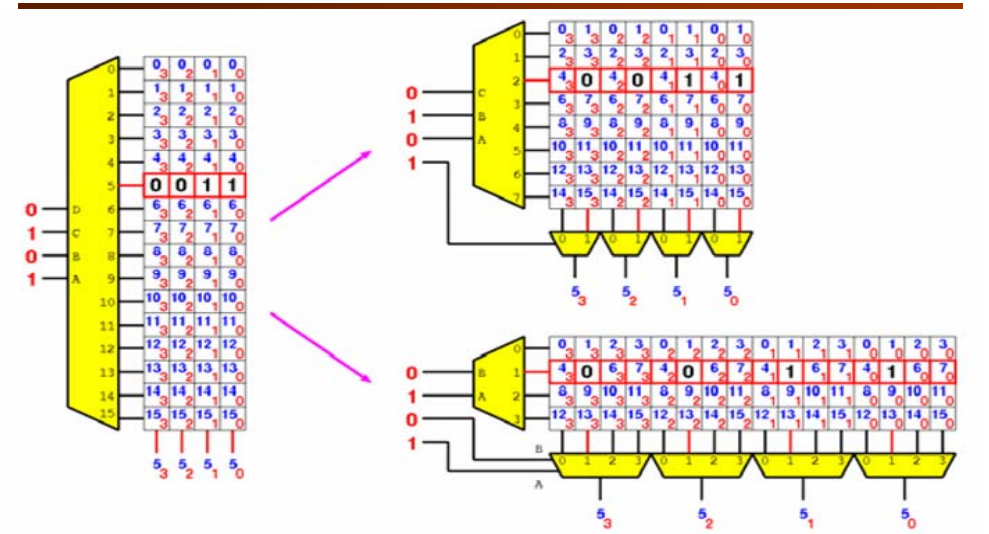
- Read: output is a combinational function of the address input
- Write is synchronous
  - If enabled, input data is written to selected word on the clock edge
- Often multi-ported



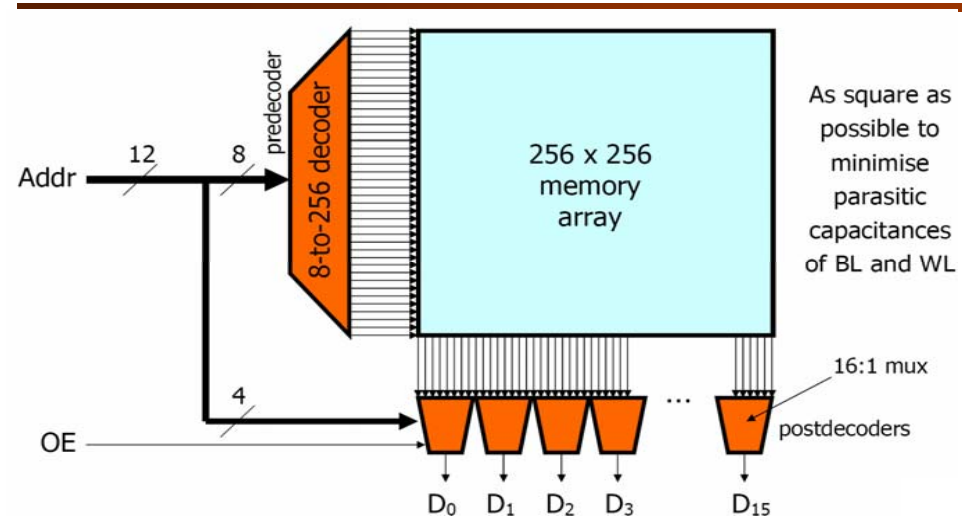
Simplified RAM organization



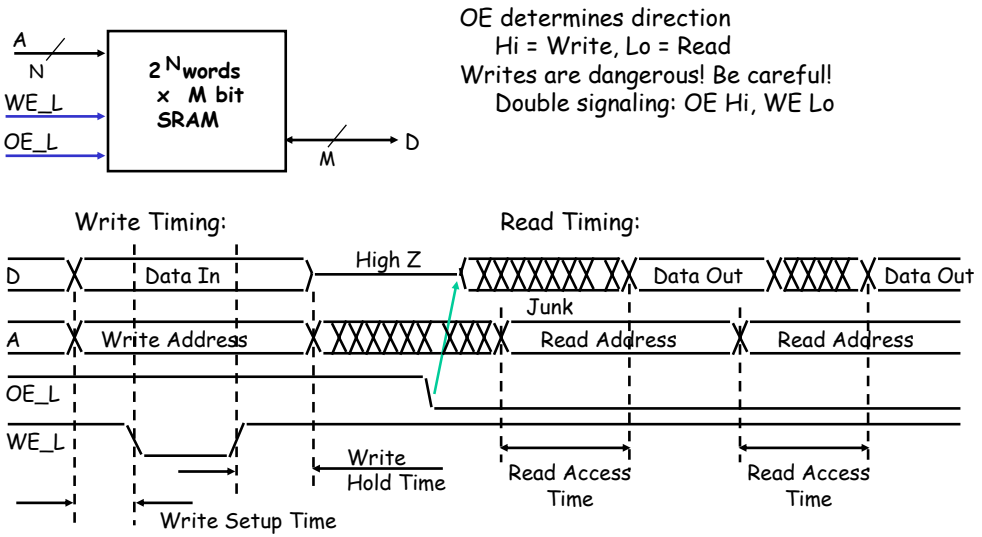
Different internal array organization



Typical organisation for a 4K x 16 bit RAM

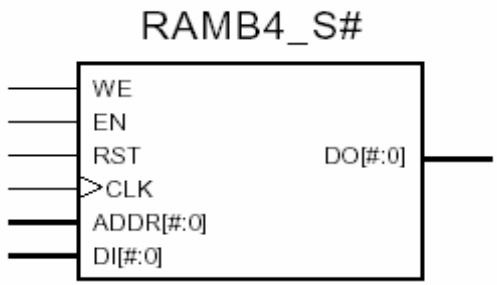


Typical SRAM Timing

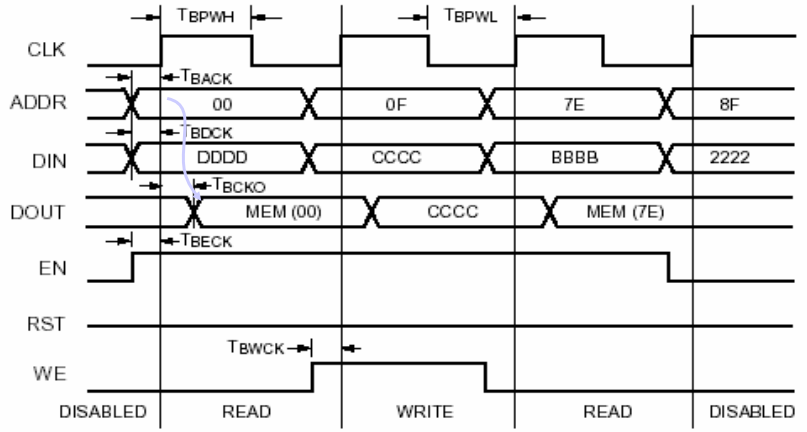


Memory Blocks in FPGAs

- LUTs can double as small RAM blocks:
- Newer FPGA families include larger on-chip RAM blocks (usually dual ported):
  - Called **block selectRAMs** in Xilinx Virtex series
  - 4k bits each



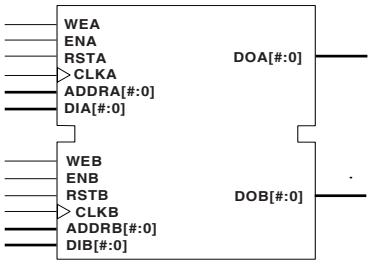
Synchronous SRAM timing



Timing Diagram for Single-Port Block SelectRAM+ Memory

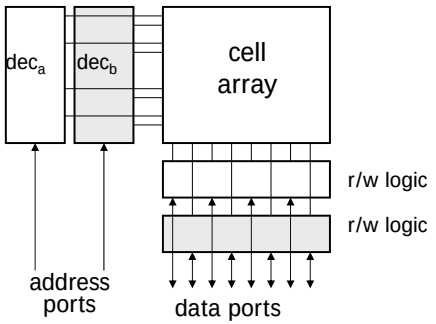
Virtex "Block RAMs"

- Each block SelectRAM (block RAM) is a fully synchronous (synchronous write *and* read) dual-ported (true dual port) 4096-bit RAM with independent control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion.
- CLKA and CLKB can be independent, providing an easy way to "cross clock boundaries".
- Around 160 of these on the XCV2000E. Multiples can be combined to implement, wider or deeper memories.

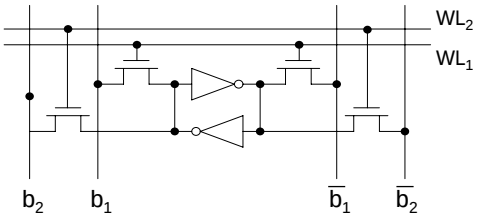


Dual-ported Memory Internals

- Add decoder, another set of read/write logic, bits lines, word lines:



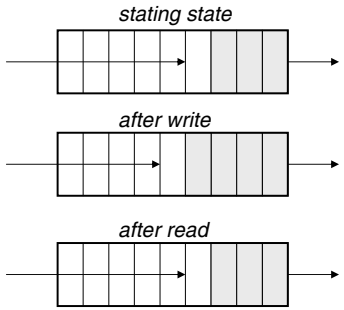
- Example cell: SRAM



- Repeat everything but cross-coupled inverters.
- This scheme extends up to a couple more ports, then need to add additional transistors.

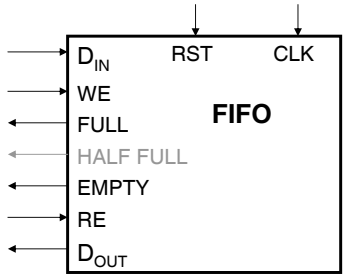
First-in-first-out (FIFO) Memory

- Used to implement *queues*.
- These find common use in computers and communication circuits.
- Generally, used for rate matching data producer and consumer:



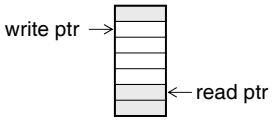
- Producer can perform many writes without consumer performing any reads (or vice versa). However, because of finite buffer size, on average, need equal number of reads and writes.
- Typical uses:
  - interfacing I/O devices. Example network interface. Data bursts from network, then processor bursts to memory buffer (or reads one word at a time from interface). Operations not synchronized.
  - Example: Audio output. Processor produces output samples in bursts (during process swap-in time). Audio DAC clocks it out at constant sample rate.

FIFO Interfaces

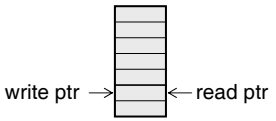


- After write or read operation, FULL and EMPTY indicate status of buffer.
- Used by external logic to control own reading from or writing to the buffer.
- FIFO resets to EMPTY state.
- HALF FULL (or other indicator of partial fullness) is optional.

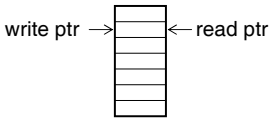
- Address pointers are used internally to keep next write position and next read position into a dual-port memory.



- If pointers equal after write ⇒ FULL:



- If pointers equal after read ⇒ EMPTY:



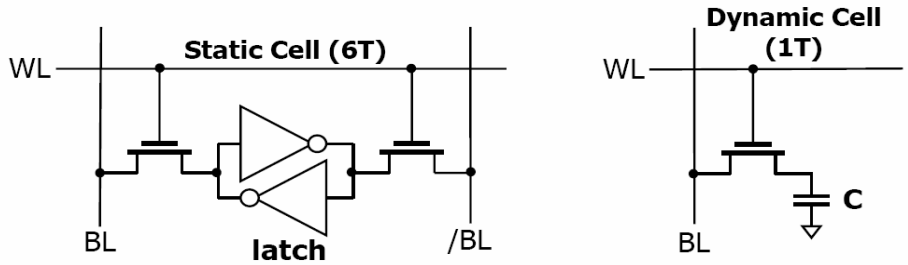
FIFO Implementation

- Assume, dual-port memory with asynchronous read, synchronous write.
- Binary counter for each of read and write address. CEs controlled by WE and RE.
- Equal comparator to see when pointers match.
- Flip-flop each for FULL and EMPTY flags:

WE	RE	equal	$EMPTY_i$	$FULL_i$
0	0	0	0	0
0	0	1	$EMPTY_{i-1}$	$FULL_{i-1}$
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	1
1	1	0	0	0
1	1	1	$EMPTY_{i-1}$	$FULL_{i-1}$

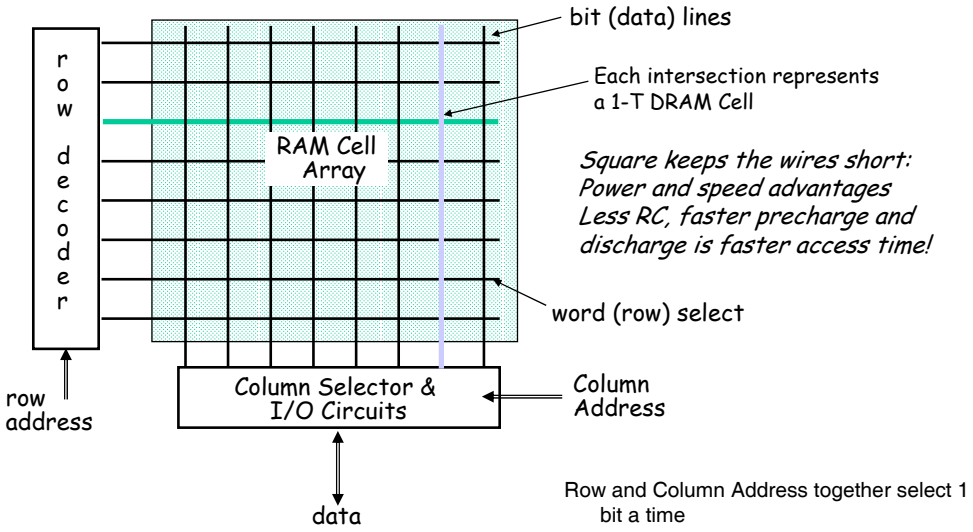
SRAM vs DRAM

- Completely different storage mechanism:



- Capacitor discharges (leakage) and needs refresh every some time ("seldom", i.e. typ. ~10ms)
- Additional design complexity (sometimes hidden from the designer, esp. in ASIC)

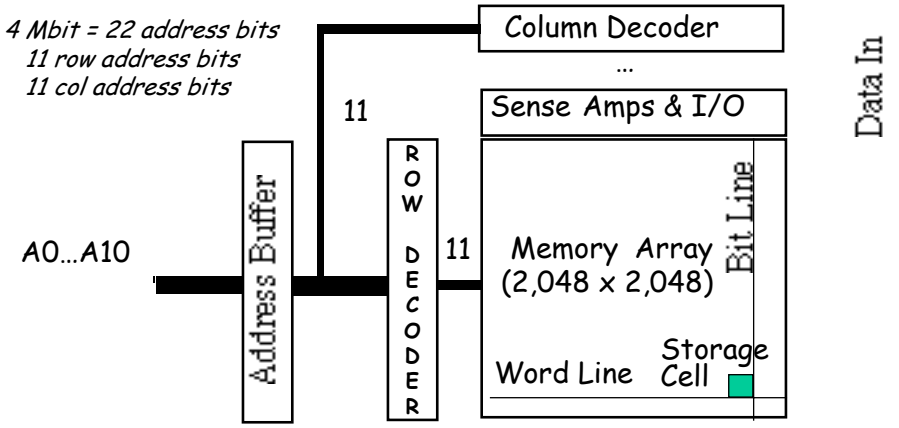
Classical DRAM Organization (Square)



Square keeps the wires short:
Power and speed advantages
Less RC, faster precharge and discharge is faster access time!

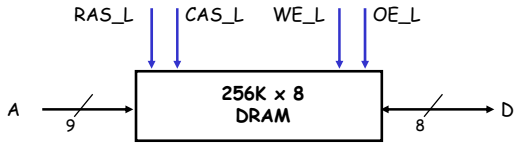
Row and Column Address together select 1 bit a time

DRAM Logical Organization (4 Mbit)



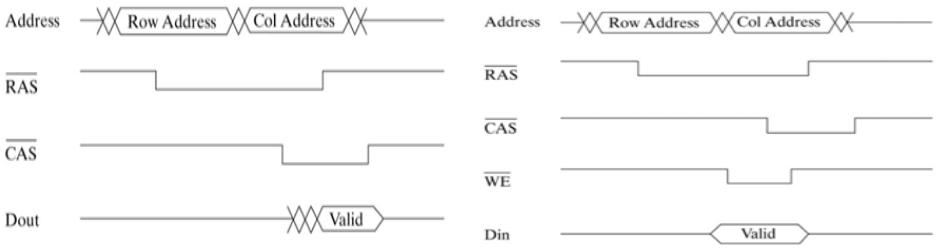
- Square root of bits per RAS/CAS
  - Row selects 1 row of 2048 bits from 2048 rows
  - Col selects 1 bit out of 2048 bits in such a row

Logic Diagram of a Typical DRAM



- Control Signals (RAS_L, CAS_L, WE_L, OE_L) are all active low
- Din and Dout are combined (D):
  - WE_L is asserted (Low), OE_L is disasserted (High)
    - ➤ D serves as the data input pin
  - WE_L is disasserted (High), OE_L is asserted (Low)
    - ➤ D is the data output pin
- Row and column addresses share the same pins (A)
  - RAS_L goes low: Pins A are latched in as row address
  - CAS_L goes low: Pins A are latched in as column address
  - RAS/CAS edge-sensitive

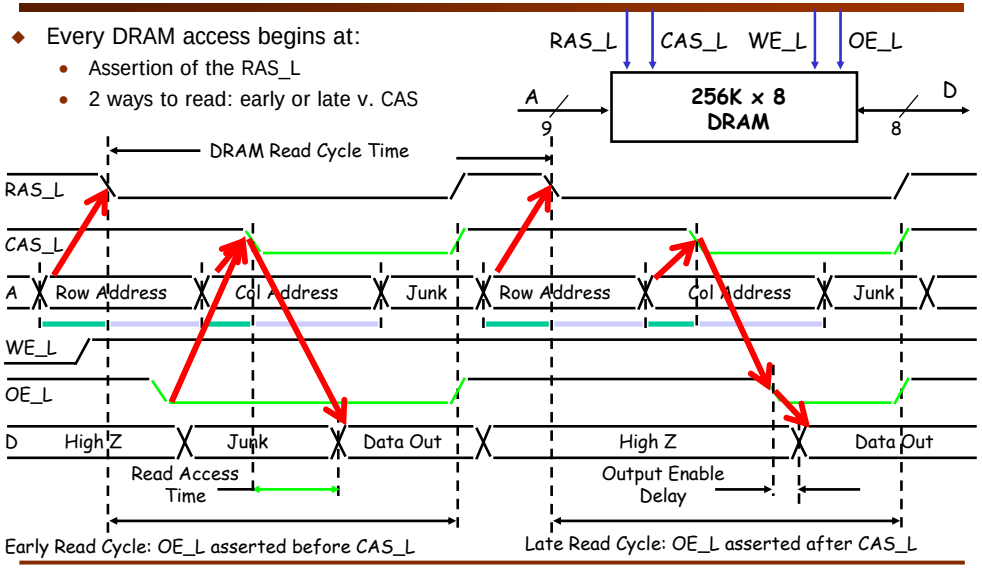
Basic DRAM read & write



- Strobe address in two steps

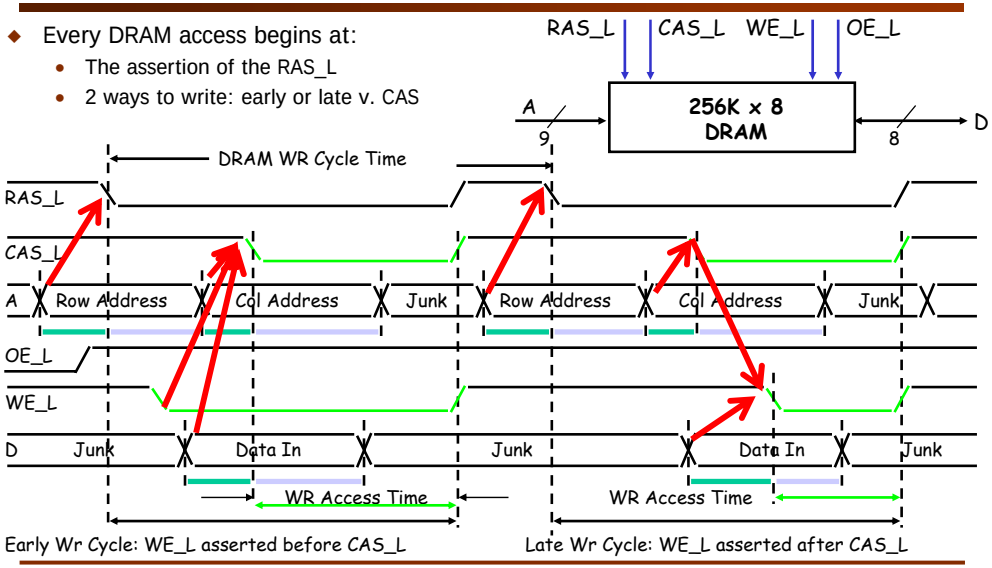
DRAM READ Timing

- Every DRAM access begins at:
  - Assertion of the RAS_L
  - 2 ways to read: early or late v. CAS



DRAM WRITE Timing

- Every DRAM access begins at:
  - The assertion of the RAS_L
  - 2 ways to write: early or late v. CAS

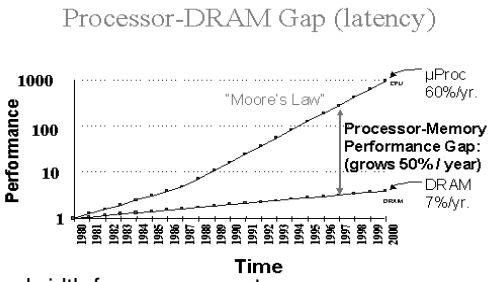


Key DRAM Timing Parameters

- t_{RAC}: minimum time from RAS line falling to the valid data output.
  - Quoted as the speed of a DRAM
  - A fast 4Mb DRAM t_{RAC} = 60 ns
- t_{RC}: minimum time from the start of one row access to the start of the next.
  - t_{RC} = 110 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns
- t_{CAC}: minimum time from CAS line falling to valid data output.
  - 15 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns
- t_{PC}: minimum time from the start of one column access to the start of the next.
  - 35 ns for a 4Mbit DRAM with a t_{RAC} of 60 ns

Memory in Desktop Computer Systems:

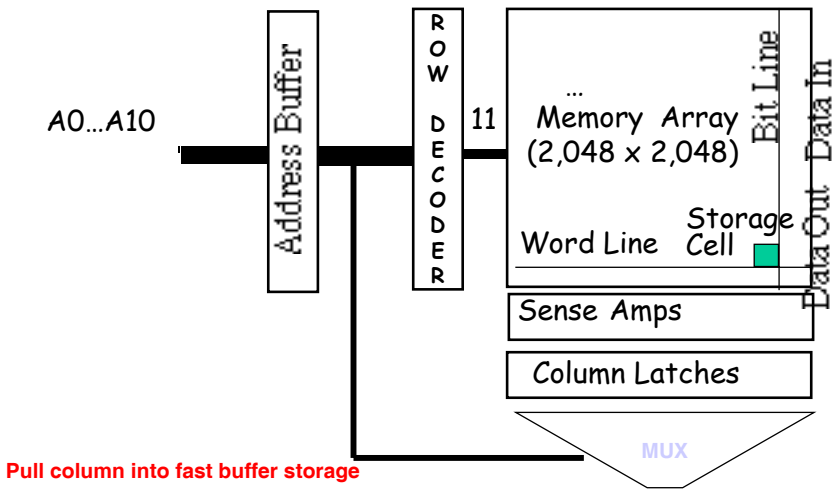
- **SRAM** (lower density, higher speed) used in CPU register file, on- and off-chip caches.
- **DRAM** (higher density, lower speed) used in main memory



Closing the GAP:

1. Caches are growing in size.
2. Innovation targeted towards higher bandwidth for memory systems:
   - SDRAM - synchronous DRAM
   - RDRAM - Rambus DRAM
   - EDORAM - extended data out SRAM
   - Three-dimensional RAM
   - hyper-page mode DRAM video RAM
   - multibank DRAM

DRAM with Column buffer

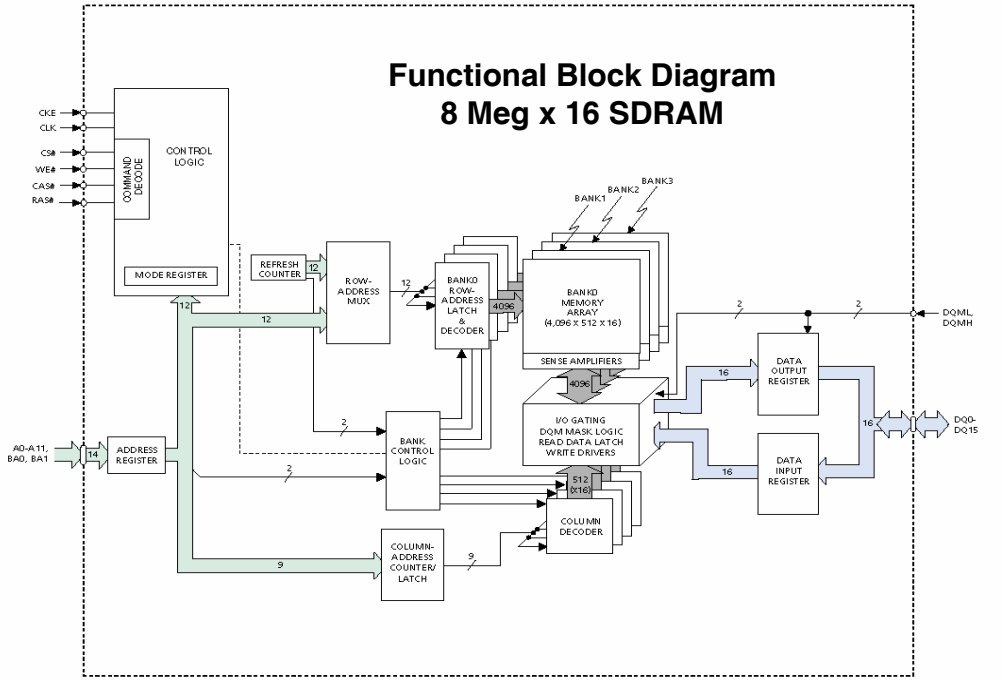


Pull column into fast buffer storage

Access sequence of bit from there

Optimized Access to Cols in Row

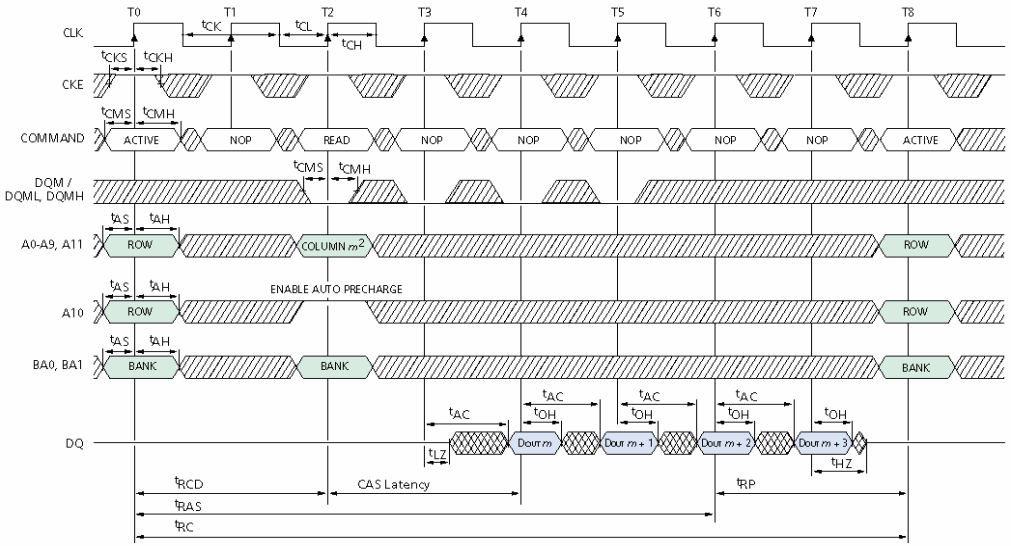
- Often want to access a sequence of bits
- Page mode
  - After RAS / CAS, can access additional bits in the row by changing column address and strobing CAS
- Static Column mode
  - Change column address (without repeated CAS) to get different bit
- Nibble mode
  - Pulsing CAS gives next bit mod 4
- Video ram
  - Serial access



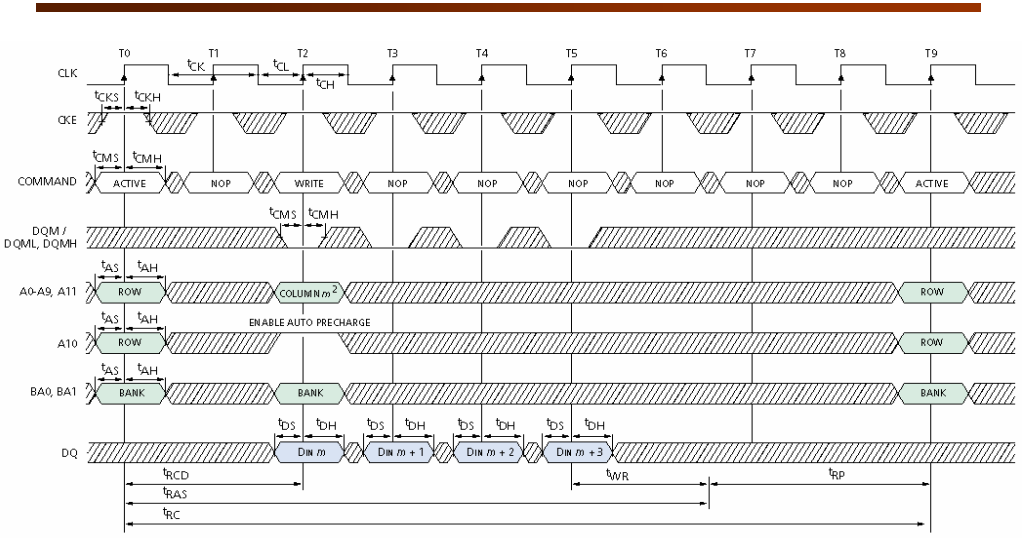
SDRAM Details

- Multiple “banks” of cell arrays are used to reduce access time:
  - Each bank is 4K rows by 512 “columns” by 16 bits (for our part)
- Read and Write operations as split into RAS (row access) followed by CAS (column access)
- These operations are controlled by sending commands
  - Commands are sent using the RAS, CAS, CS, & WE pins.
- Address pins are “time multiplexed”
  - During RAS operation, address lines select the bank and row
  - During CAS operation, address lines select the column.
- “ACTIVE” command “opens” a row for operation
  - transfers the contents of the entire to a row buffer
- Subsequent “READ” or “WRITE” commands modify the contents of the row buffer.
- For burst reads and writes during “READ” or “WRITE” the starting address of the block is supplied.
  - Burst length is programmable as 1, 2, 4, 8 or a “full page” (entire row) with a burst terminate option.
- Special commands are used for initialization (burst options etc.)
- A burst operation takes ≈ 4 + n cycles (for n words)

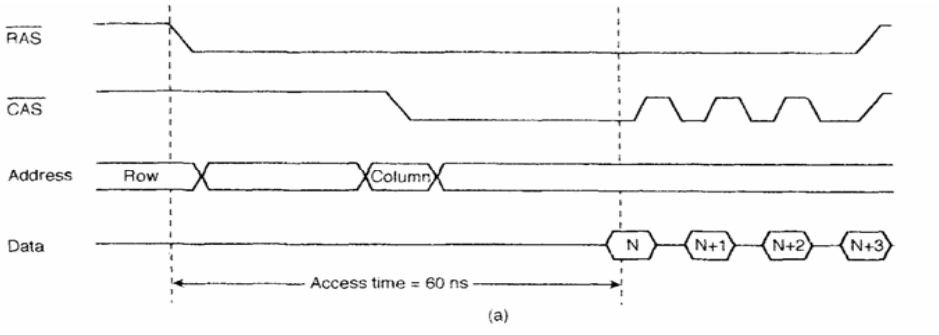
READ burst (with auto precharge)



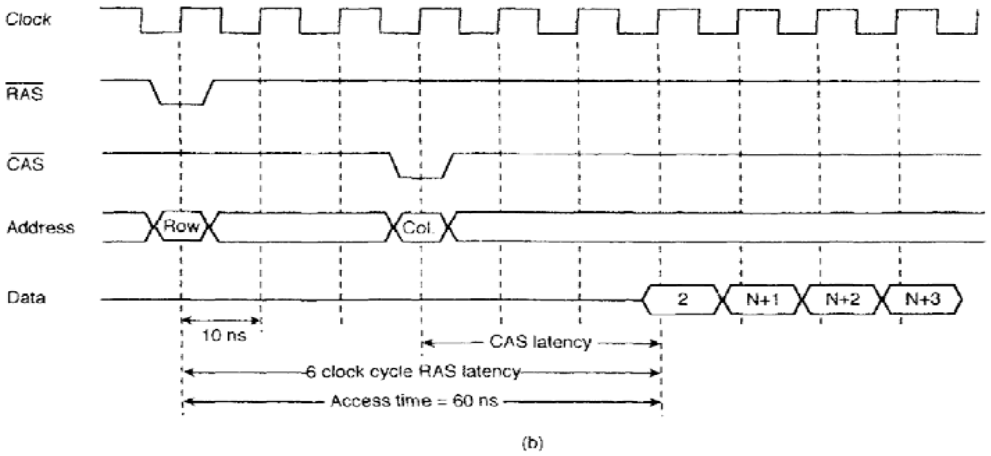
WRITE burst (with auto precharge)



Simplified DRAM timing (burst mode)



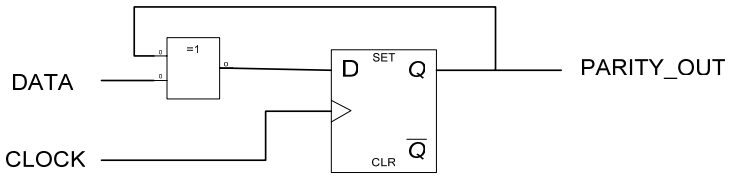
Simplified SDRAM Timing (burst mode)



Error Detection in Memory

- Add extra data bit to ensure total number of 1's is EVEN – even parity
- Parity bit is the exclusive-OR of all other data bits
- If one of the received bit is wrong, then number of 1's will be odd (including parity bit) and error is detected

Data	Parity bit	number of 1's
1 1 0 1 1 1 0	1	6
0 1 0 1 1 0 1	0	4



Error Correction in Memory

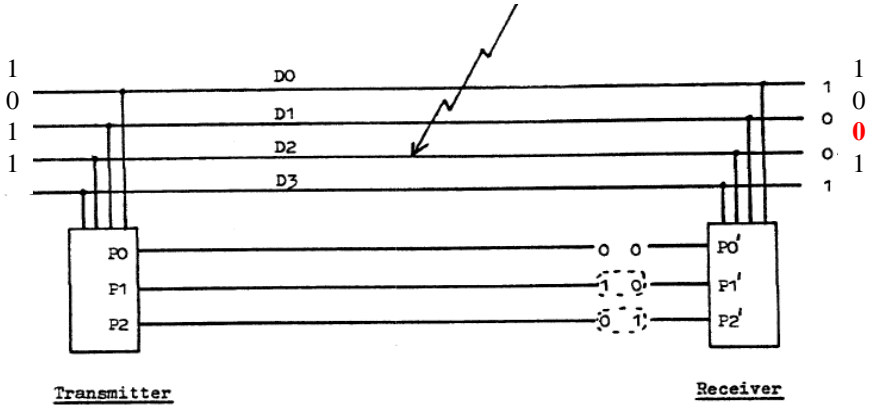
- Simply parity is limiting in its ability to detect error
- It cannot CORRECT error
- Use multiple check bits, each giving parity for a DIFFERENT COMBINATION of data bits
- If a data bit is wrong, it will affect only some check bits – from which deduce WHICH bit is wrong:

	D0	D1	D2	D3	
P0:	x	x		x	P0 = D0 ⊕ D1 ⊕ D3
P1:	x		x	x	P1 = D0 ⊕ D2 ⊖ D3
P2:		x	x	x	P2 = D1 ⊕ D2 ⊕ D3
Code:	3	5	6	7	(0,1,2,4 not used)

- Each data bit affects a UNIQUE combination of check bits
- Each data bit affects at least TWO check bits – this means an error in check bit is not confused with error in data bit

Example

- If D2 is in error, check bits P1 & P2 will be wrong:



More on check bits

- N check bits are sufficient for 2^N – N - 1 data bits
  - Each data bit must affect a different combination of check bits
  - There are 2^N possible combinations for N check bits
  - Of these, N affect only 1 bit and 1 affects no bits at all
- If you regard the combinations as binary numbers, then for 3 check bits, numbers 0, 1, 2 and 4 are ruled out, leaving 3, 5, 6, 7 as useful:

	D0	D1	D2	D3	
P0:	x	x		x	P0 = D0 ⊕ D1 ⊕ D3
P1:	x		x	x	P1 = D0 ⊕ D2 ⊖ D3
P2:		x	x	x	P2 = D1 ⊕ D2 ⊕ D3
Code:	3	5	6	7	(0,1,2,4 not used)

ECC for 8 bit bytes

- Detect and correct any single bit error
- Can add an extra check bit giving overall parity to both data & check bits
  - will detect double bit errors but not correct
  - For double errors, overall parity bit will be OK, but check bits is wrong

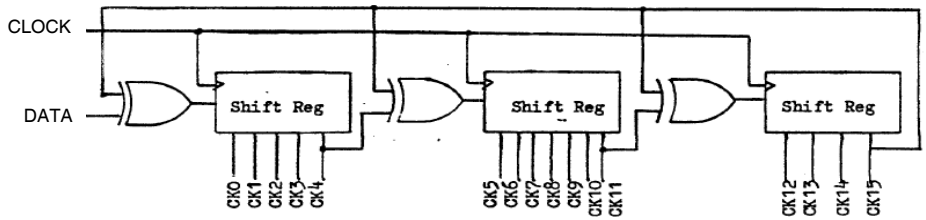
	D0	D1	D2	D3	D4	D5	D6	D7	
P0:	x	x		x	x		x		
P1:	x		x	x		x	x		
P2:		x	x	x				x	
P3:					x	x	x	x	
Code:	3	5	6	7	9	10	11	12	(0,1,2,4,8 not used, 13,14,15 spare)

Error detection in block of data

- Method 1: Add up all data words, ignore carry(s) and store result as CHECKSUM
  - Advantages: Very easy to do
  - Disadvantages: Errors can cancel out; will not detect gross errors such as MSB stuck at '0'
- Method 2: Form checksum as before, but add an extra 1 each time you get a carry.
  - Equivalent to doing addition modulo 2^n-1 instead of module 2^n, where n is the number of bits in each data word
  - Advantages: Almost as easy as before; less likely to ignore gross errors
  - Disadvantages: Errors can still cancel – e.g. a '1' changes to '0' and later a '0' changes to '1' in the same bit

Error detection in block of data - CRC

- Method 3: Cyclic Redundancy Check (CRC)
  - Send data through a shift register incorporating a feedback loop
  - Implement 'polynomial division' check because the action of the shift register is mathematically equivalent to dividing one algebraic polynomial by another polynomial.
  - Shown here is one using polynomial $x^{16}+x^{12}+x^5+1$: used by floppy disks
  - Whatever is left in shift register after all data bits are received is used a checksum



Primitive Polynomial Table

Degree (n)	Polynomial	Degree (n)	Polynomial
2, 3, 4, 6, 7, 15, 22, 60, 63	$x^n + x + 1$	12	$x^n + x^7 + x^4 + x^3 + 1$
5, 11, 21, 29, 35	$x^n + x^2 + 1$	33	$x^n + x^{13} + 1$
8, 19, 38, 43	$x^n + x^6 + x^5 + x + 1$	34	$x^n + x^{15} + x^{14} + x + 1$
9, 39	$x^n + x^4 + 1$	36	$x^n + x^{11} + 1$
10, 17, 20, 25, 28, 31, 41, 52	$x^n + x^3 + 1$	37	$x^n + x^{12} + x^{10} + x^2 + 1$
13, 24, 45, 64	$x^n + x^4 + x^3 + x + 1$	40	$x^n + x^{21} + x^{19} + x^2 + 1$
14, 16	$x^n + x^5 + x^4 + x^3 + 1$	42	$x^n + x^{23} + x^{22} + x + 1$
18, 57	$x^n + x^7 + 1$	46	$x^n + x^{21} + x^{20} + x + 1$
23, 47	$x^n + x^5 + 1$	54	$x^n + x^{37} + x^{36} + x + 1$
26, 27	$x^n + x^{12} + x^{11} + x + 1$	55	$x^n + x^{24} + 1$
30, 51, 53, 61, 70	$x^n + x^{16} + x^{15} + x + 1$	58	$x^n + x^{19} + 1$
32, 48	$x^n + x^{28} + x^{27} + x + 1$	65	$x^n + x^{18} + 1$
44, 50	$x^n + x^{27} + x^{26} + x + 1$	69	$x^n + x^{29} + x^{27} + x^2 + 1$
49, 68	$x^n + x^9 + 1$	71	$x^n + x^6 + 1$
56, 59	$x^n + x^{22} + x^{21} + x + 1$	72	$x^n + x^{53} + x^{47} + x^6 + 1$
66, 67, 74	$x^n + x^{10} + x^9 + x + 1$	73	$x^n + x^{25} + 1$