Preface

There are more slides here than will be used in lectures. The slides not covered will be listed on the website.

At least 10 minutes or so of each lecture will be devoted to example material, including previous exam questions, for which there are no slides in this handout.

Books related to the course

Suggested books include:

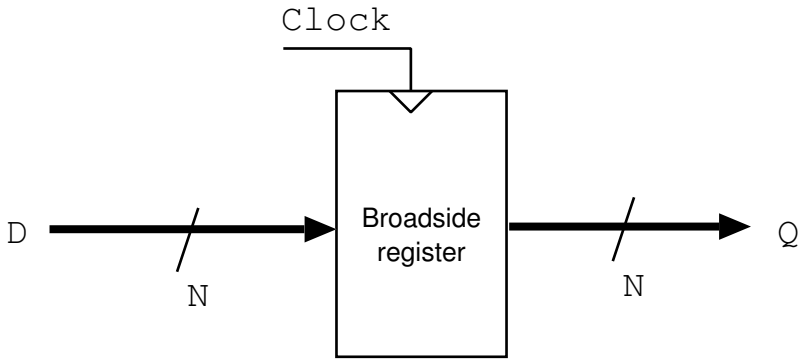
W.Ditch. *'Microelectronic Systems, A practical approach.'* Edward Arnold. The final chapters with details of the Z80 and 6502 are not relevant to this course.

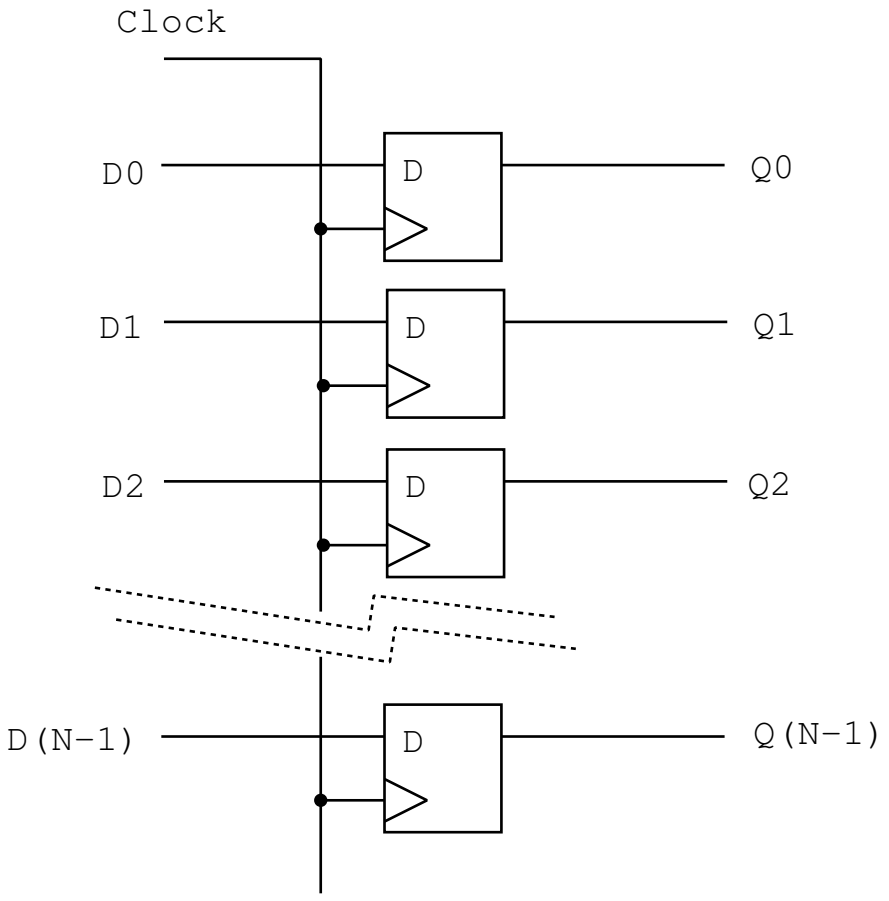
Floyd. *'Digital Fundamentals'* Prentice Hall International.

T.J. Stoneham. *'Digital Logic Techniques'* Chapman and Hall. This is a basic book and relates more to the previous course on Digital Electronics.

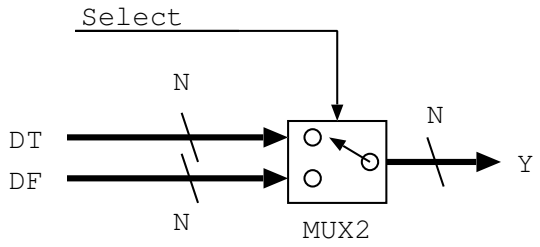
Randy H Katz. *'Contemporary logic design.'*

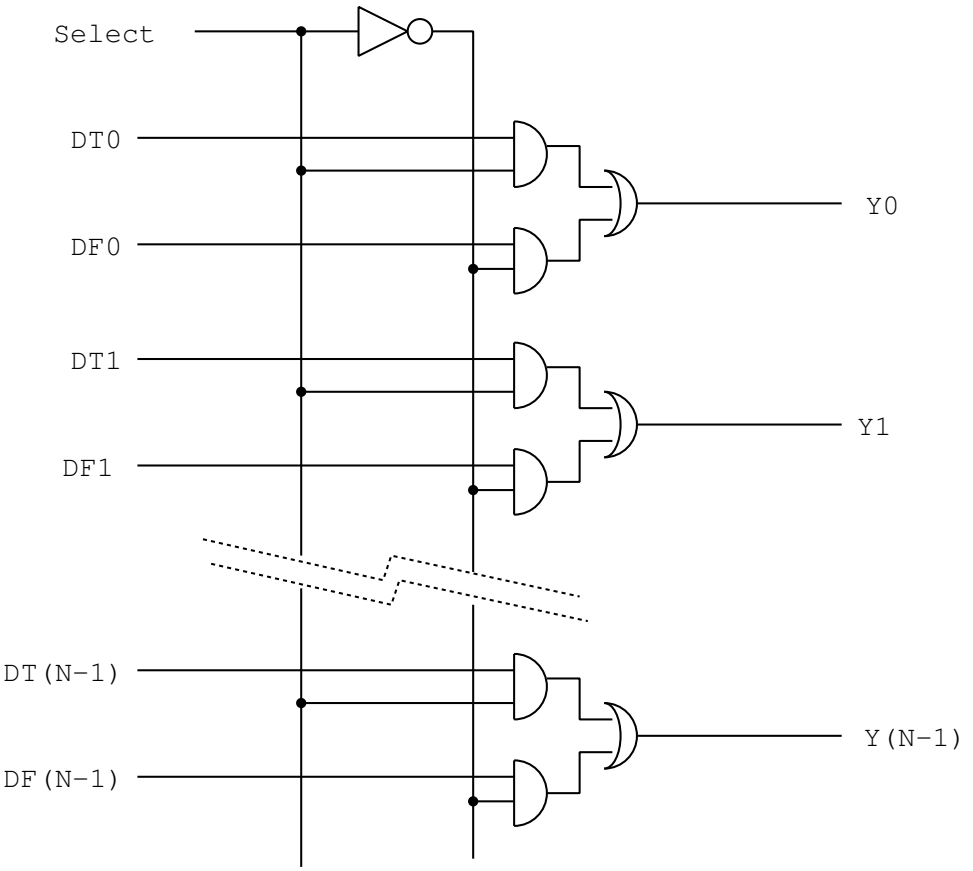
A Broadside Register



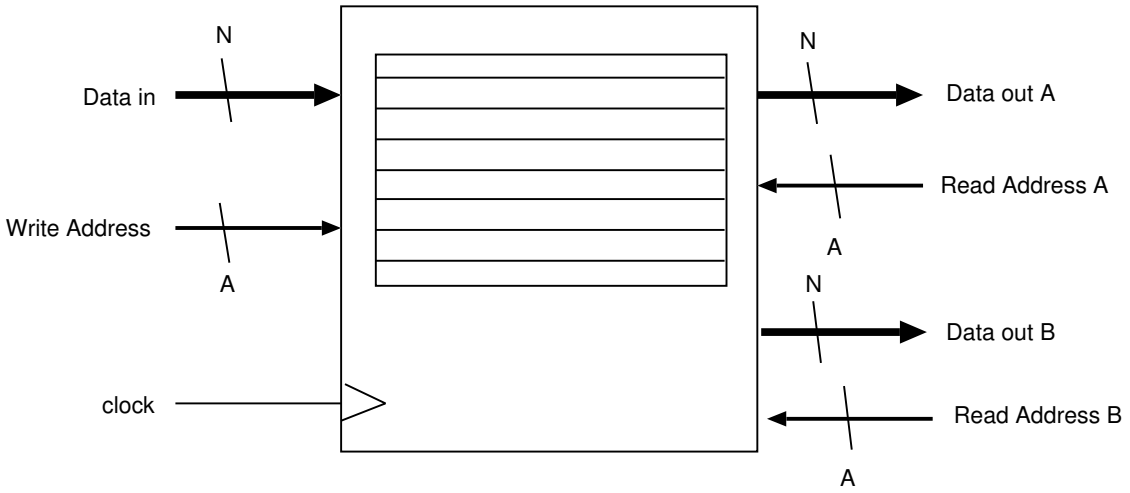


A broadside two-to-one multiplexor

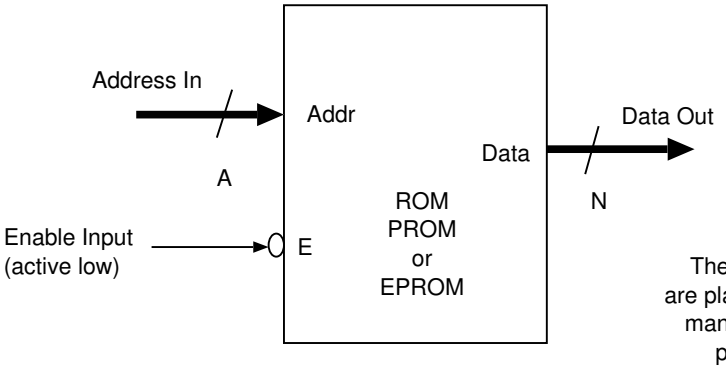




A dual port register file

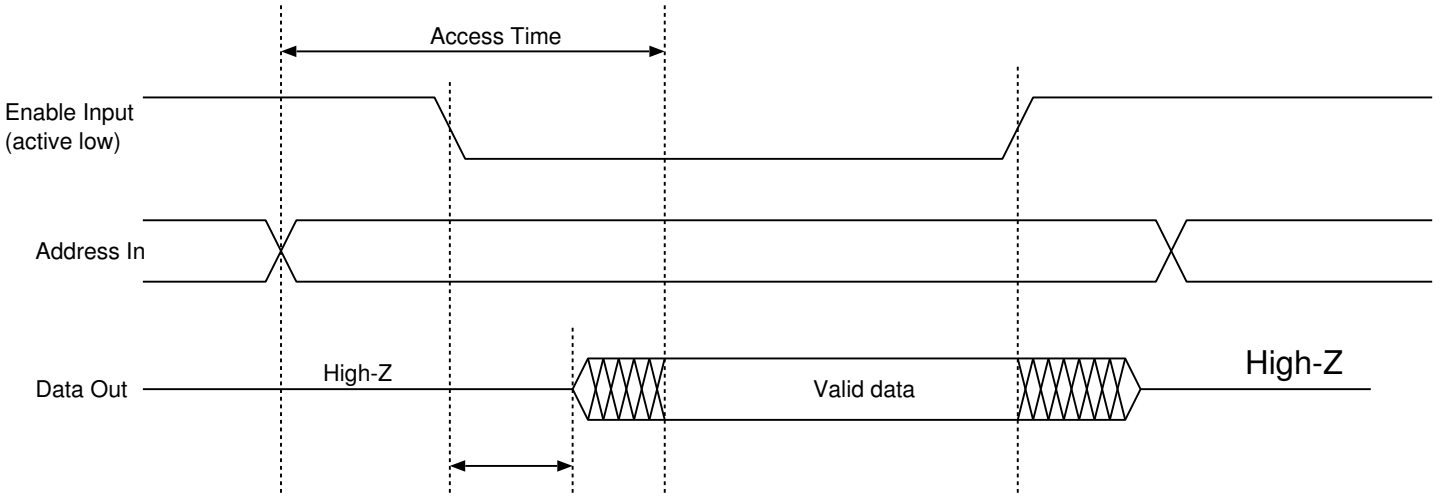


Read Only Memory (ROM)

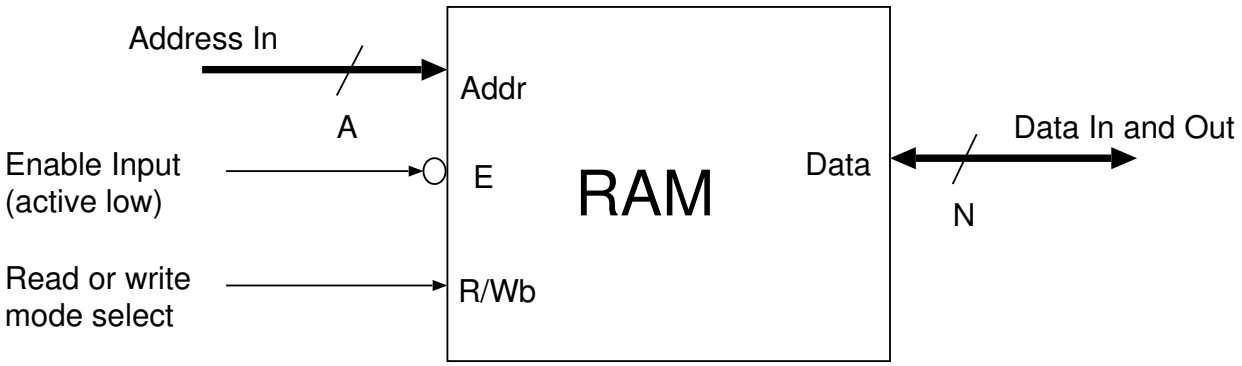


The ROM contents are placed inside during manufacture or field programming.

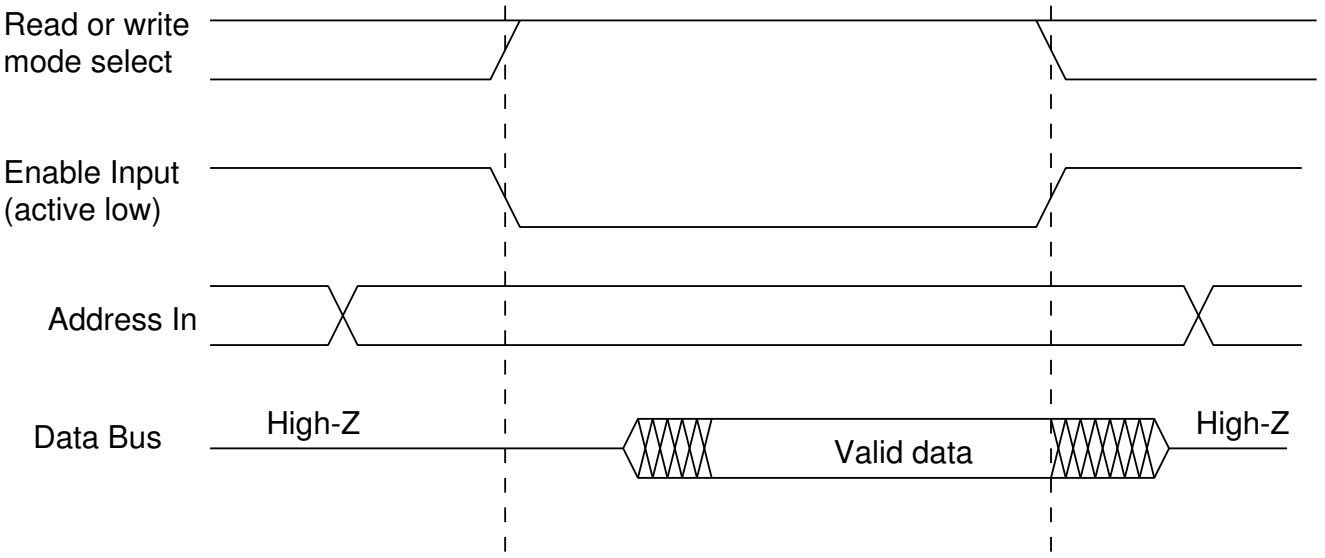
The ROM takes A address bits named A0 to A<A-1> and produces data words of N bits wide. For example, if A=5 and D=8 then the ROM contains 2**5 which is 32 locations of 8 bits each. The address lines are called A0, A1, A2, A3, A4 and the data lines D0, D1, ... D7



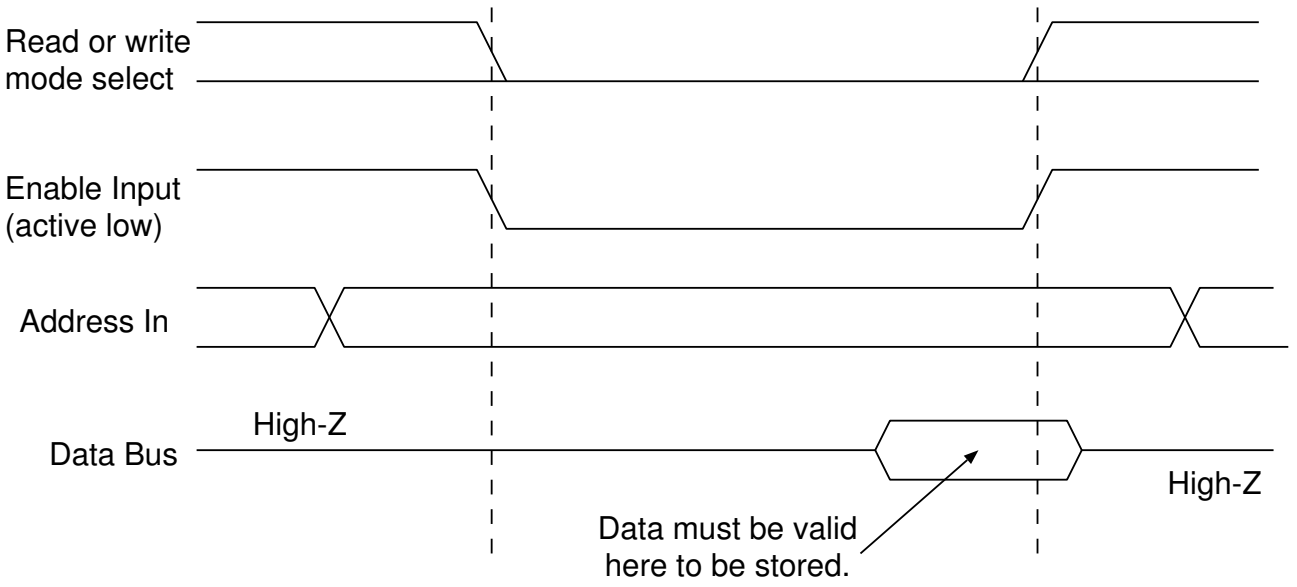
The ROM's outputs are high impedance unless the enable input is asserted (low). After the enable is low the output drivers turn on. When the address has been stable sufficiently long, valid data from that address comes out.



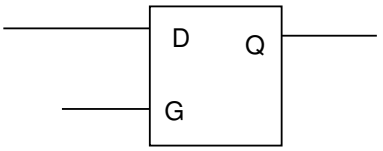
Read Cycle - Like the ROM



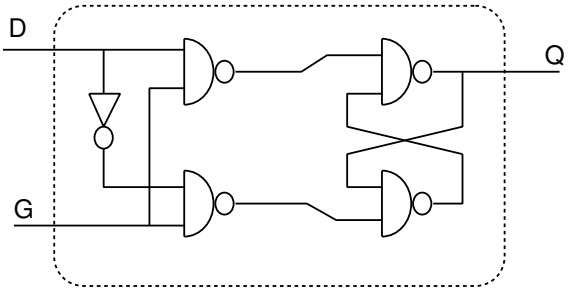
Write Cycle - Data stored internally



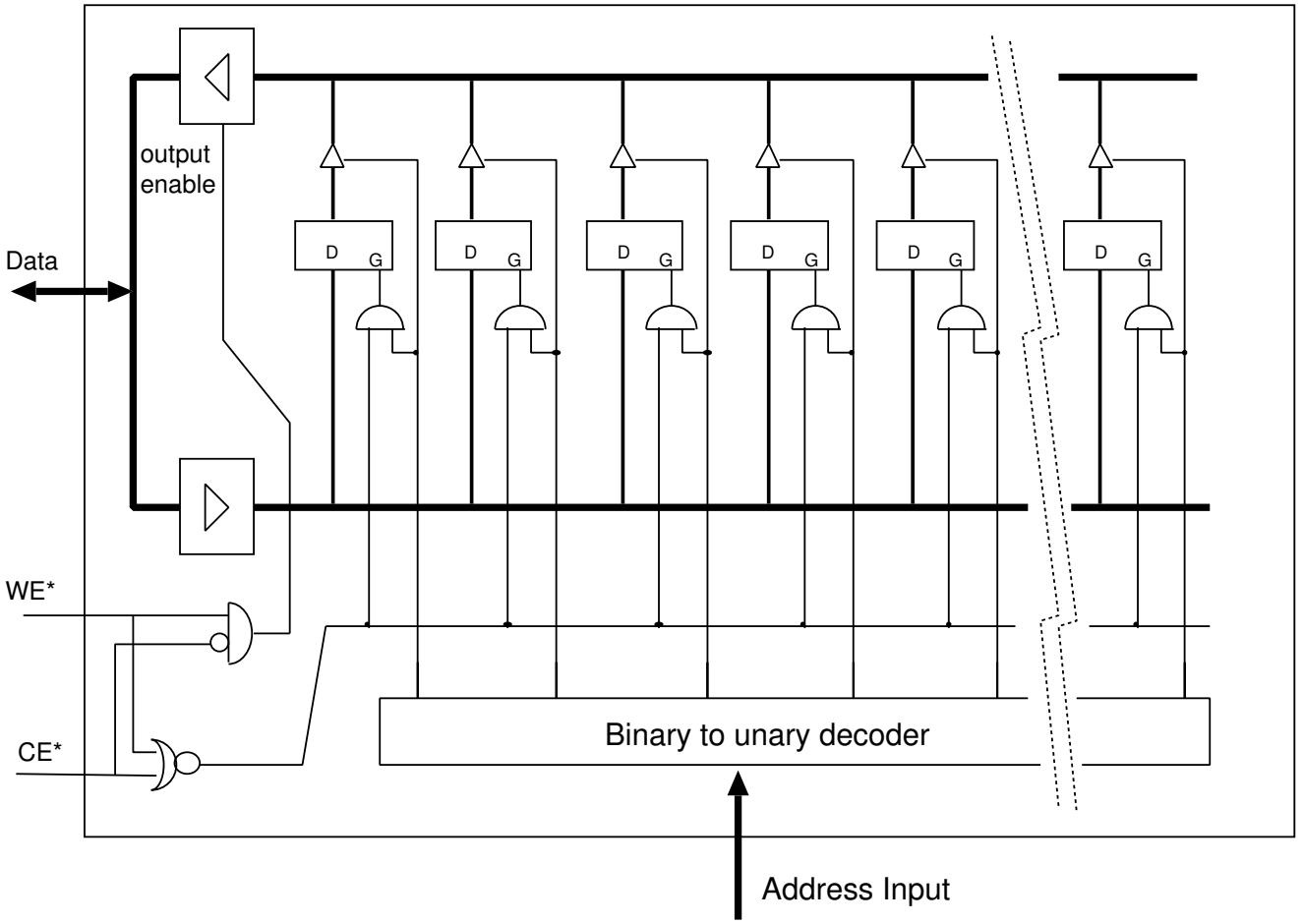
Unlike the edge-triggered flip-flop, the transparent latch passes data through in a transparent way when its enable input is high. When its enable input is low, the output stays at the current value.

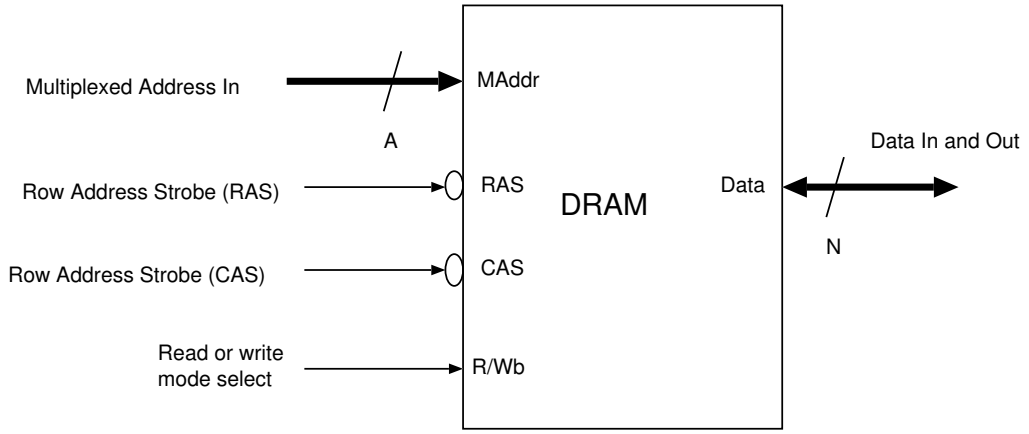


Transparent latch schematic symbol

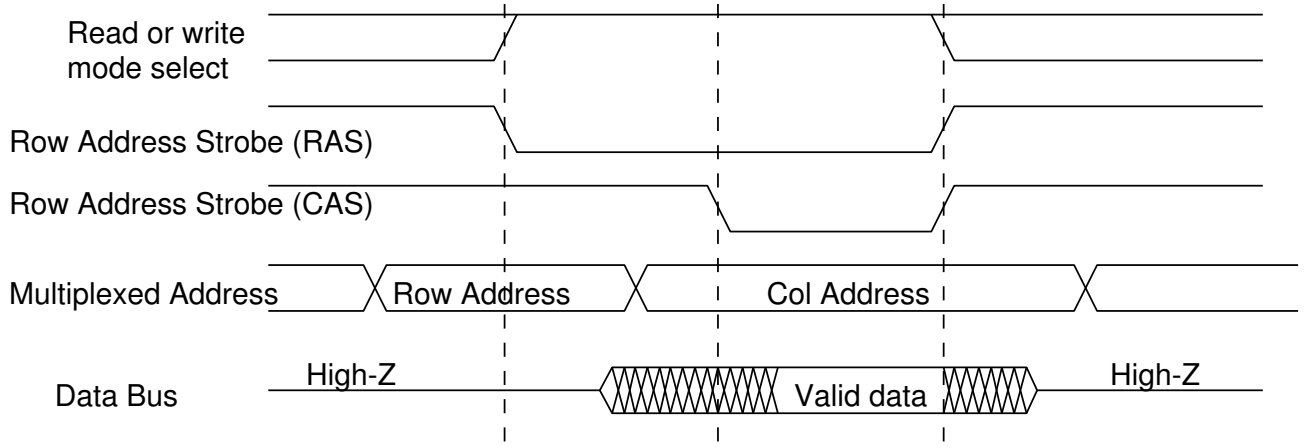


Transparent latch implemented from gates.





Read Cycle (write is similar)

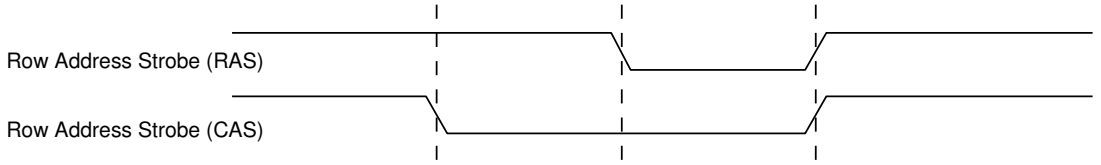


A DRAM has a multiplexed address bus and the address is presented in two halves, known as row and column addresses. So the capacity is 4**A x D. A 4 Mbit DRAM might have A=10 and D=4.

When a processor (or its cache) wishes to read many locations in sequence, only one row address needs be given and multiple col addresses can be given quickly to access data in the same row. This is known as 'page mode' access.

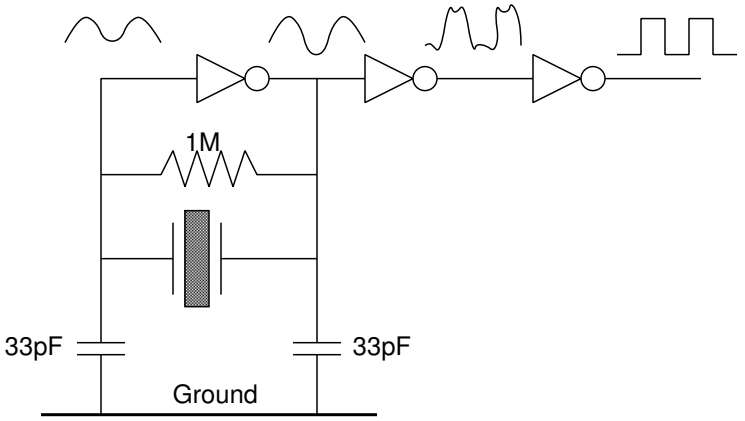
EDO (extended data out) DRAM is now quite common. This guarantees data to be valid for an exteneded period after CAS, thus helping system timing design at high CAS rates.

Refresh Cycle - must happen sufficiently often!

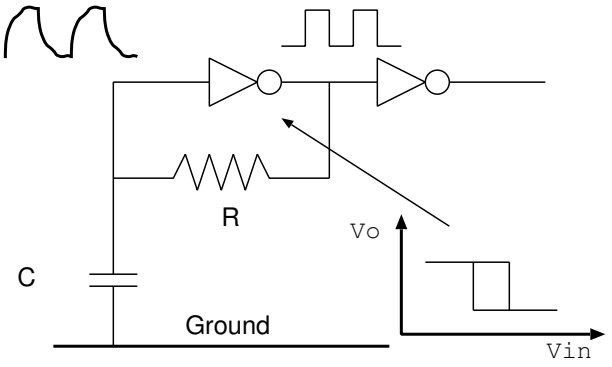


No data enters or leaves the DRAM during refresh, so it 'eats memory bandwidth'.
Typically 512 cycles of refresh must be done every 8 milliseconds.

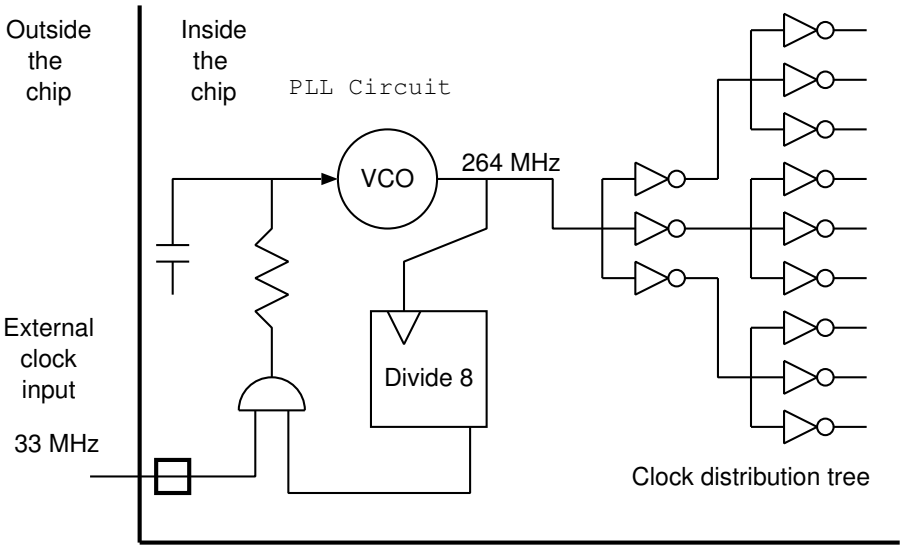
Crystal oscillator clock source



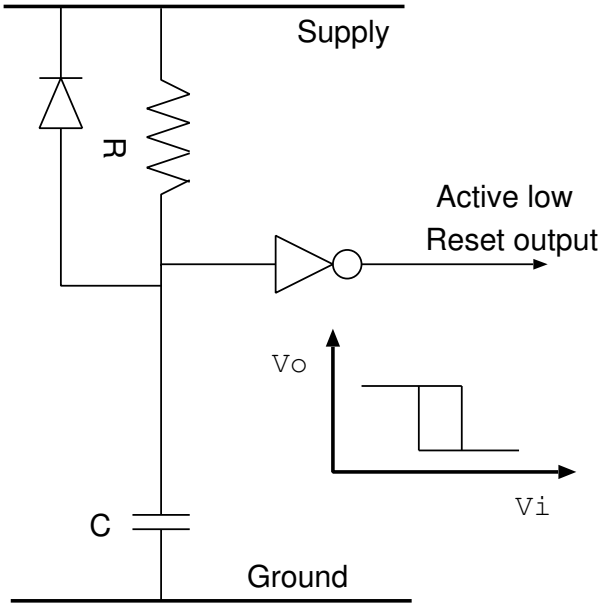
RC oscillator clock source



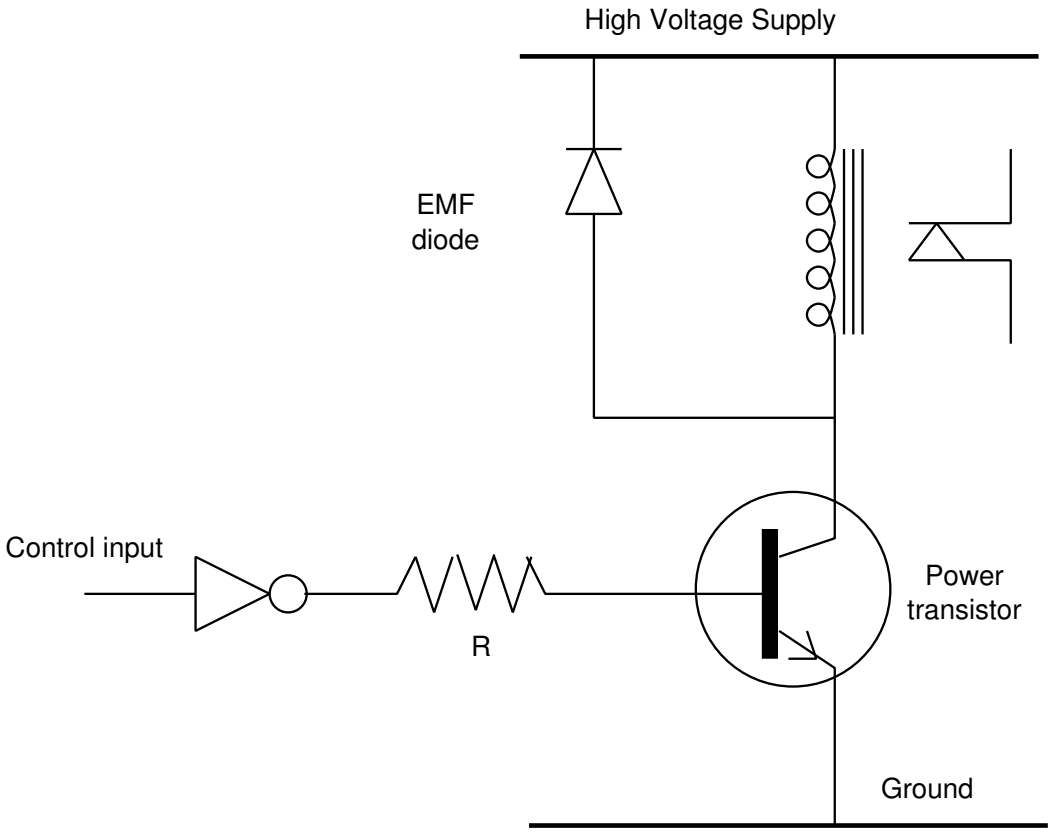
Clock multiplication and distribution



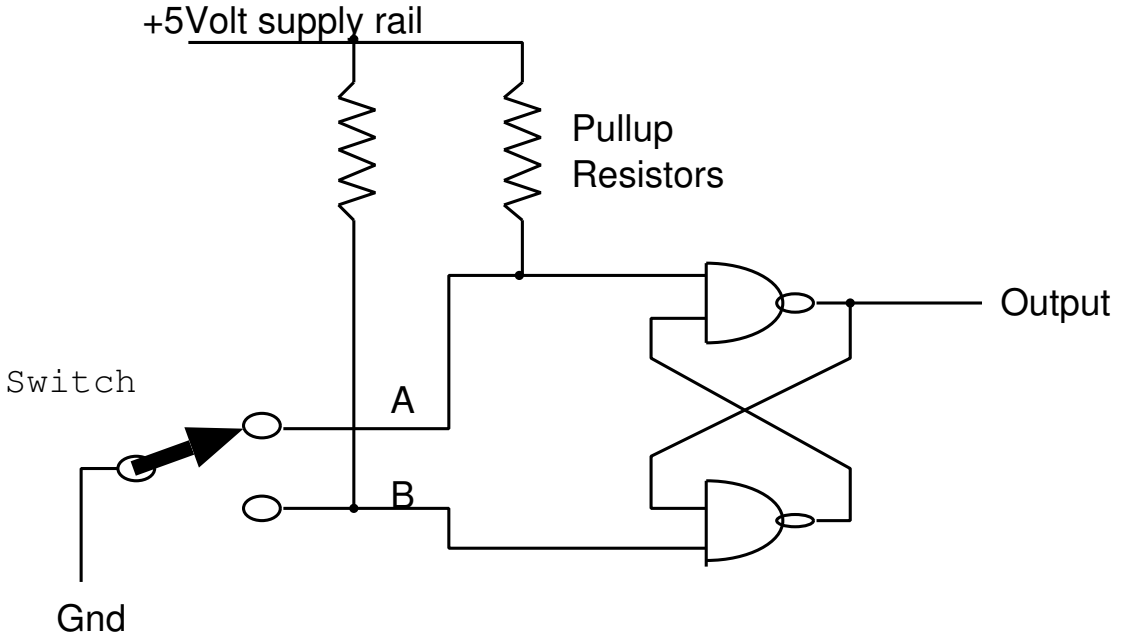
Power-on reset

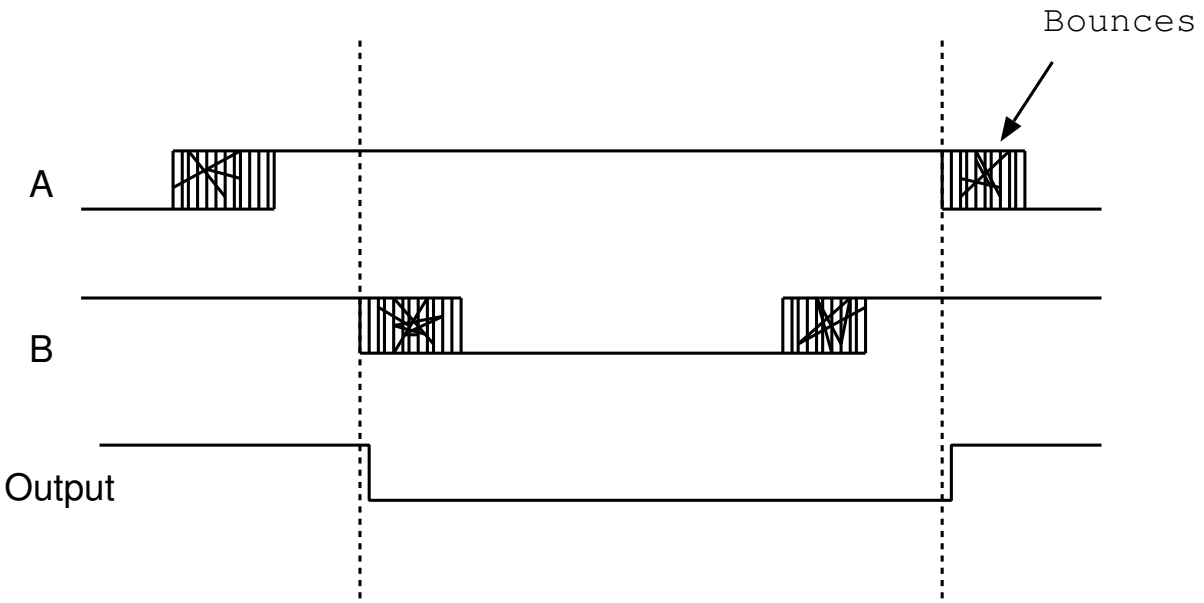


Driving a heavy current or high-voltage load

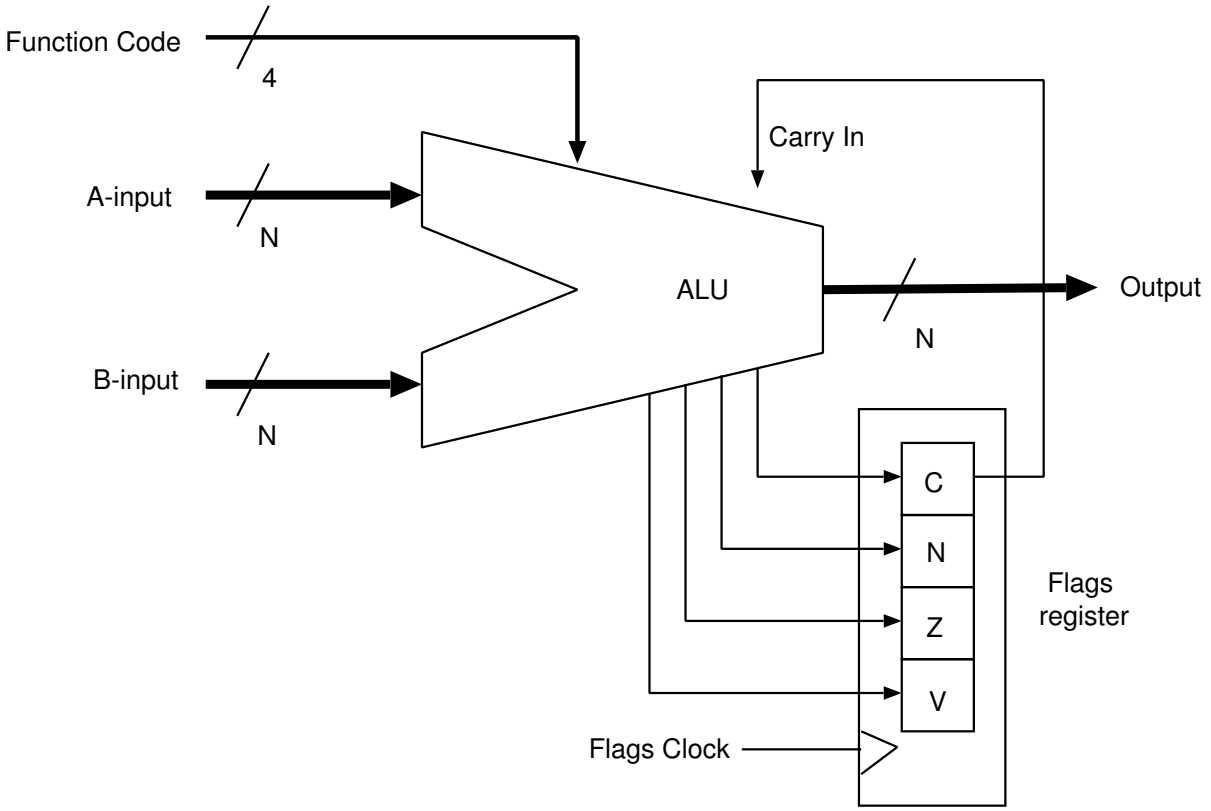


Debouncer circuit for a two-pole switch

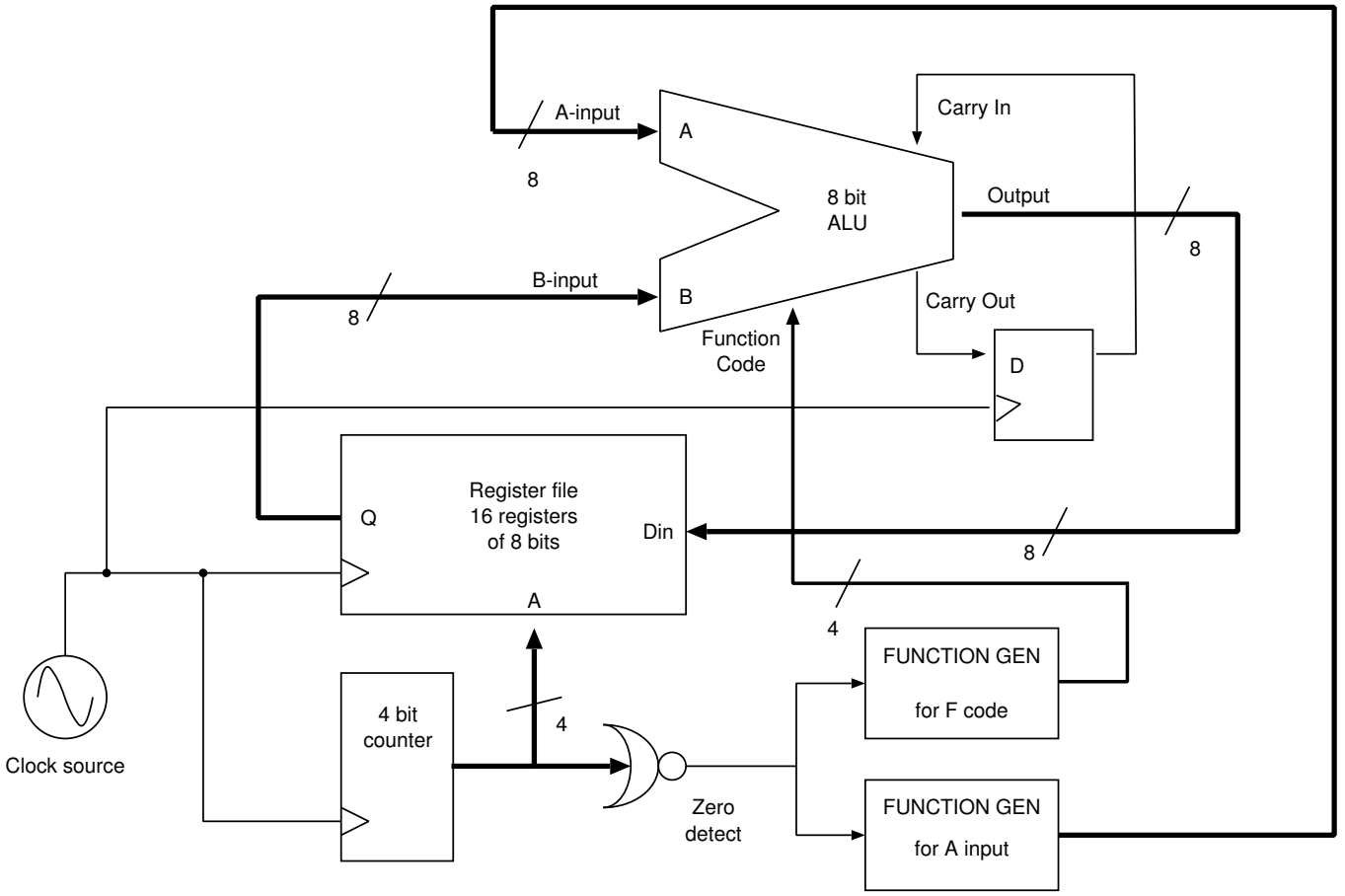




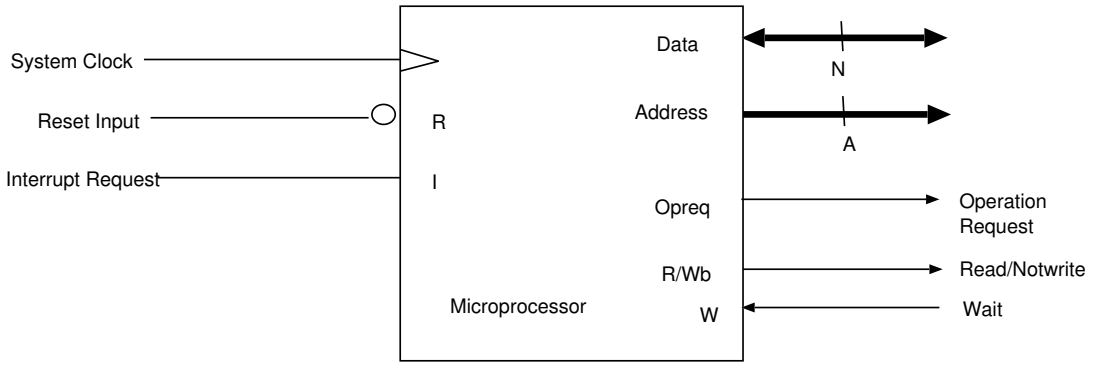
ALU and flags register



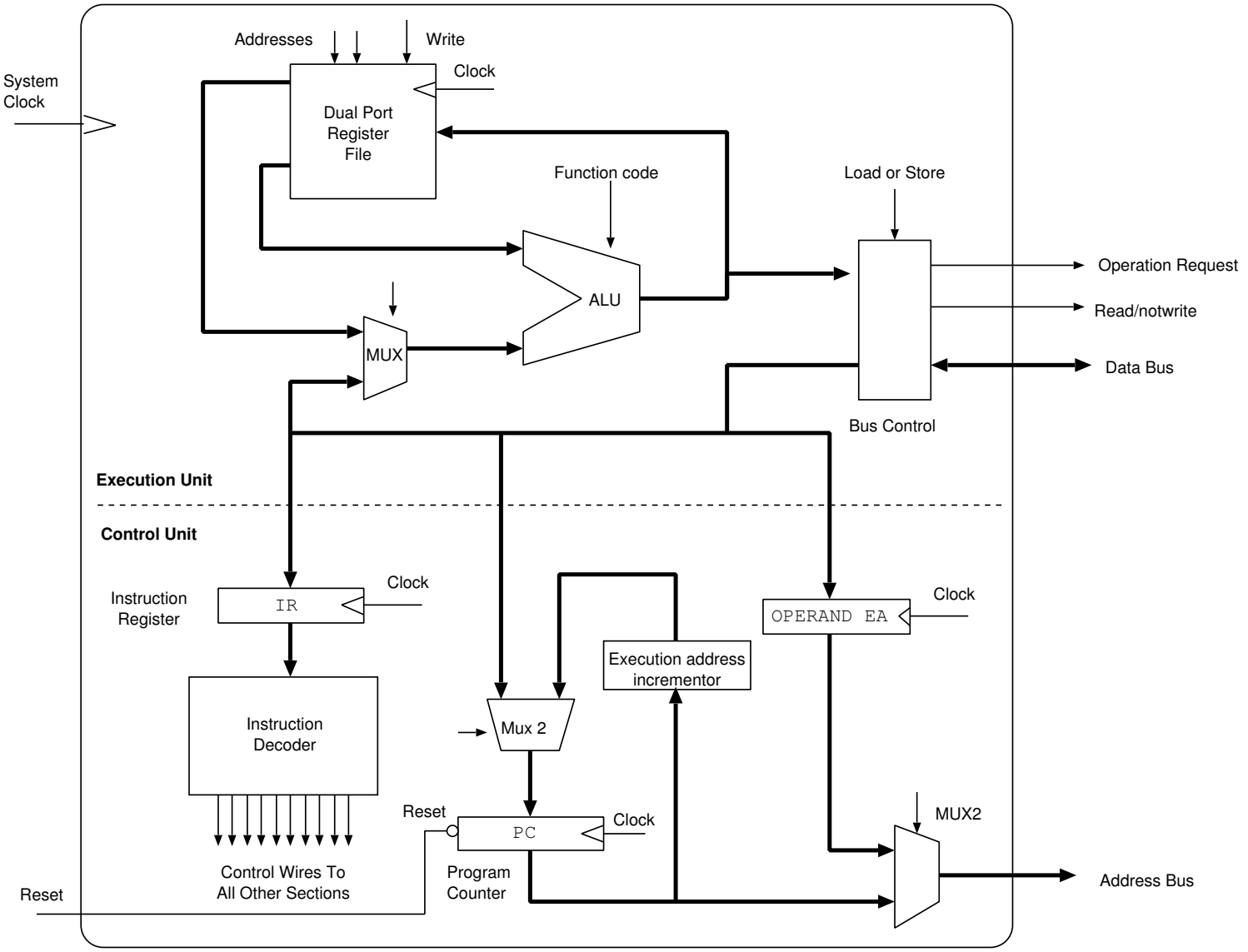
ALU and register file

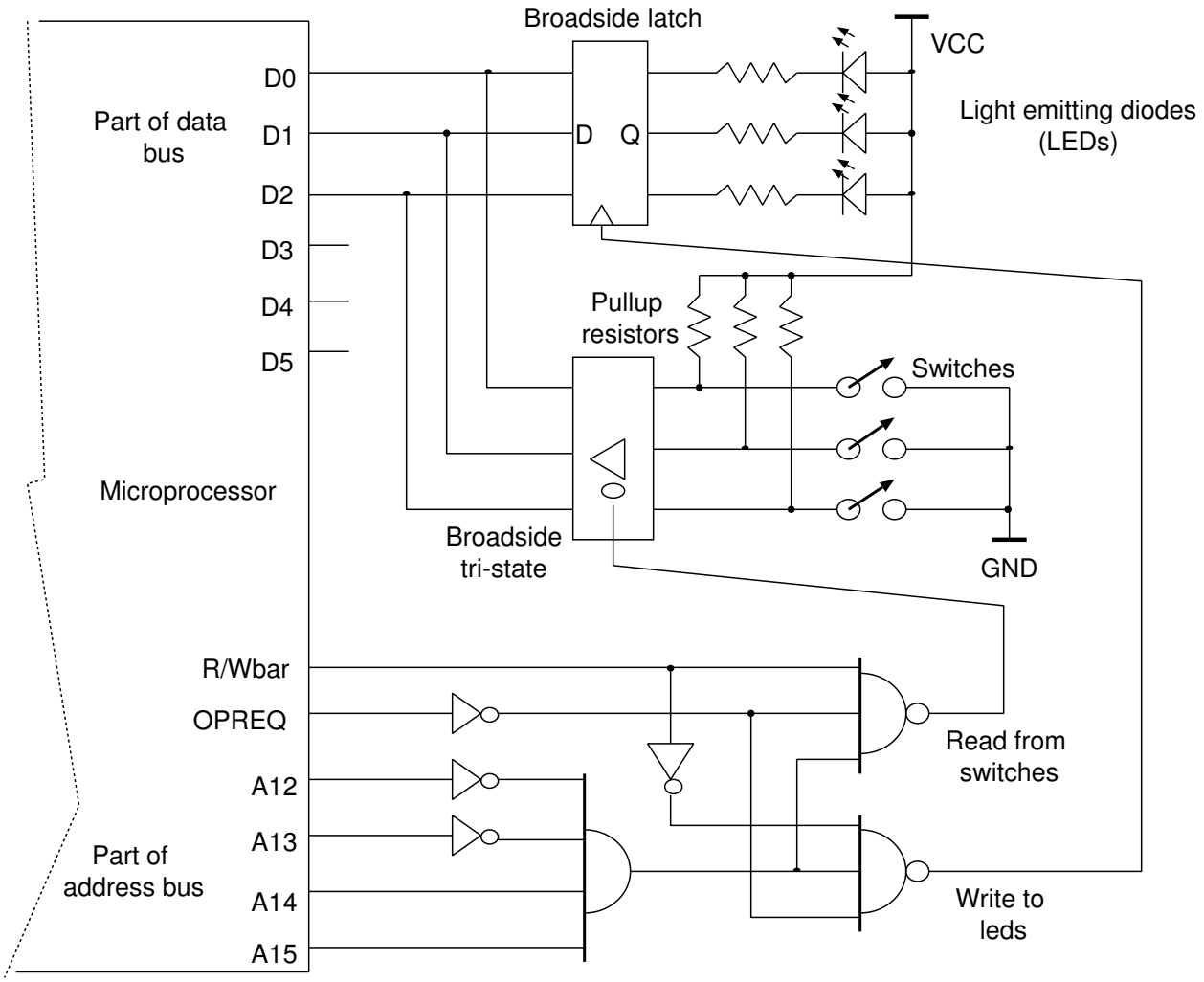


Logic Symbol



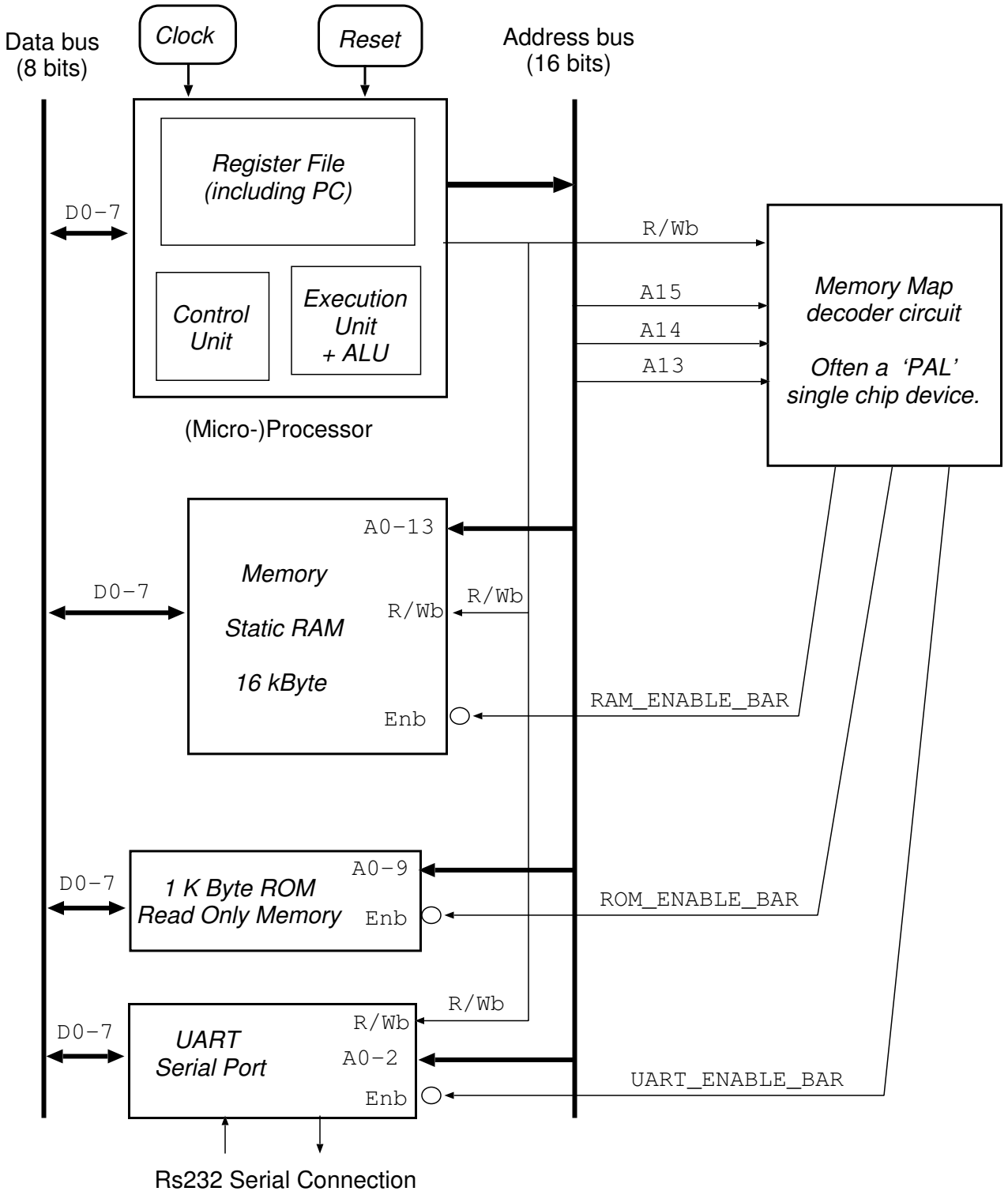
Internal Structure Block Diagram



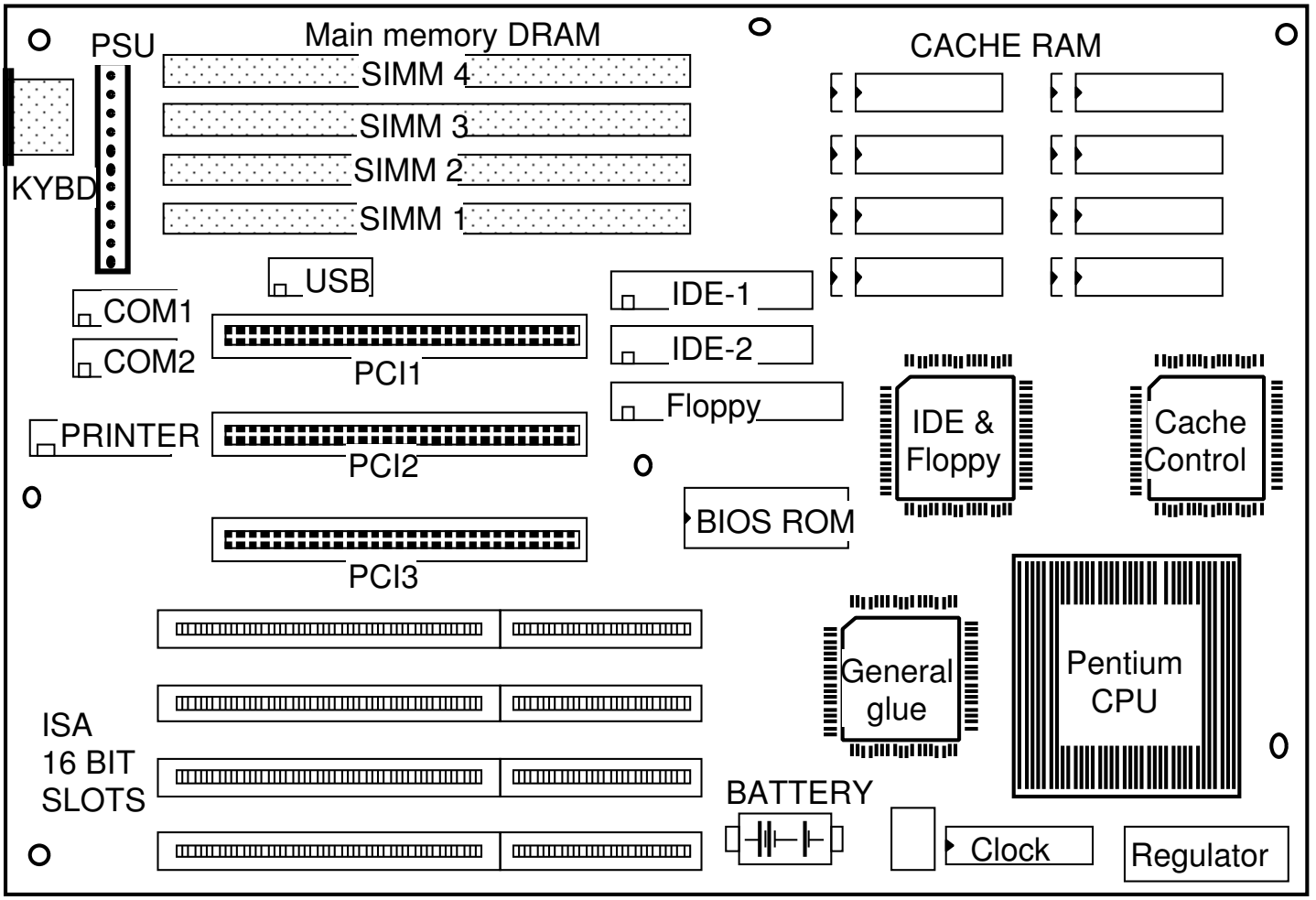


Example of memory address decode and simple LED and switch interfacing for programmed IO (PIO) to a microprocessor.

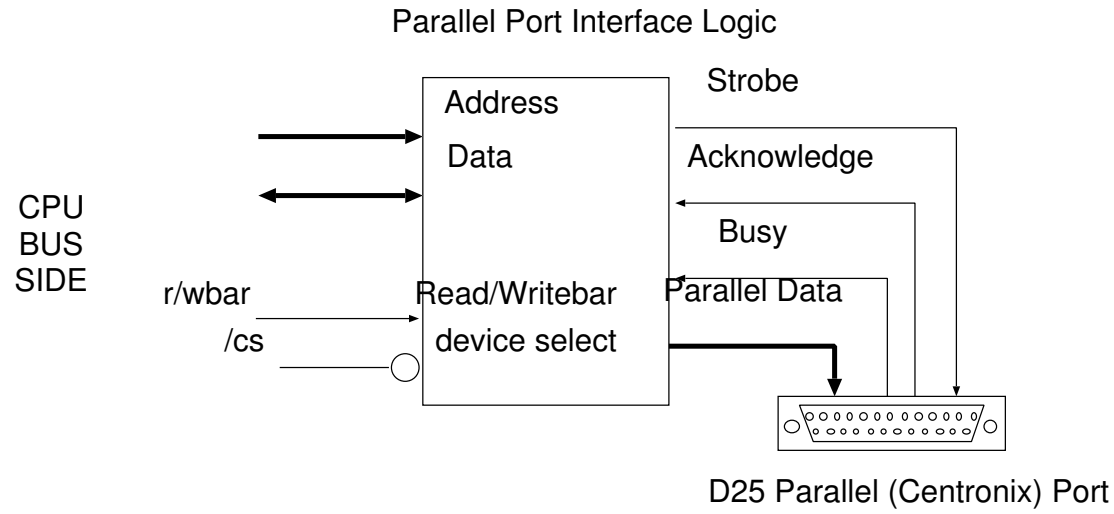
A small computer

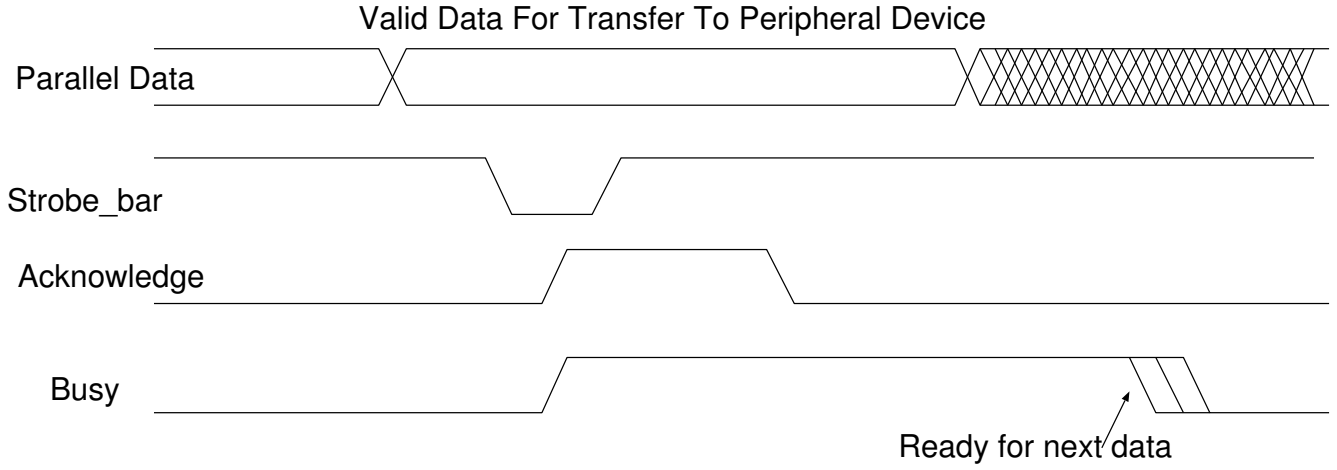


PC Motherboard, 1997 vintage



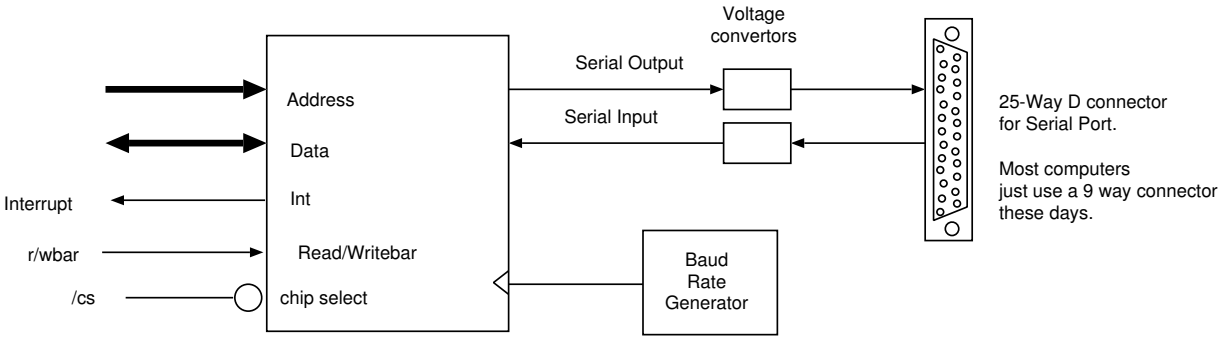
Parallel Port

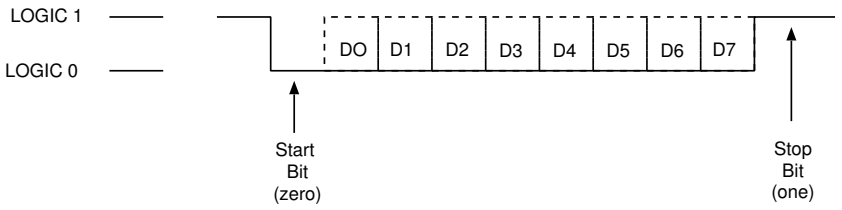




Flow control: New data is not sent while the busy wire is high.

Serial Port (UART)





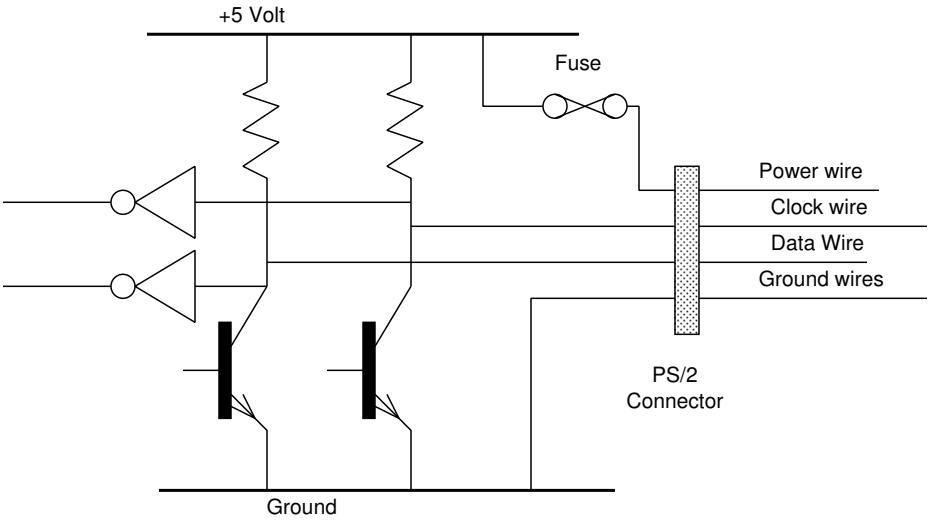
Flow control: New data can be sent at any time unless either:

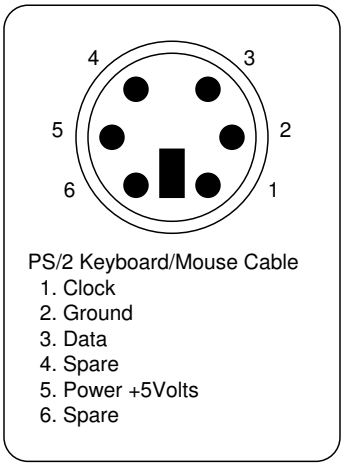
additional signals are used to indicate clear to send

or

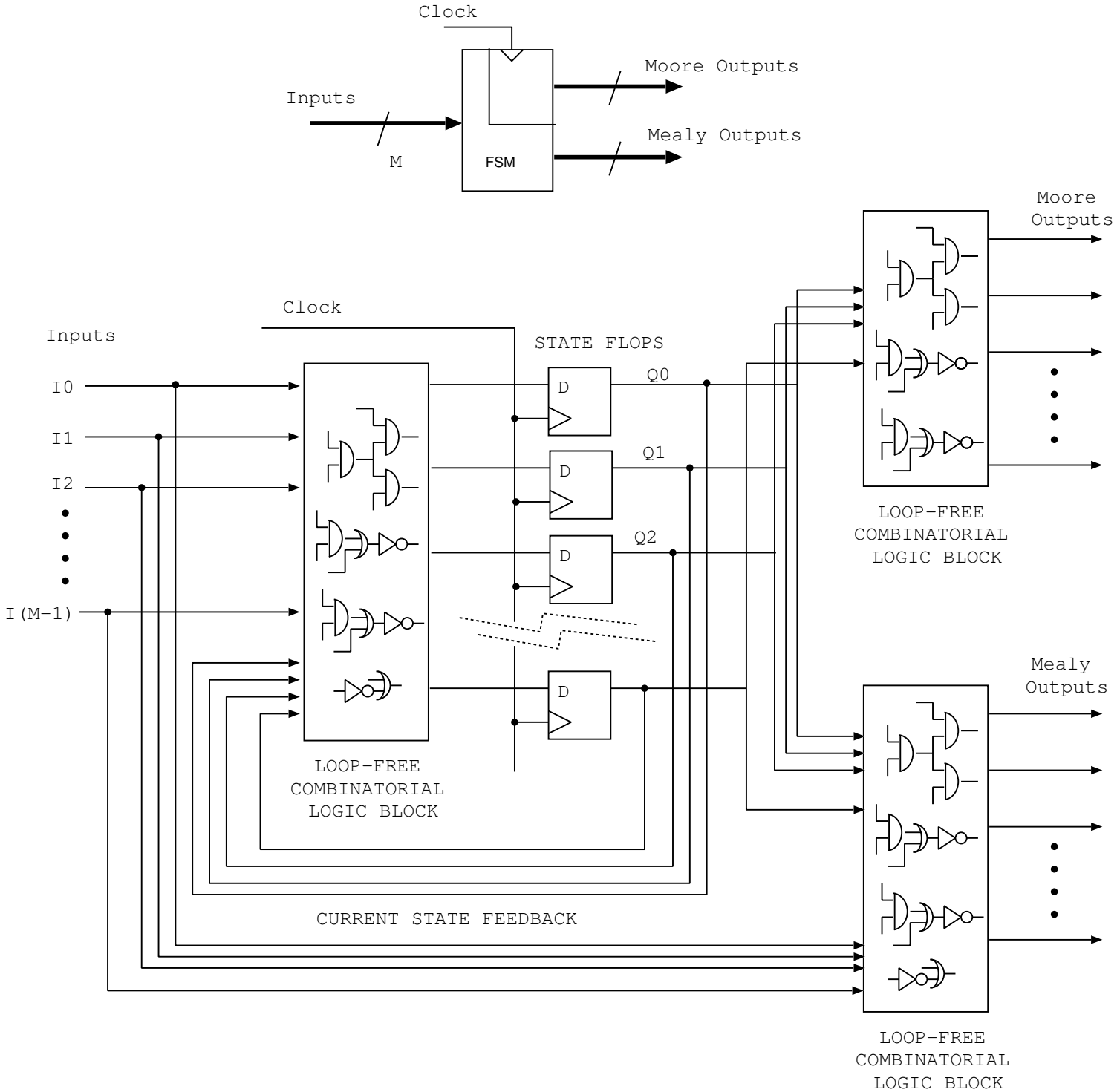
a software protocol is defined to run on top (Xon/Xoff) by reserving certain of the bytes.

Keyboard and/or PS/2 port

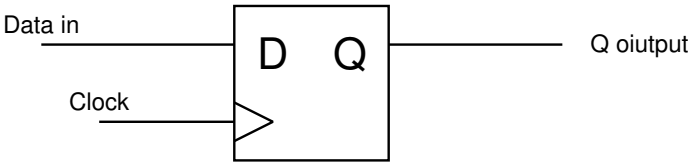


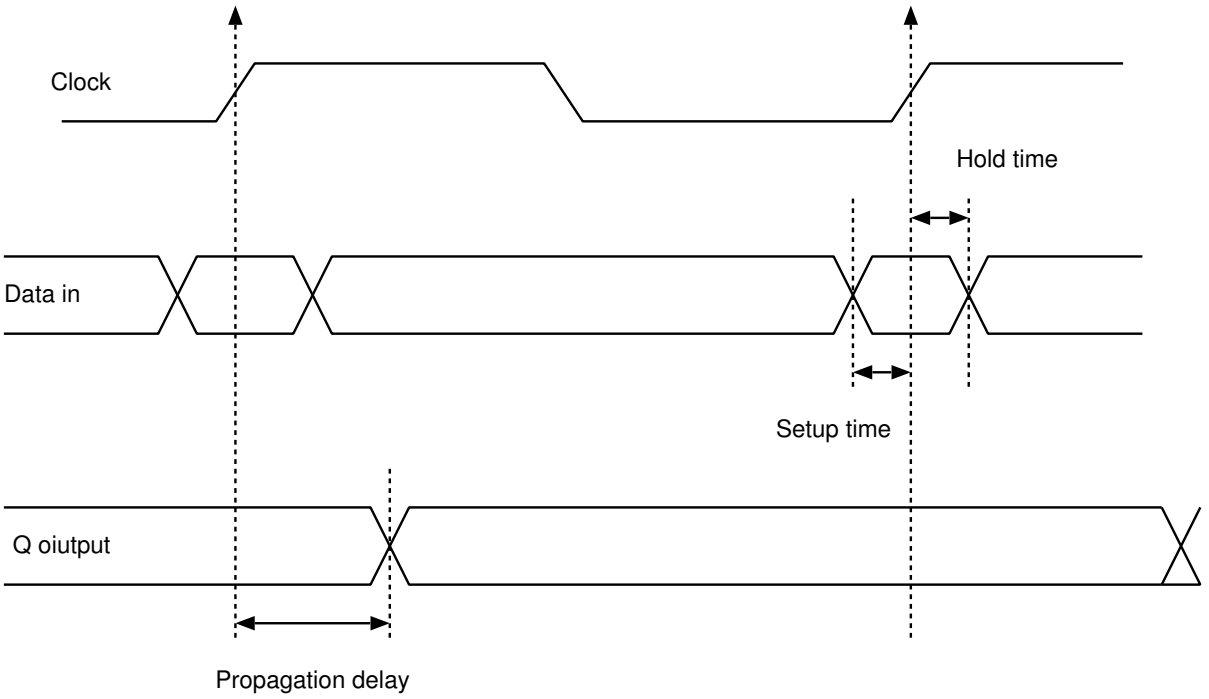


Canonical synchronous FSM

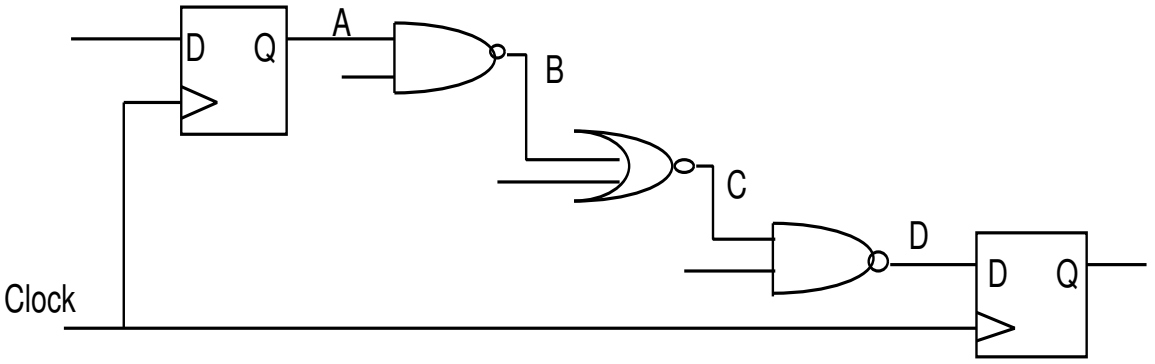


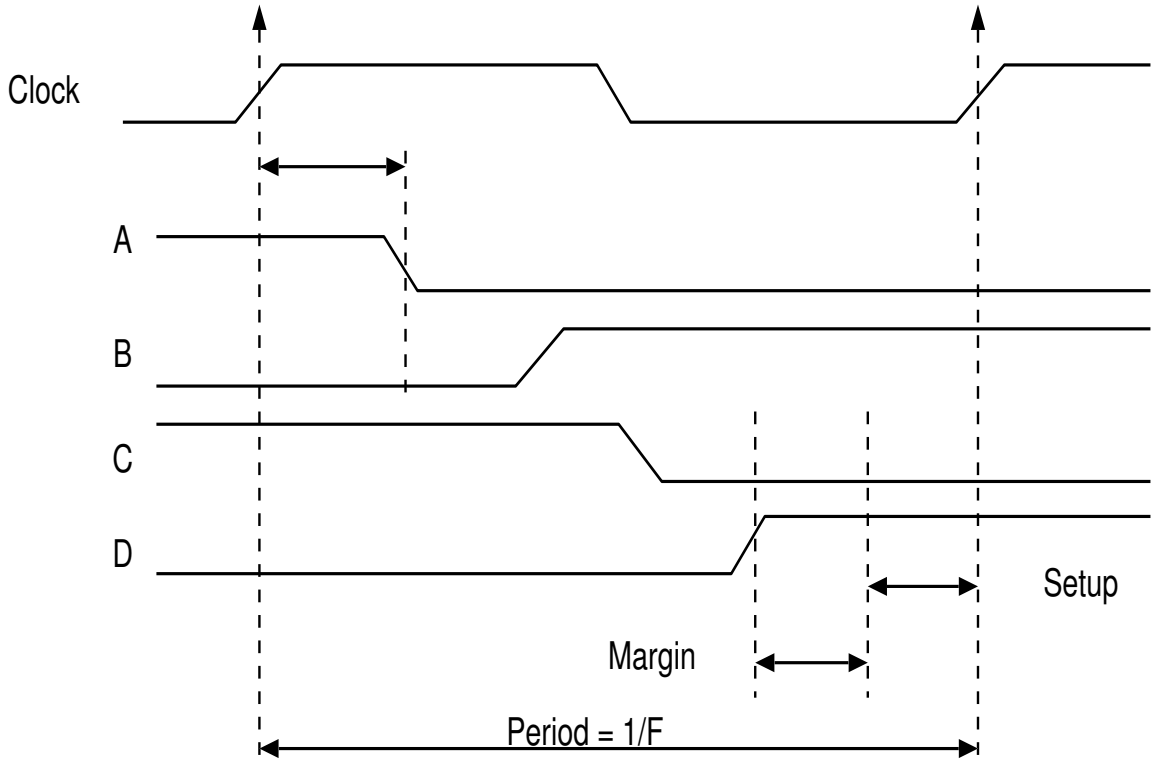
Timing Specifications



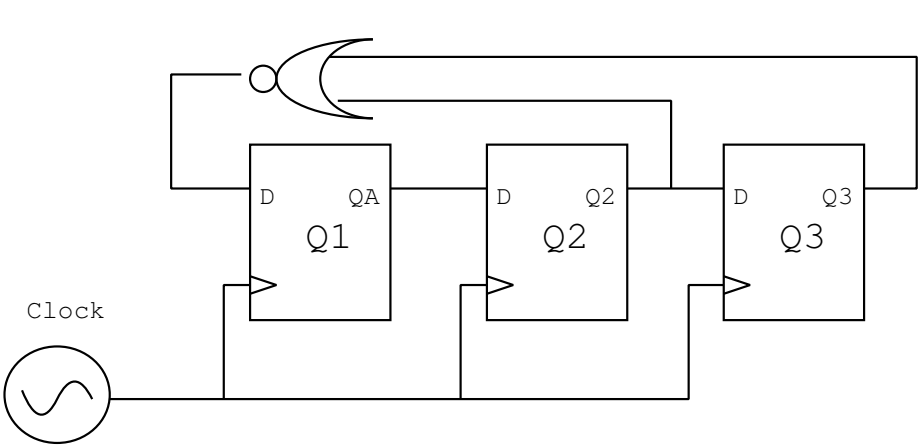


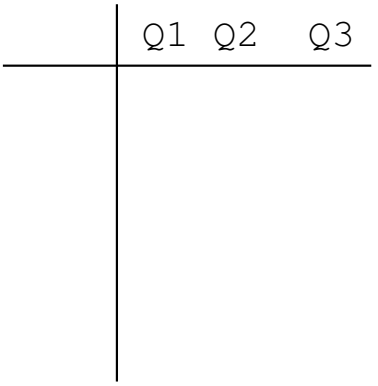
Typical nature of a critical path



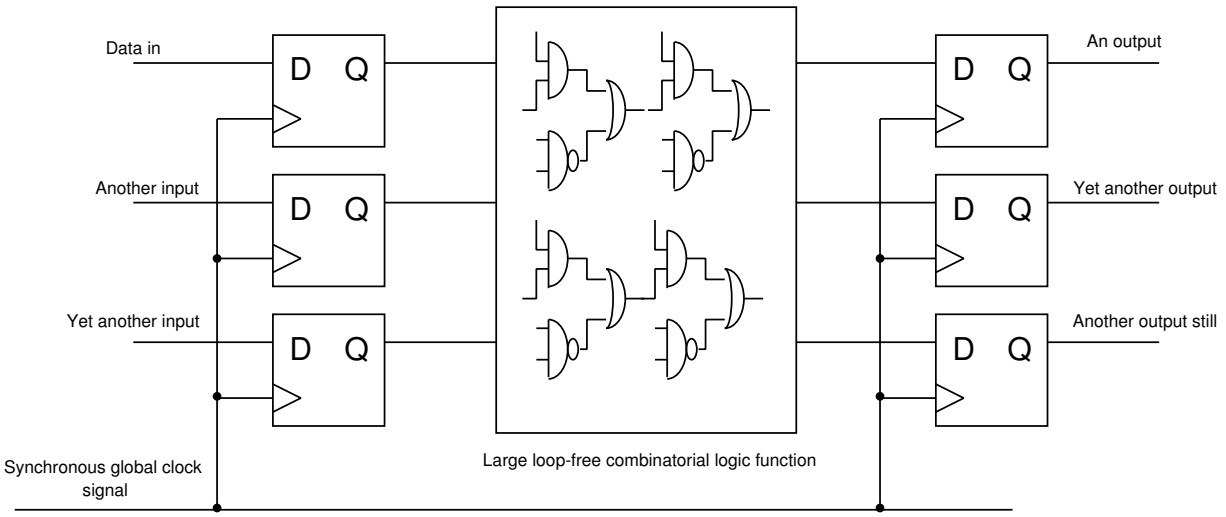


Johnson counters

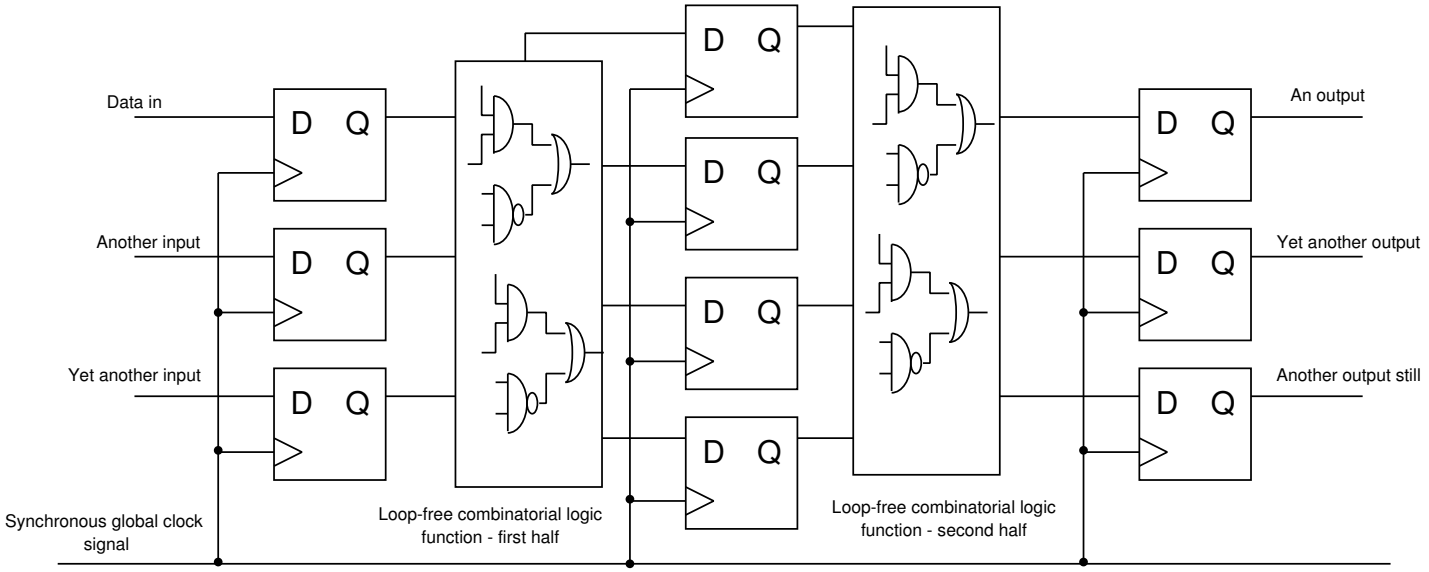




Pipelining

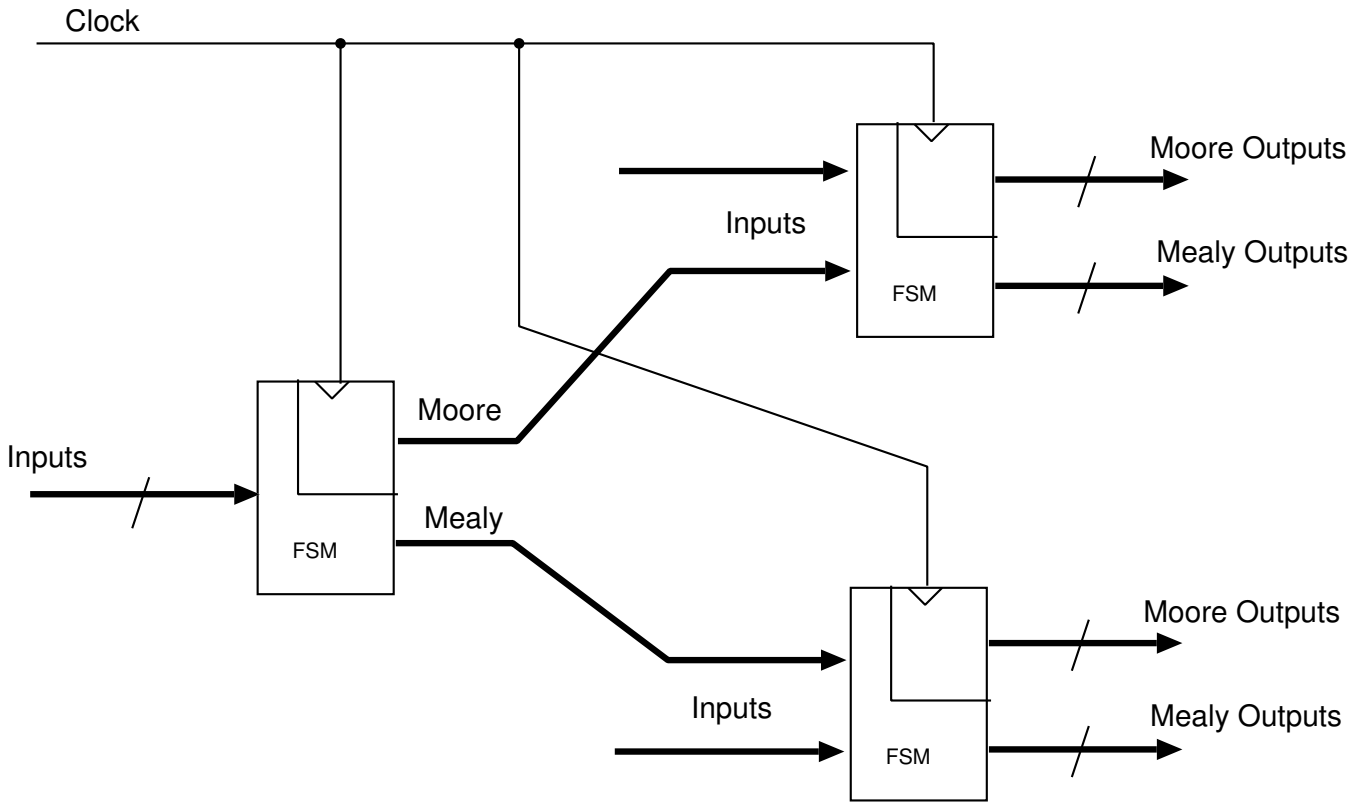


Desired logic function

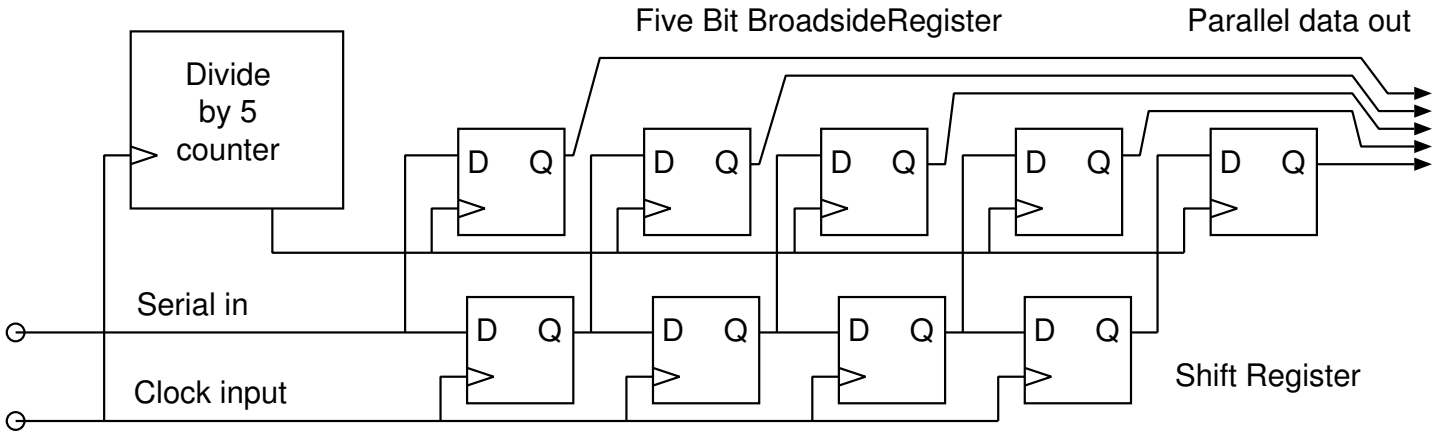


Desired logic function - pipelined version.

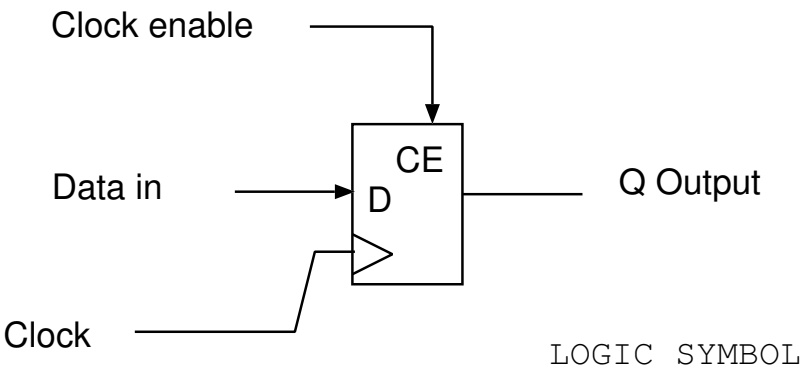
Cascading FSMs



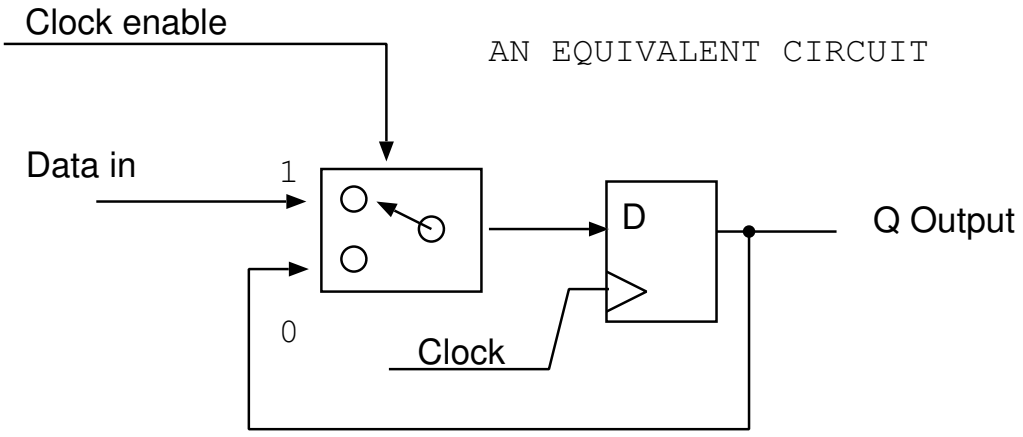
An example that uses (badly) a derived clock: a serial-to-parallel converter



A D-type with clock-enable

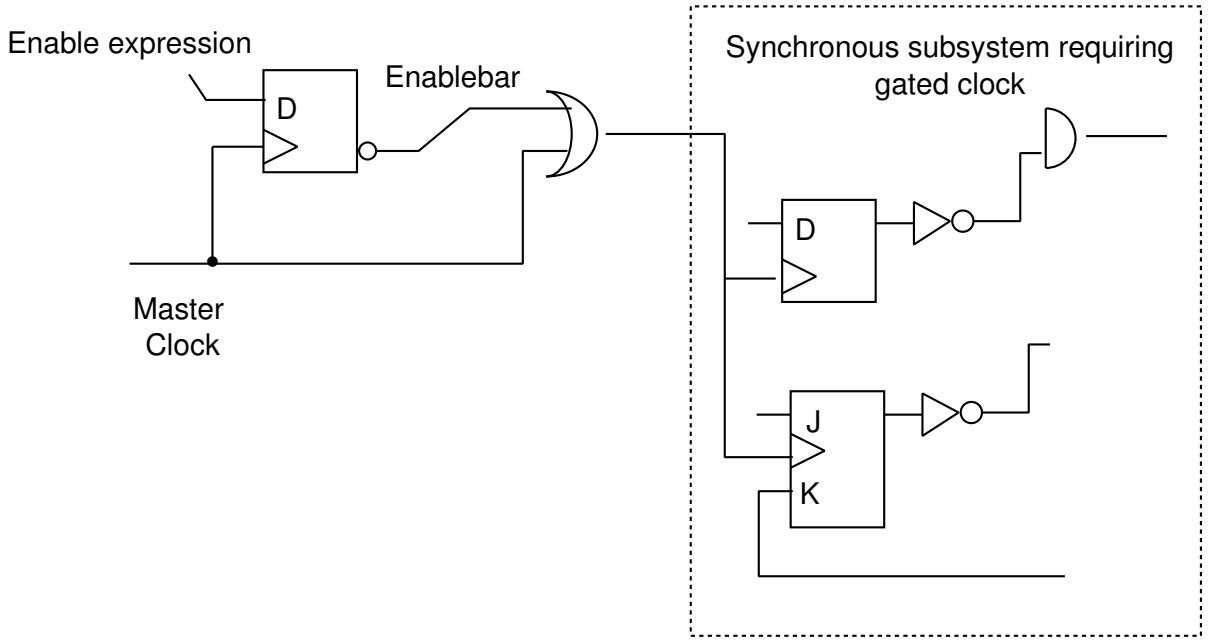


LOGIC SYMBOL

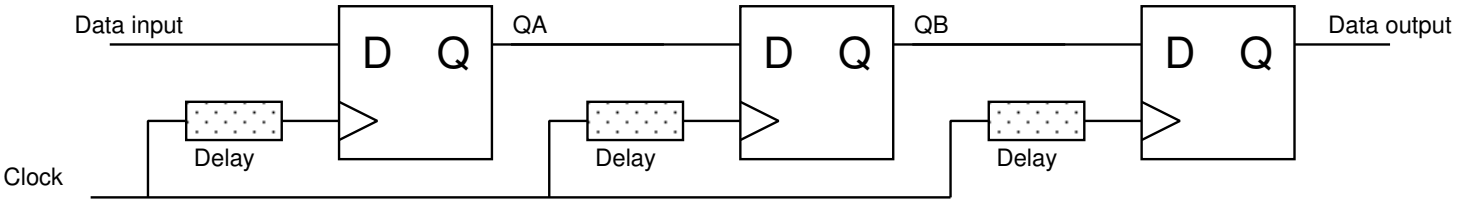


AN EQUIVALENT CIRCUIT

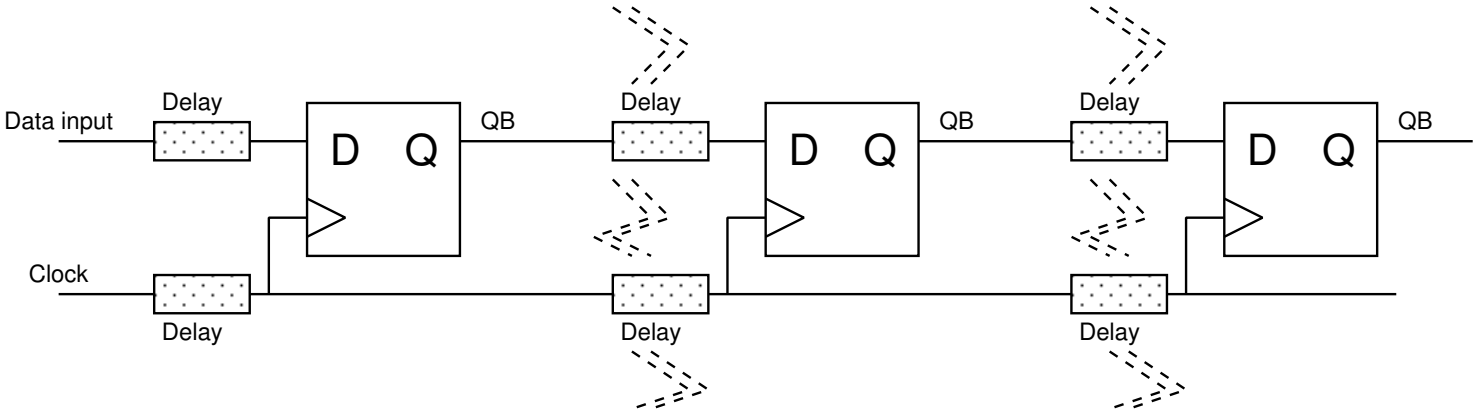
A Gated Clock



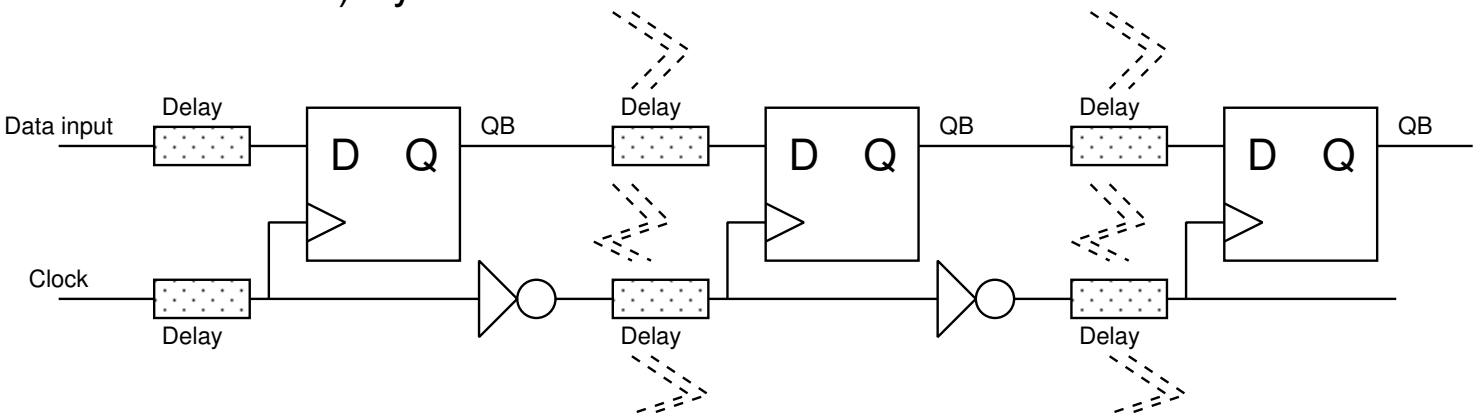
Clock Skew



a) A three-stage shift register with some clock skew delays.

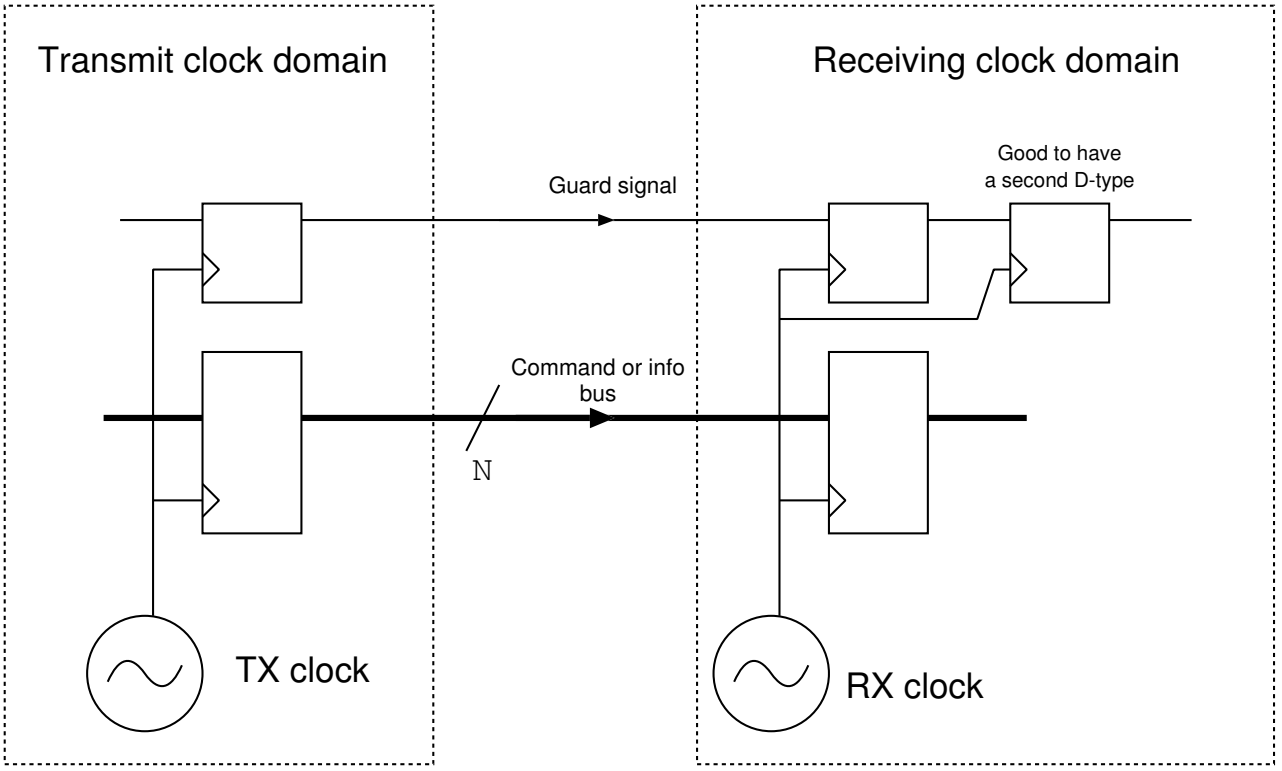


b) System interconnection with clock skews



c) A solution for serious skew and delay problems ?

Crossing an async boundary

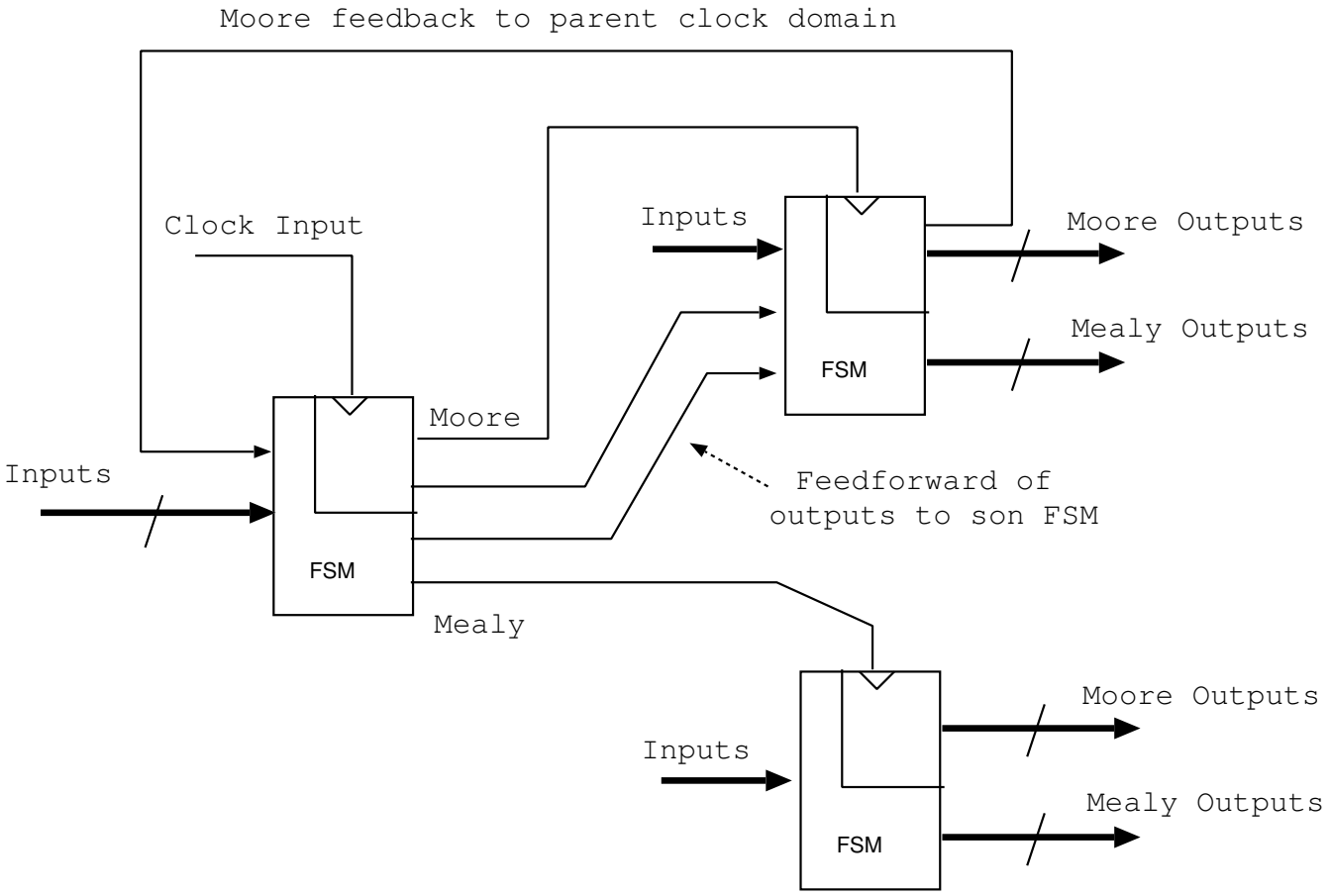


1. The wider the bus width, N, the fewer the number of transactions per second needed and the greater the timing flexibility in reading the data from the receiving latch.

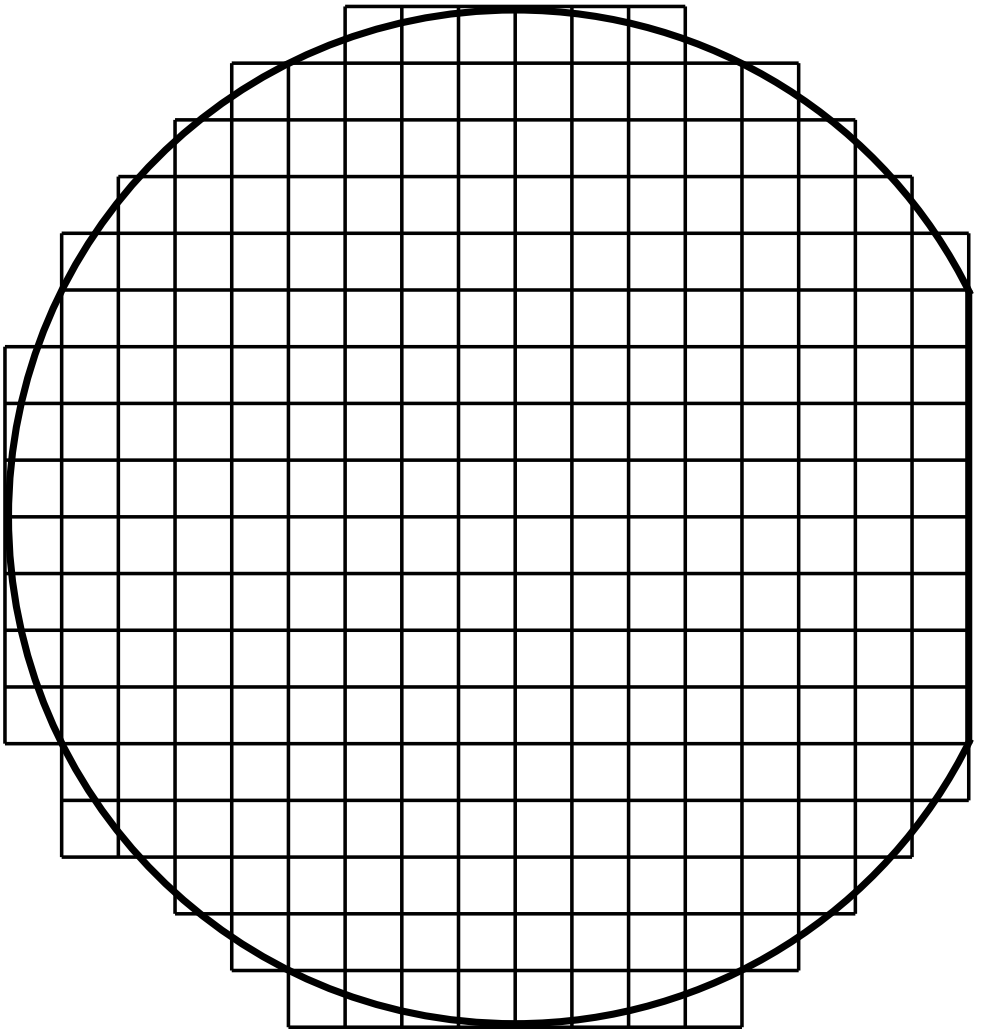
2. Make sure that the transmitter does not change the guard and the data in the same transmit clock cycle.

3. Place a second flip-flop after the receiving decision flip-flop so that on the rare occurances when the first is metastable for a significant length of time (e.g. 1/2 a clock cycle) the second willpresent a good clean signal to the rest of the receiving system.

Paths between FSMs w/ derived clocks

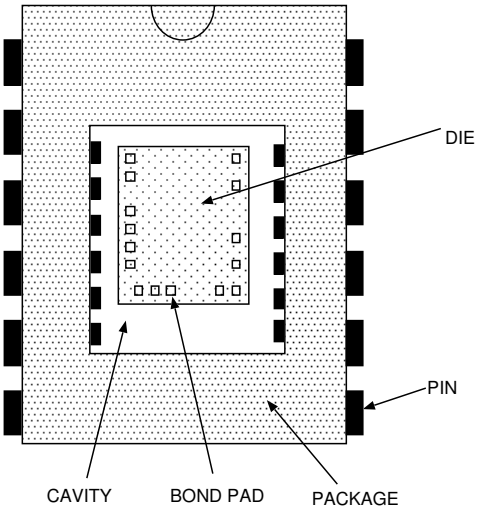


Dicing a wafer

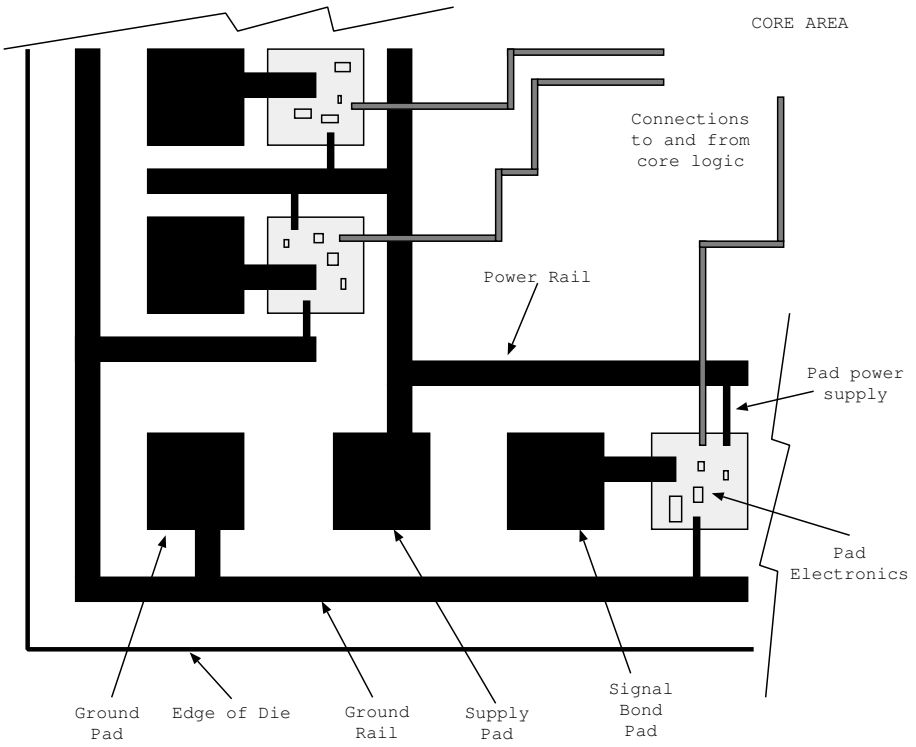


(Chips are not always square)

A chip in its package, ready for bond wires



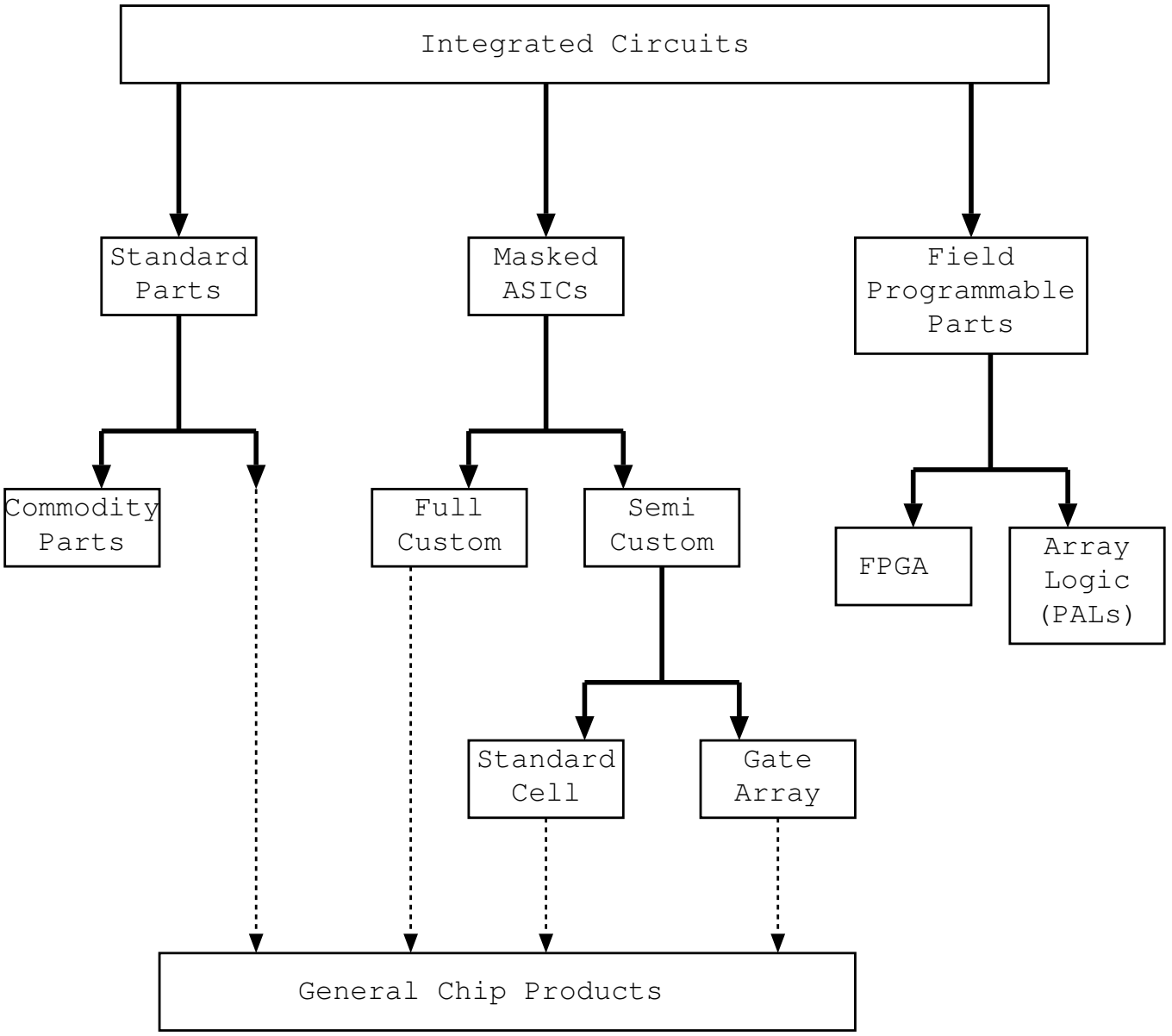
IO and power pads



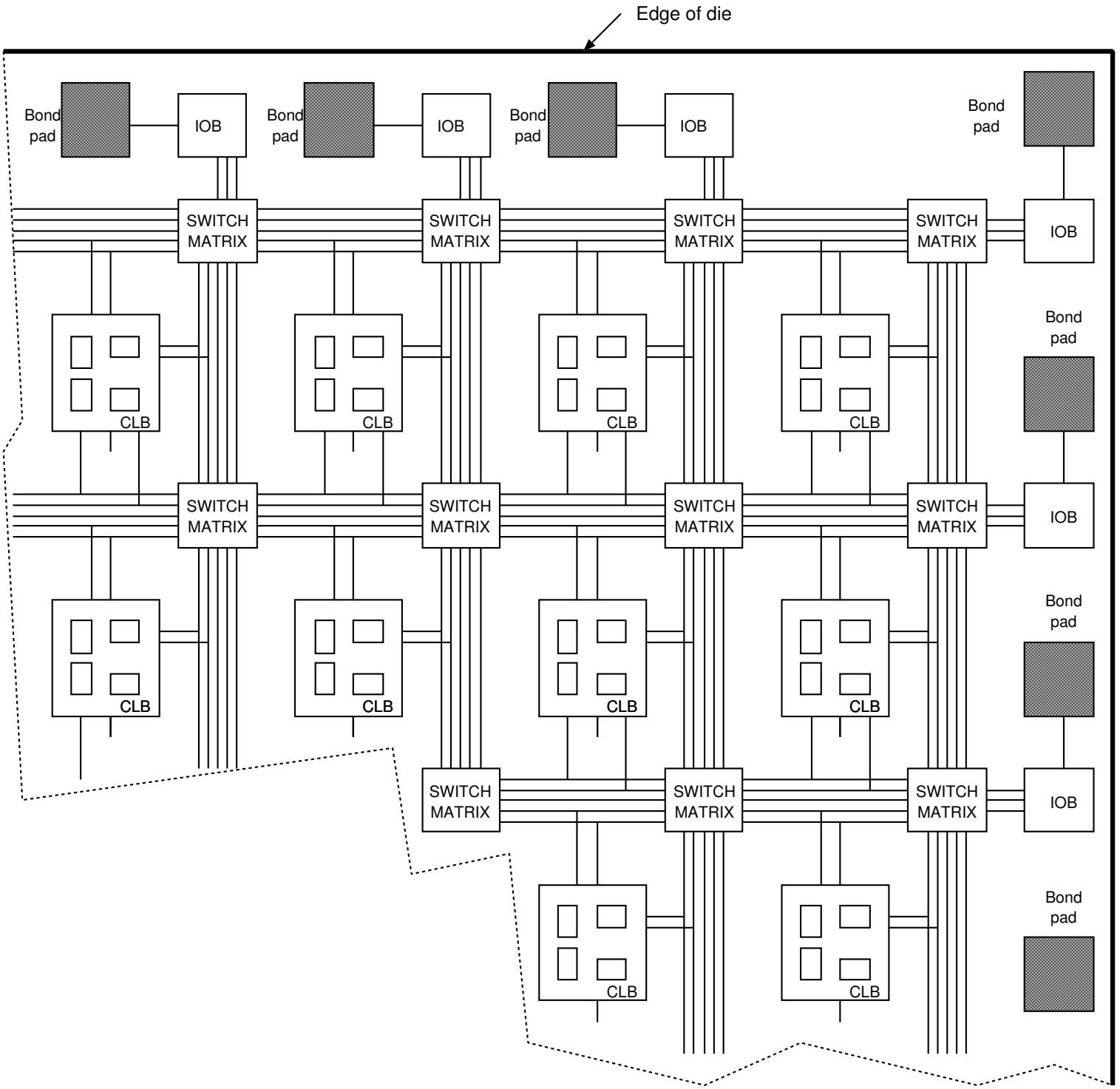
Die cost example

Area	Wafer dies	Working dies	Cost per working die
2	9000	8910	0.56
3	6000	5910	0.85
4	4500	4411	1.13
6	3000	2911	1.72
9	2000	1912	2.62
13	1385	1297	3.85
19	947	861	5.81
28	643	559	8.95
42	429	347	14.40
63	286	208	24.00
94	191	120	41.83
141	128	63	79.41
211	85	30	168.78
316	57	12	427.85
474	38	4	1416.89

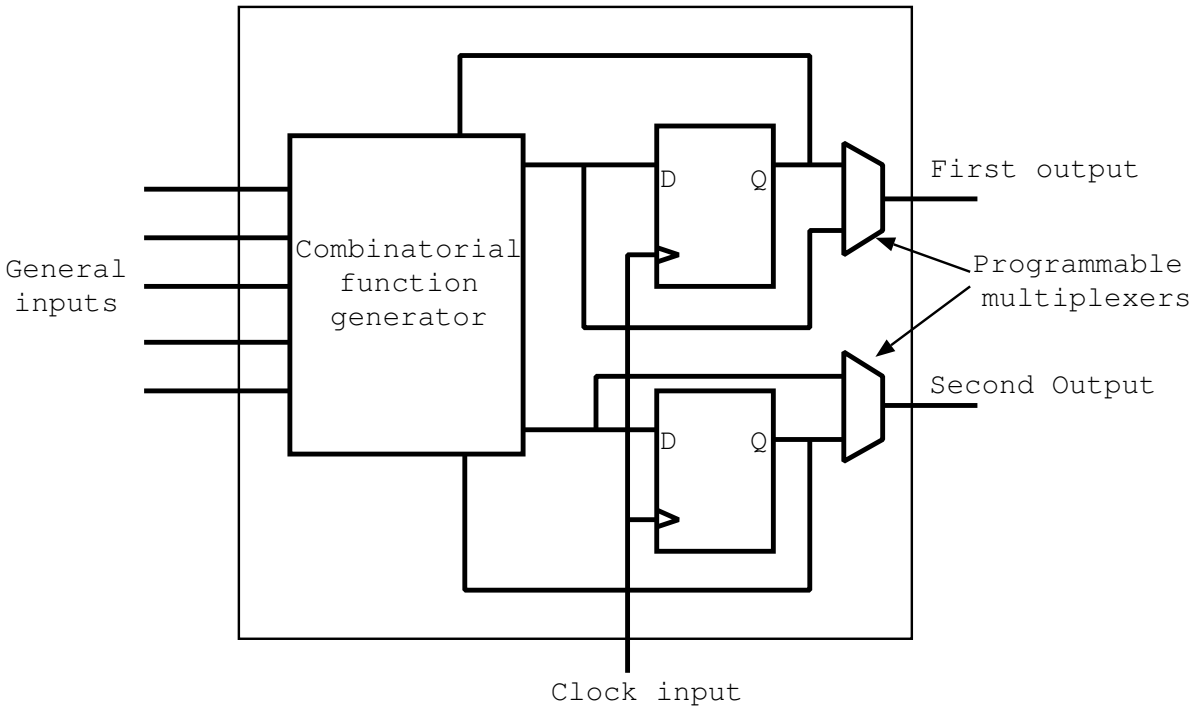
A taxonomy of ICs



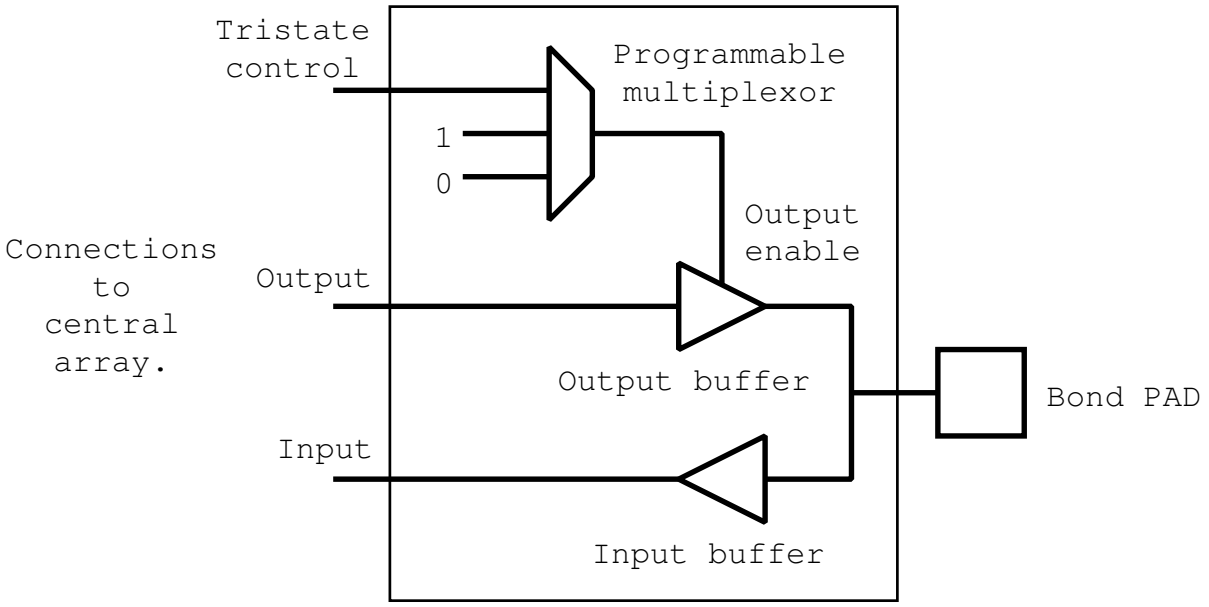
Field Programmable Gate Arrays

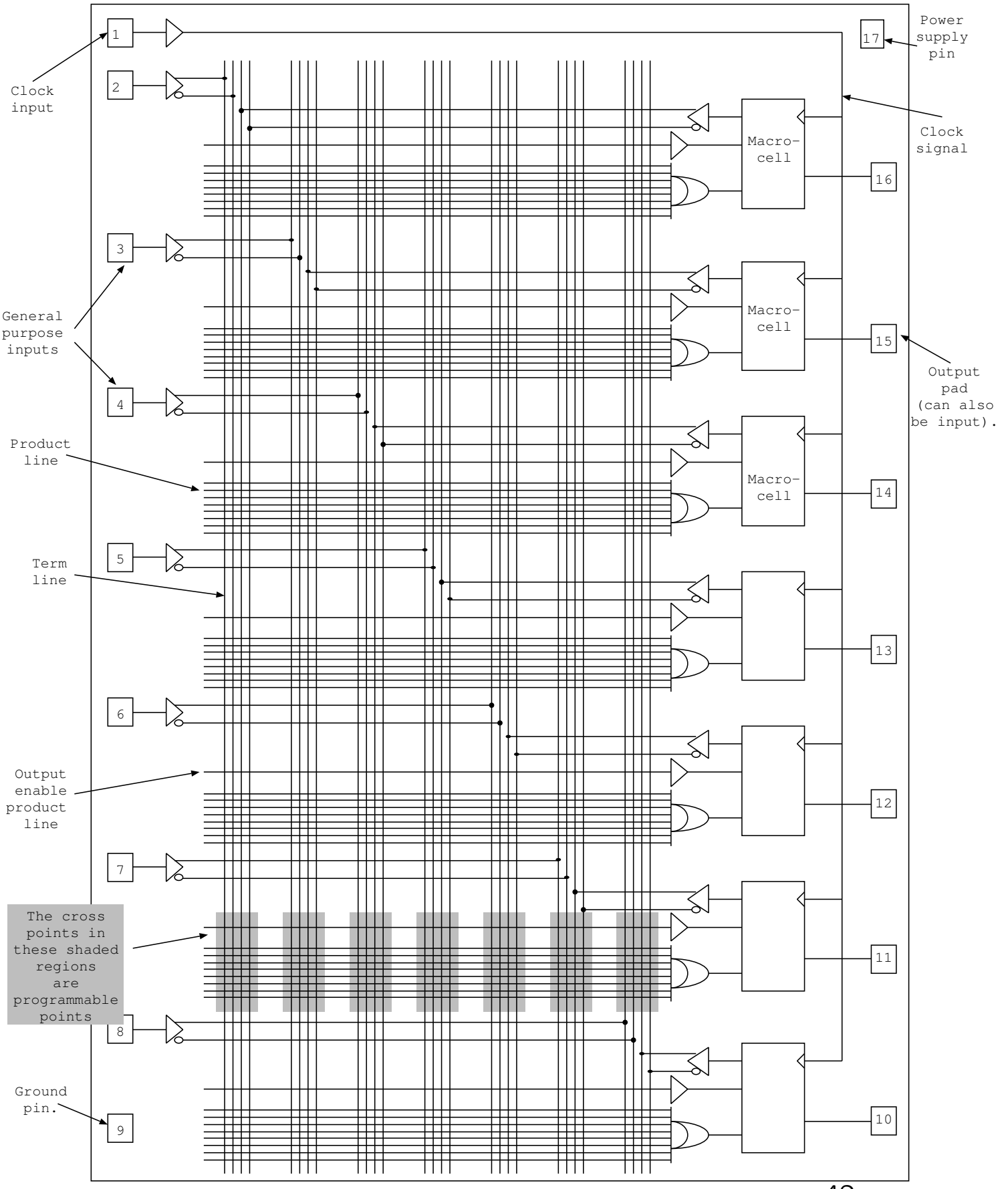


A configurable logic block for a look-up-table based FPGA

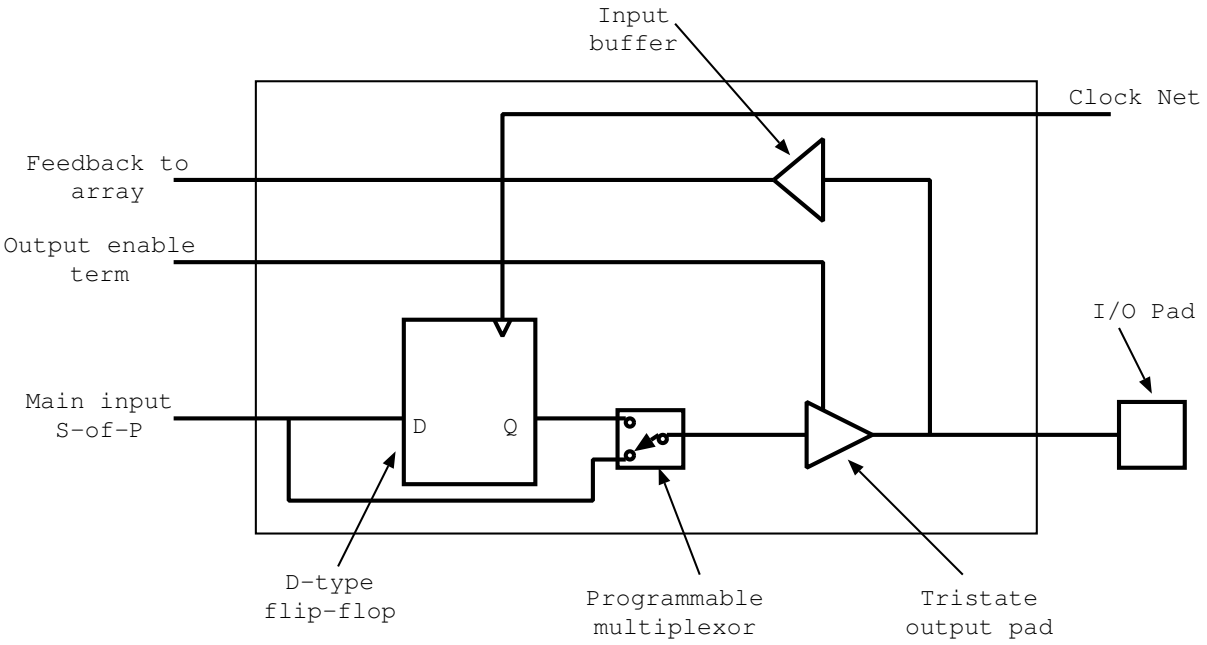


A simple IO block FPGA




Clock input
1
2
3
General purpose inputs
4
Product line
5
Term line
6
Output enable product line
7
The cross points in these shaded regions are programmable points
8
Ground pin.
9
17
Power supply pin
Clock signal
Macro-cell
16
Macro-cell
15
Output pad (can also be input).
Macro-cell
14
13
12
11
10

Contents of the PAL macrocell



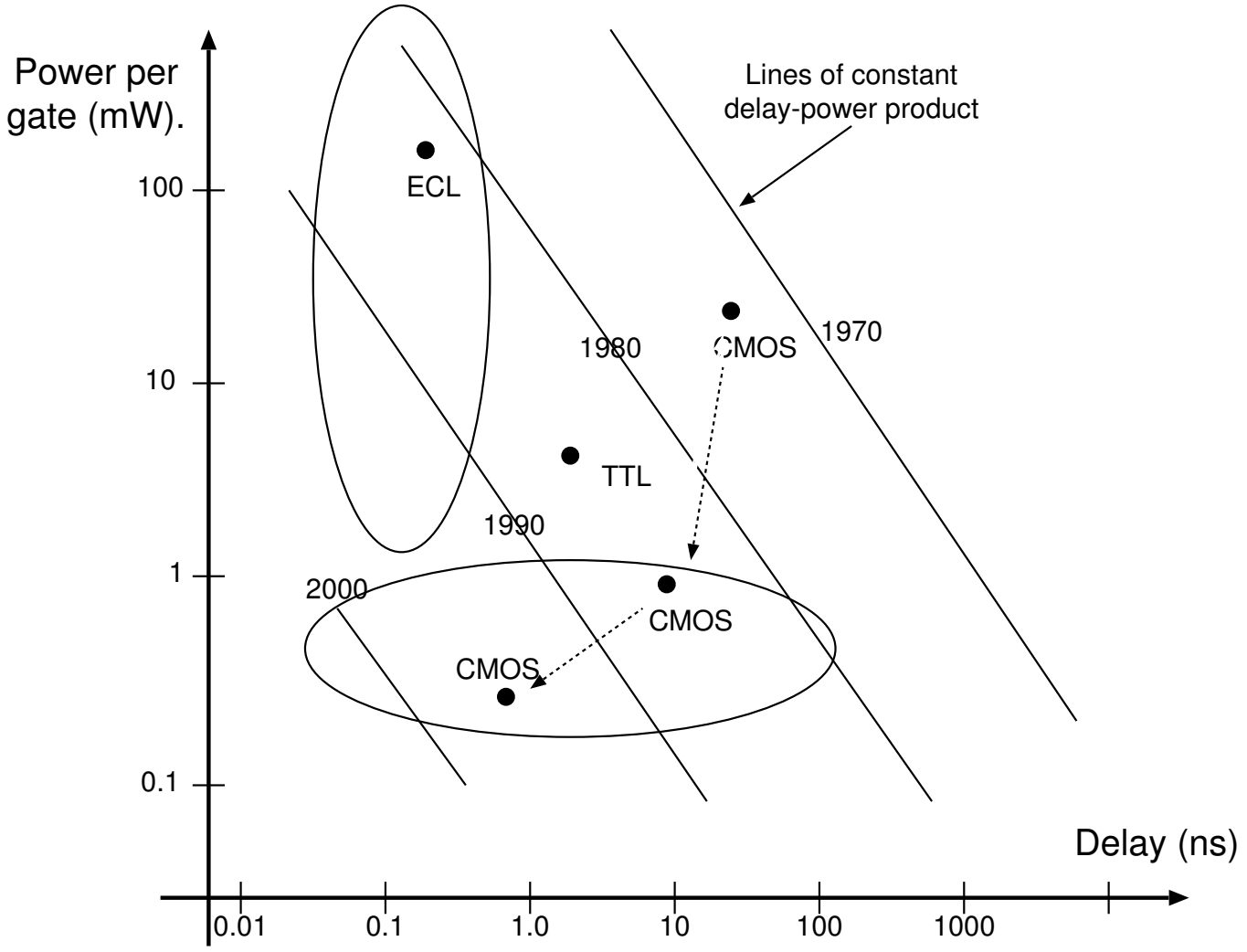
Example programming of a PAL showing only fuses for the top macrocell

```
pin 16 = o1;
pin 2 = a;
pin 3 = b;
pin 4 = c

o1.oe = ~a;
o1 = (b & o1) | c;
```

```
-x-- ---- ---- ---- ---- ---- ----  (oe term)
--x- x--- ---- ---- ---- ---- ----  (pin 3 and 16)
---- ---- x--- ---- ---- ---- ----  (pin 4)
xxxx xxxx xxxx xxxx xxxx xxxx xxxx
xxxx xxxx xxxx xxxx xxxx xxxx xxxx
xxxx xxxx xxxx xxxx xxxx xxxx xxxx
xxxx xxxx xxxx xxxx xxxx xxxx xxxx
xxxx xxxx xxxx xxxx xxxx xxxx xxxx
x                                   (macrocell fuse)
```

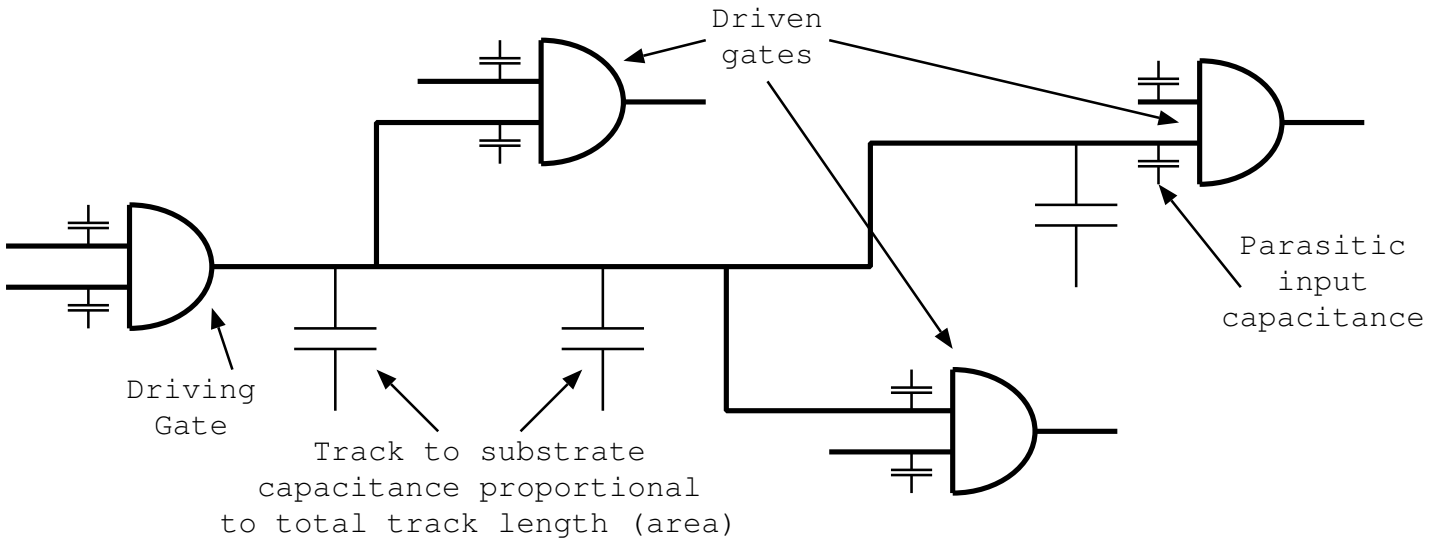
Delay-power style of technology comparison chart



	Technology	device	propagation	power	product
1977	CMOS	HEF4011	30 ns	32 mW	960 pJ
1982	ECL	sp92701	0.8 ns	200 mW	160 pJ
1983	CMOS	74hc00	7 ns	1 mW	7 pJ
1983	TTL	74f00	3.4 ns	5 mW	17 pJ
1996	CMOS	74LVT00	2.7 ns	0.4 mW	1.1 pJ

2-Input NAND gate. 74LVT00 is 3V3. On-chip logic is much faster.

Logic net with tracking and input load capacitances



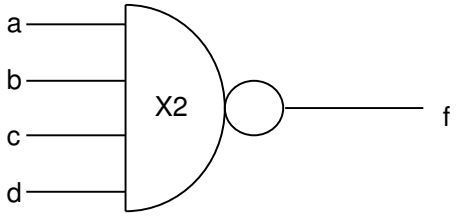
An example cell from a manufacturer's cell library

NAND4 Standard Cell

Library: CBG0.5um

4 input NAND gate with x2 drive

Schematic Symbol



Simulator/HDL Call

NAND4X2(f, a, b, c, d);

Logical Function

F = NOT(a & b & c & d)

ELECTRICAL SPECIFICATION

Switching characteristics : Nominal delays (25 deg C, 5 Volt, signal rise and fall 0.5 ns)

Inputs	Outputs	O/P Falling		O/P Rising	
		(ps)	ps/LU	ps	ps/LU
A	F	142	37	198	33
B	F	161	37	249	33
C	F	165	37	293	33
D	F	170	37	326	34

Min and Max delays depend upon temperature range, supply voltage, input edge speed and process spreads. The timing information is for guidance only. Accurate delays are used by the UDC.

CELL PARAMETERS : (One load unit = 49 fF)

Parameters	Pin	Value	Units
Input loading	a b c d	2.1 2.1 2.1 2.0	Load units
Drive capability	f	35	Load units

Current digital logic technologies

1994 - First 64 Mbit DRAM chip.

- 0.35 micron CMOS
- 1.5 micron^2 cell size (64E6 $\times$ 1.5 um^2 = 96E6)
- 170 mm^2 die size

1999 - Intel Pentium Three

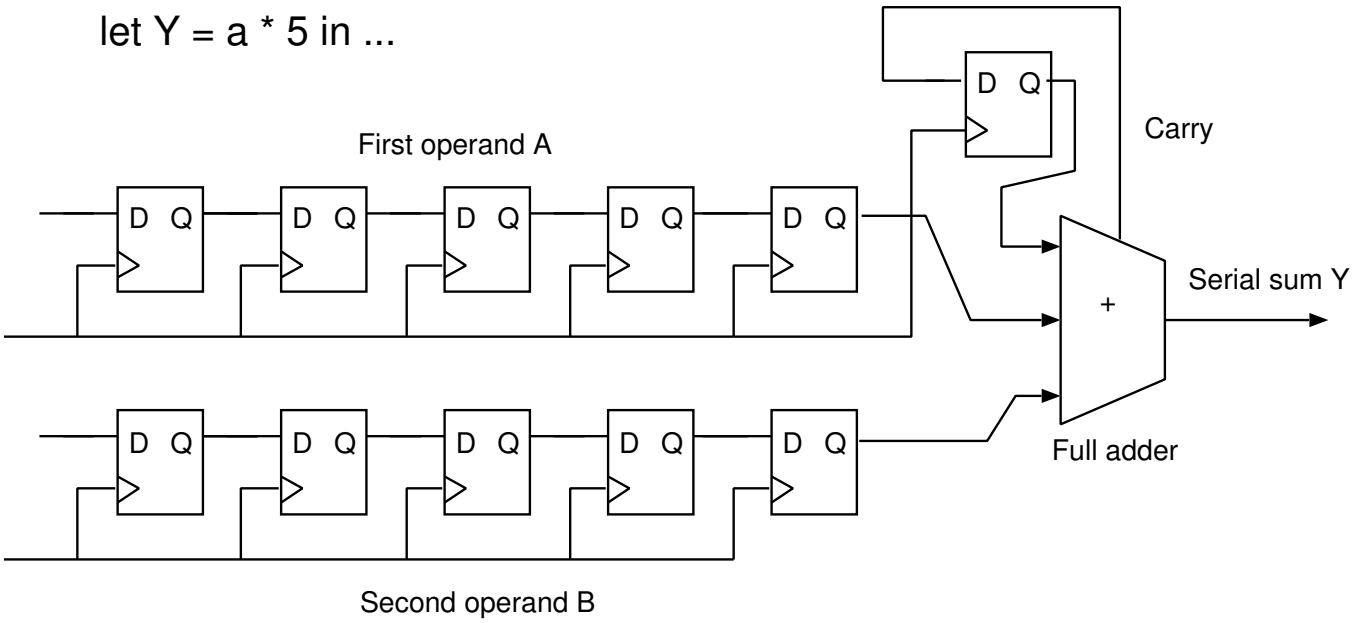
- 0.18 micron line size
- 28 million transistors
- 500-700 MHz clock speed
- 11x12 mm (140 mm^2) die size

2003 - Lattice FPGA

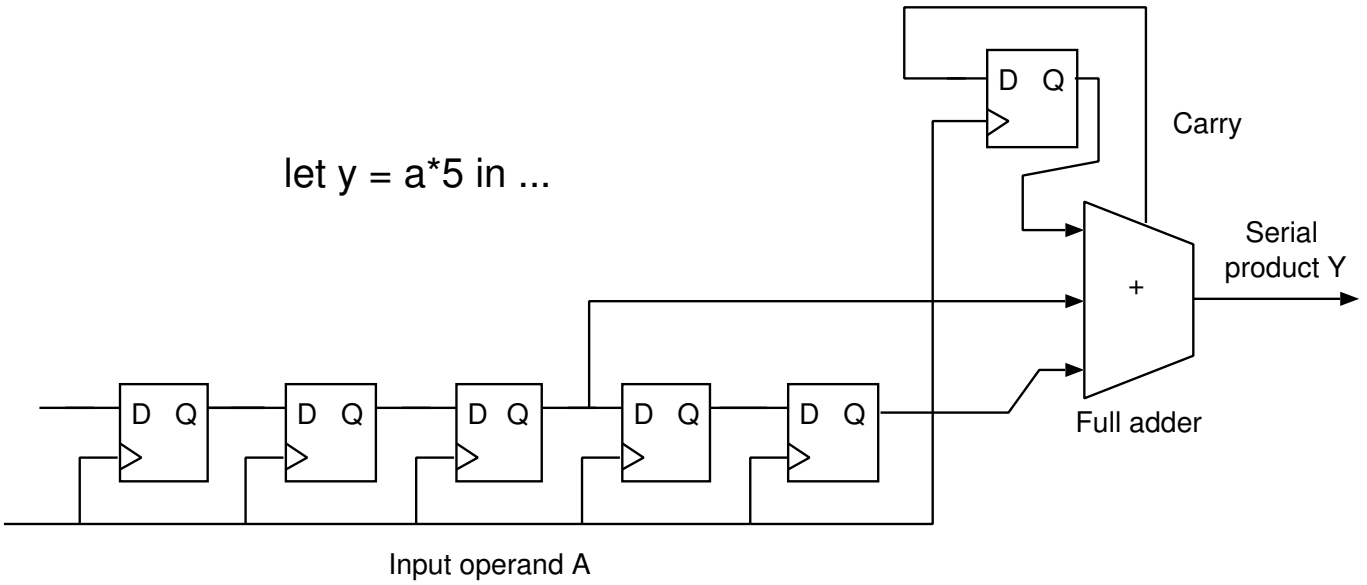
- 1.25 million use gate equivs
- 414 Kbits of SRAM
- 200 MHz Clock Speed
- same die size.

See www.icknowledge.com

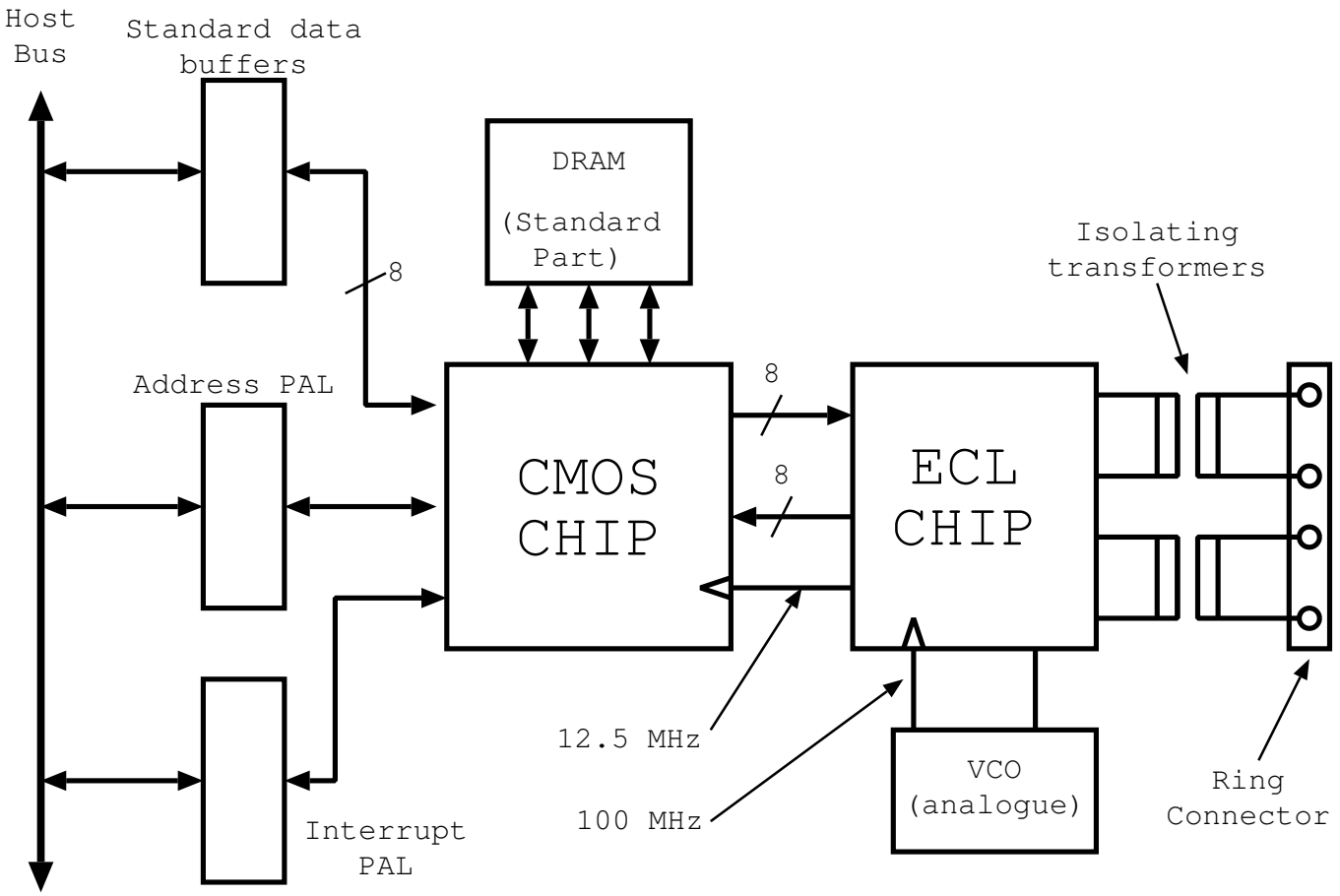
Addition of two integers serially, l.s.b first



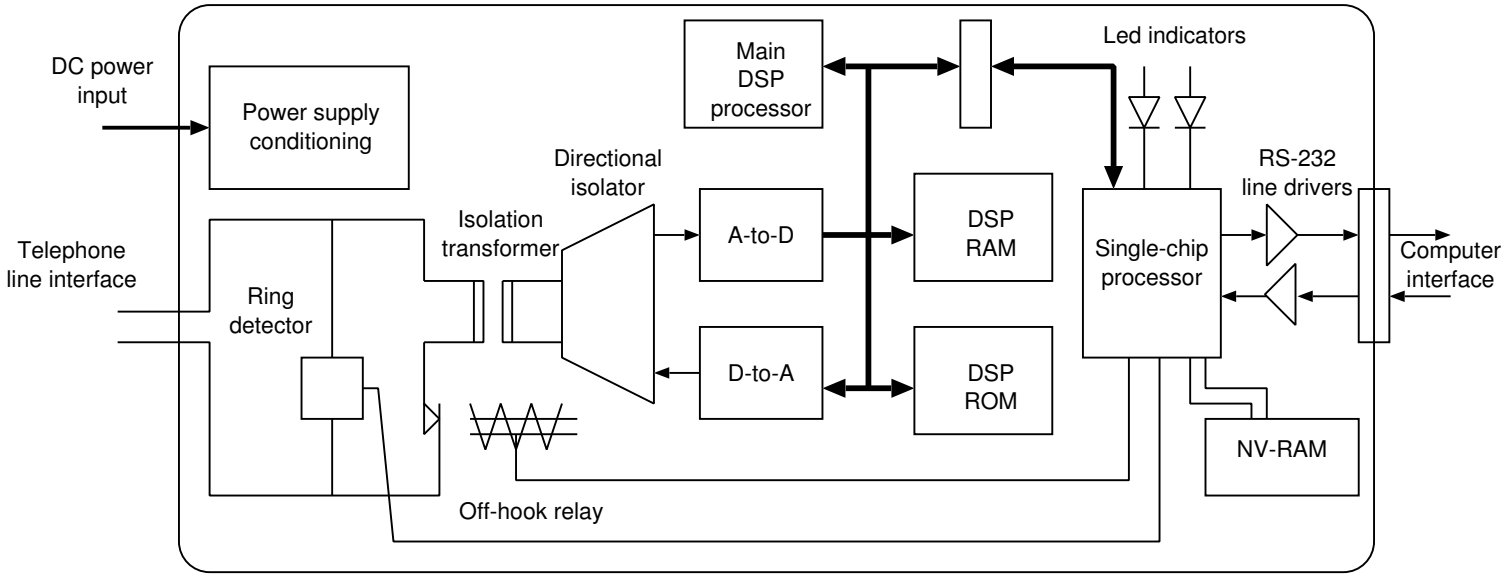
Bit-serial multiplication of an integer by a hardwired constant



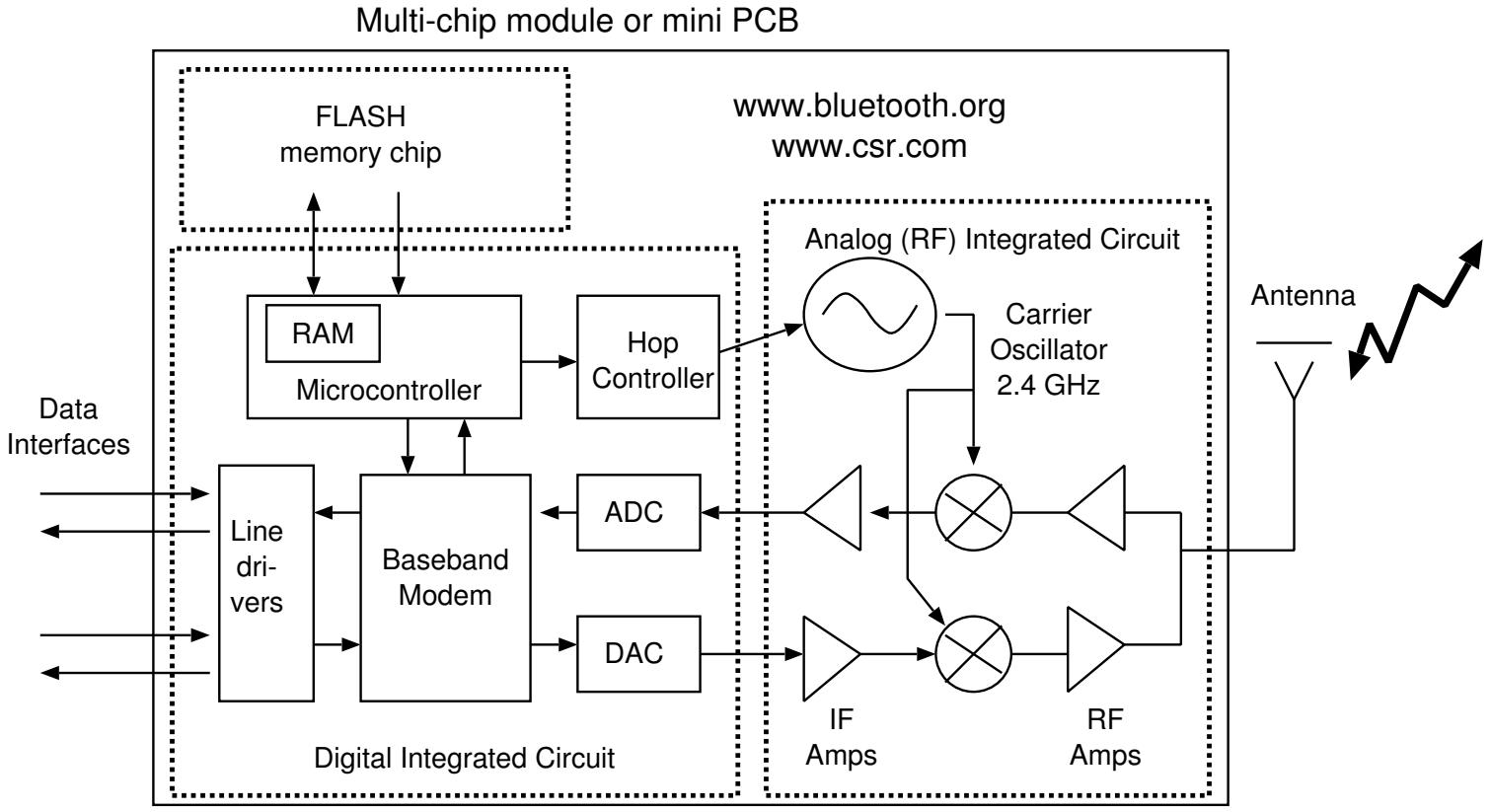
Design partitioning: The Cambridge Fast Ring



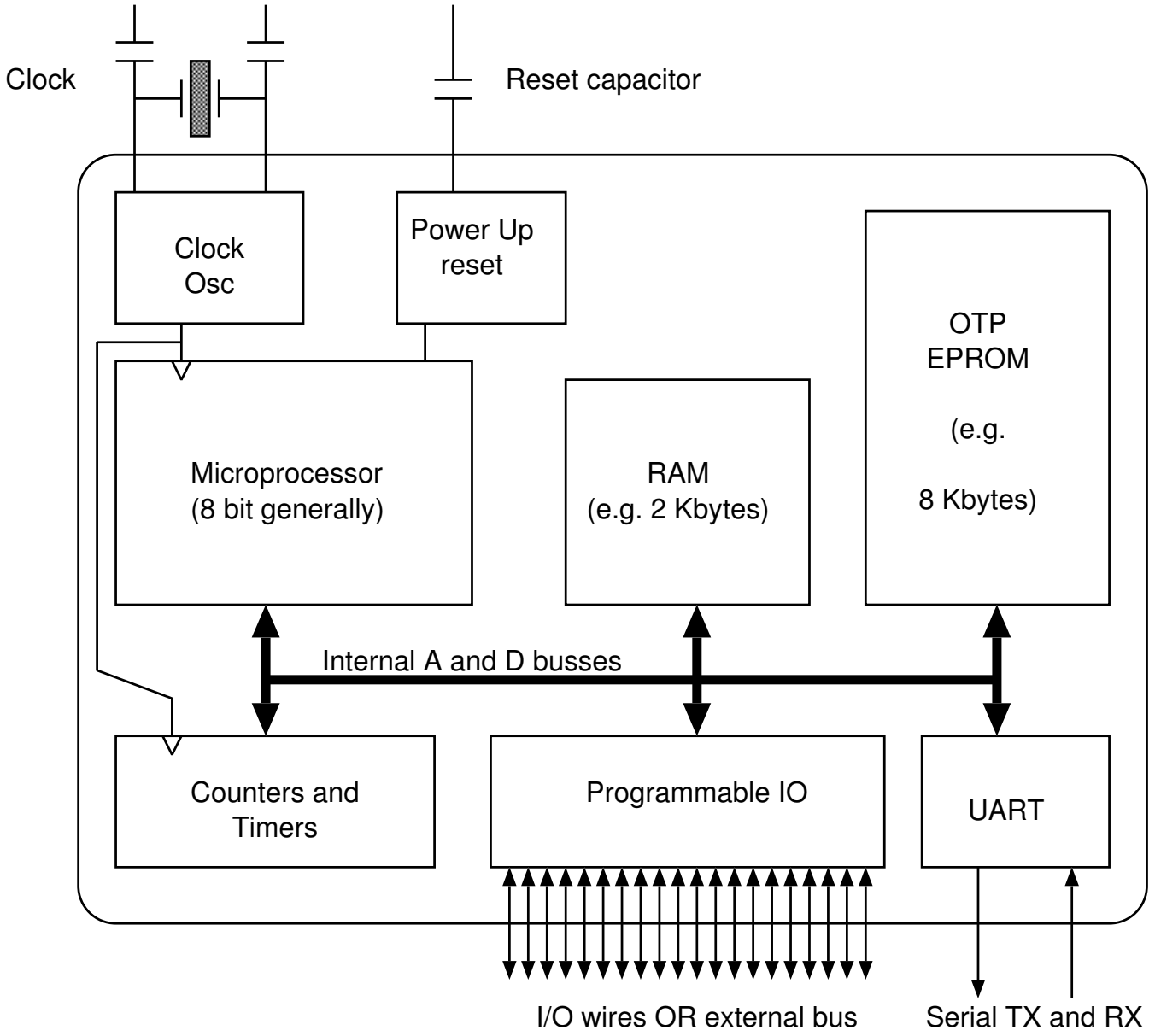
Design partitioning: An external modem



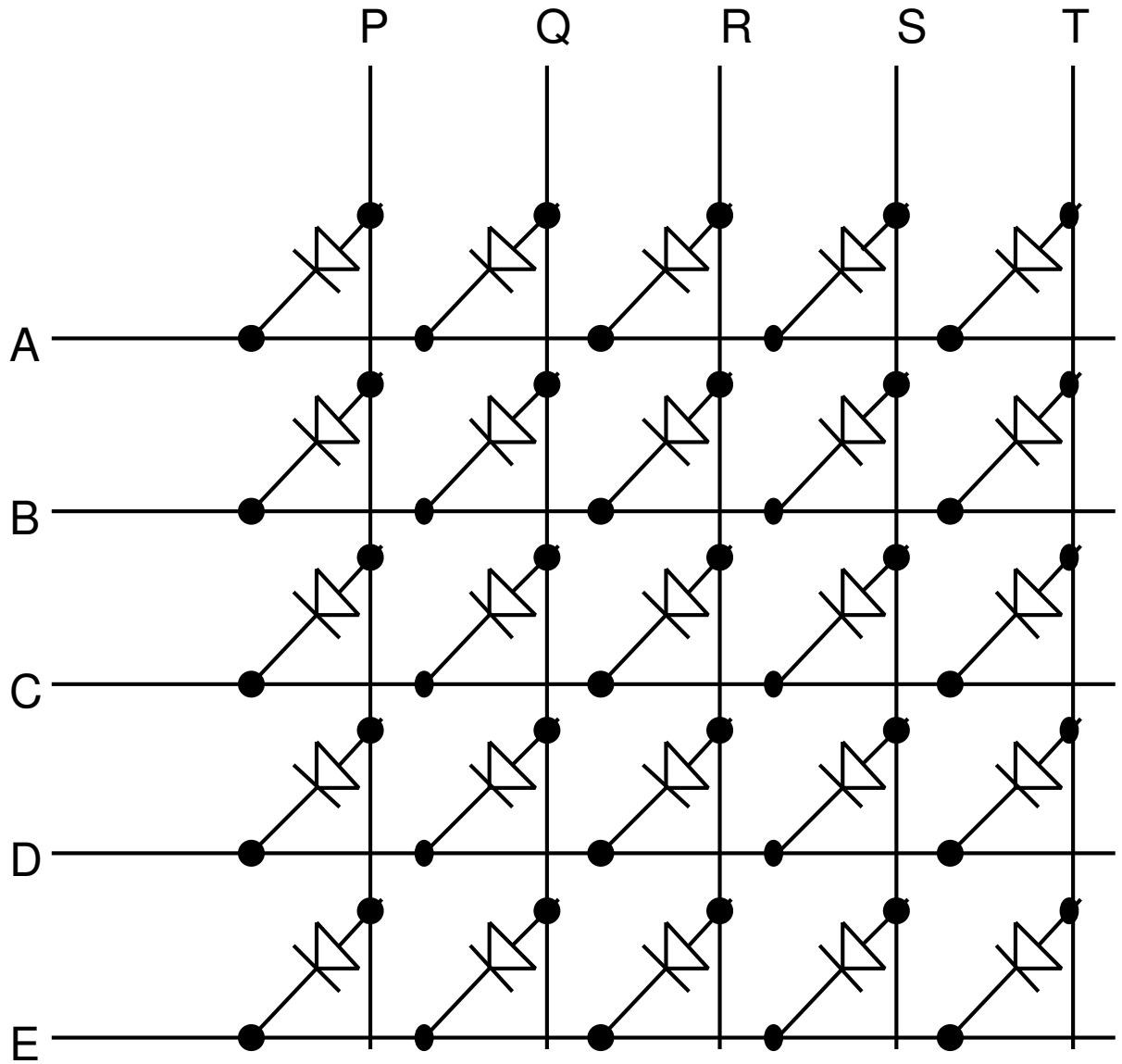
Design partitioning: A Miniature Radio Module



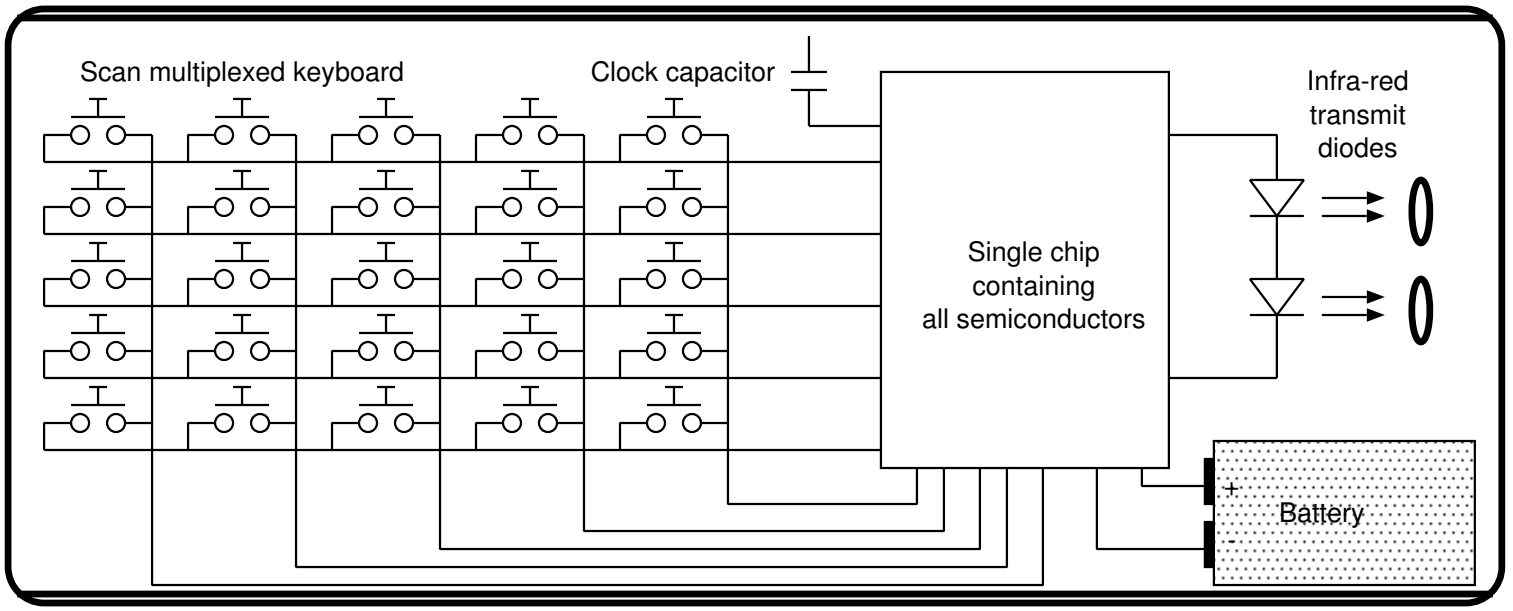
A Very Basic Microcontroller



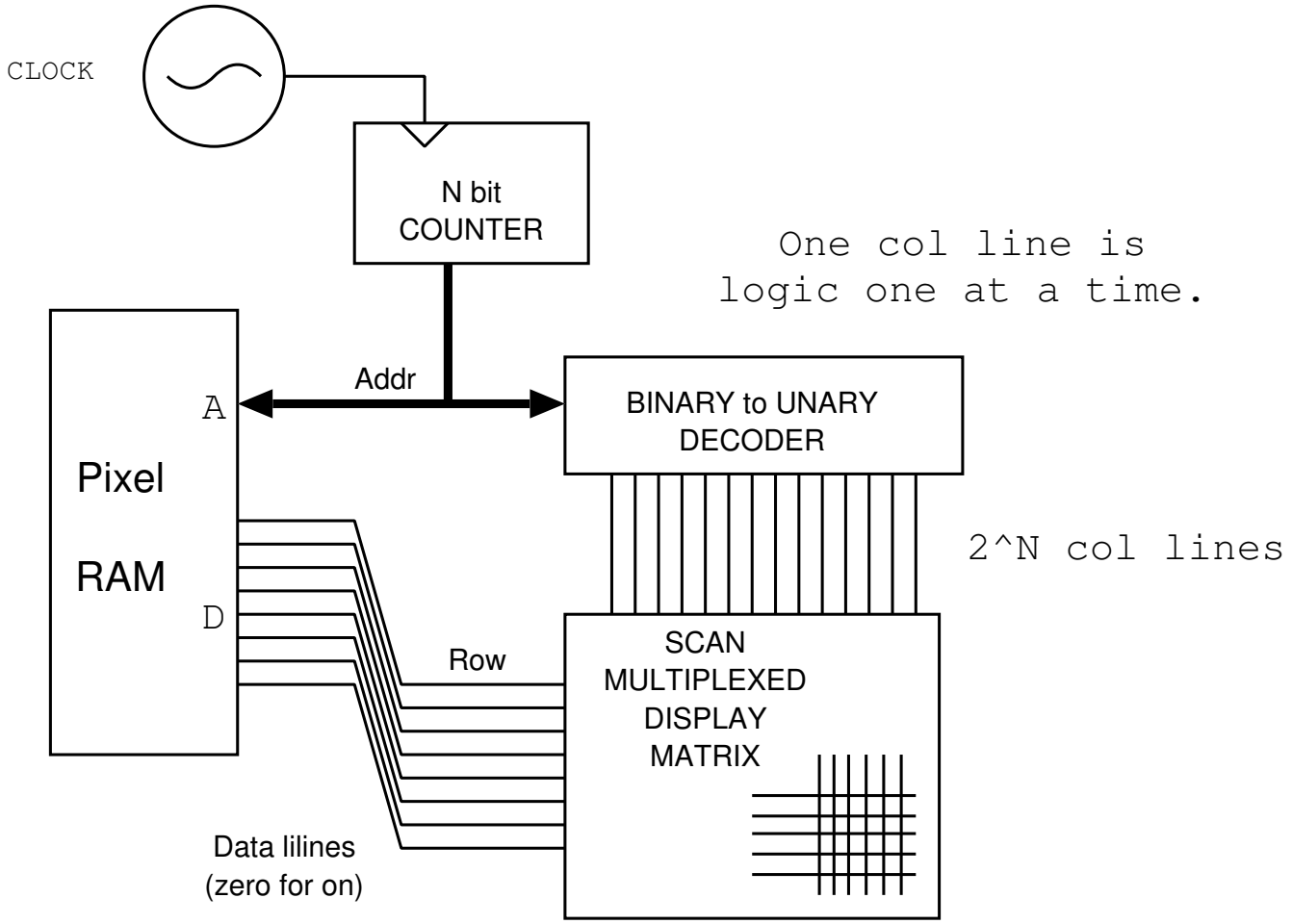
LEDs wired in a matrix to reduce external pin count



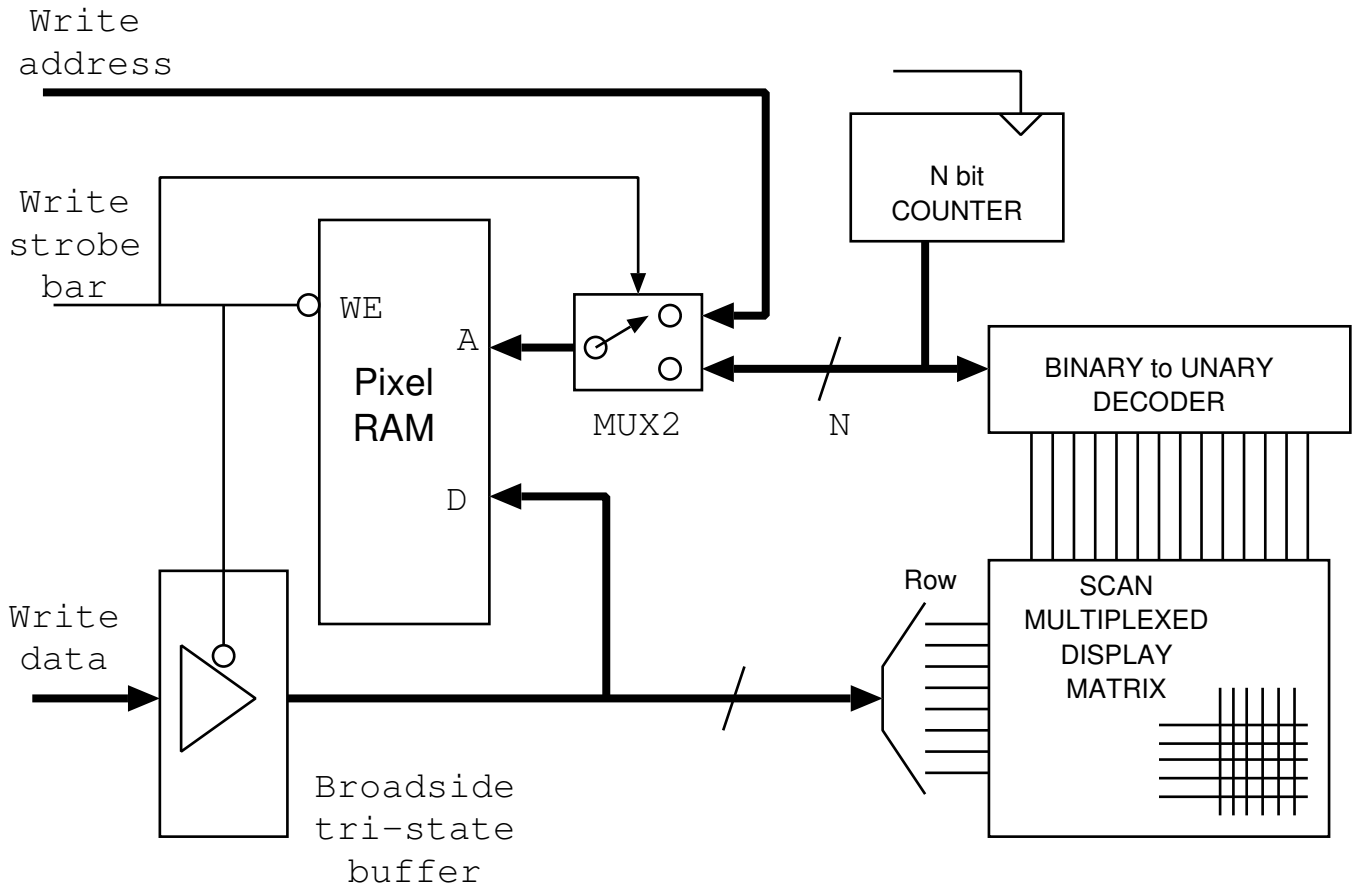
IR Handset Internal Circuit



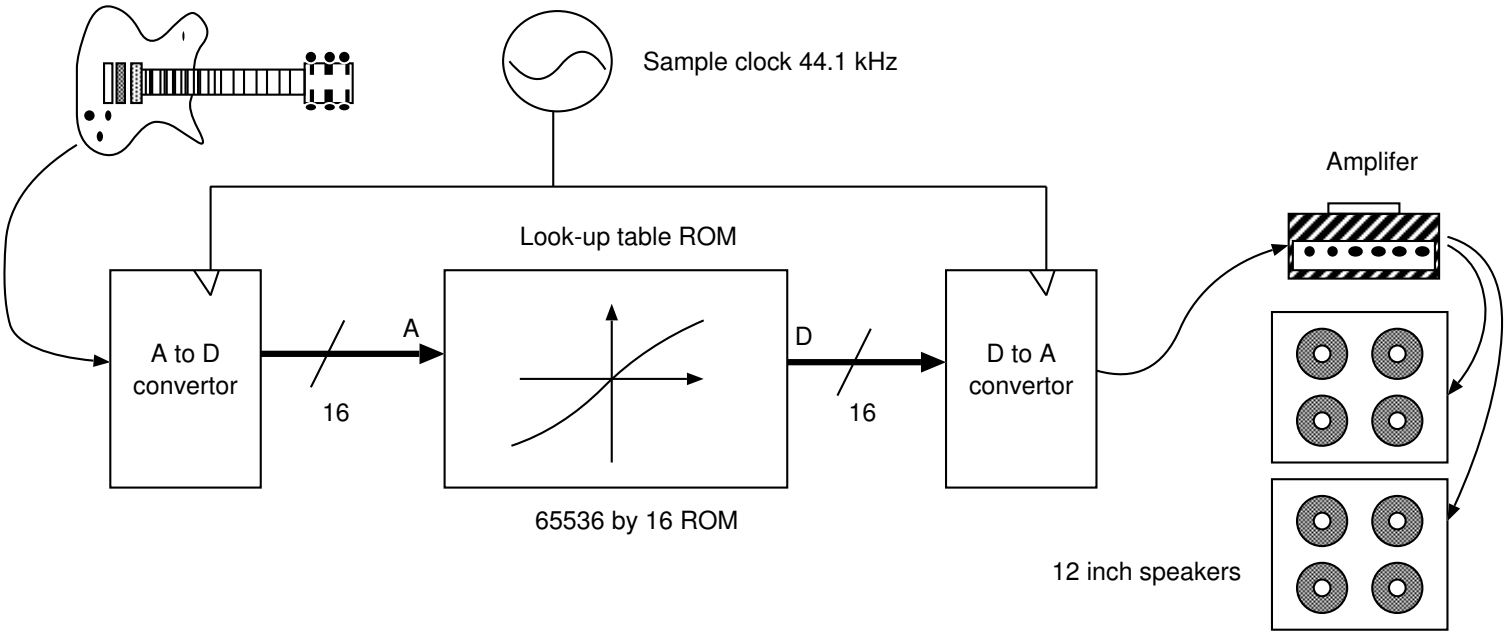
Scan multiplex logic for an LED pixel-mapped display



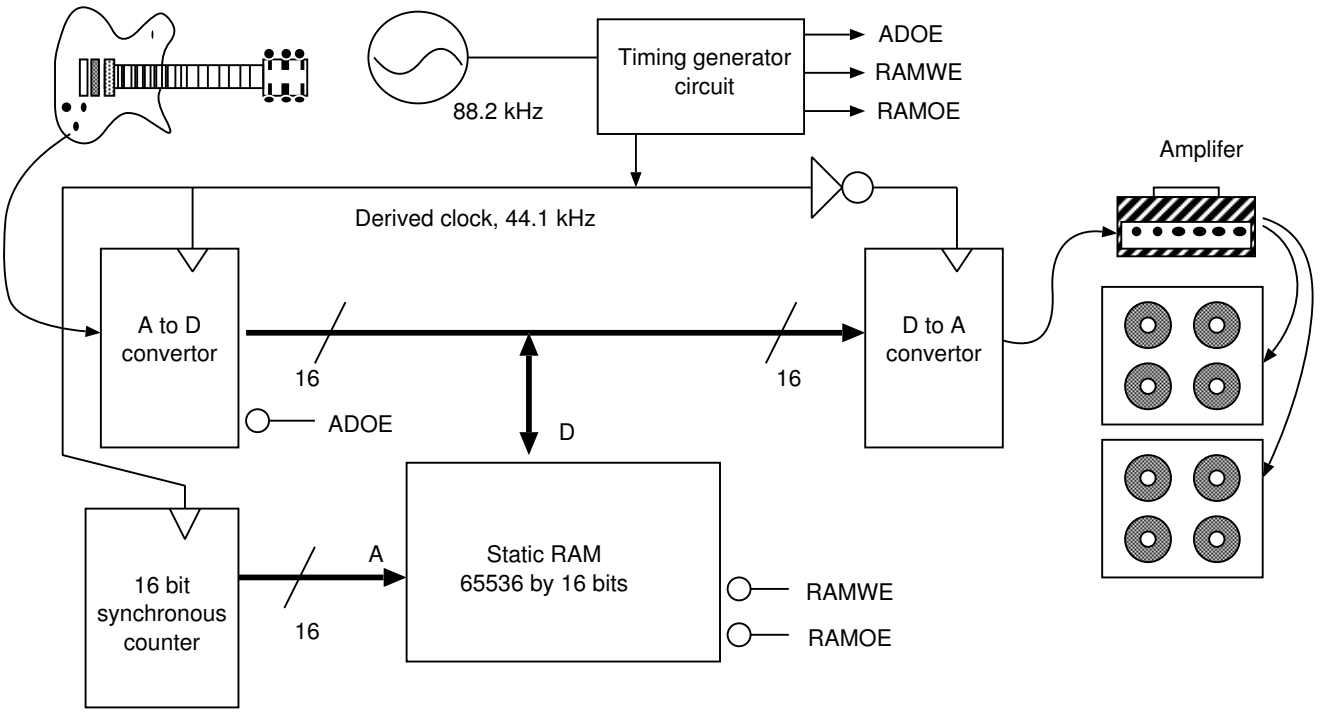
Addition of psudo dual-porting logic

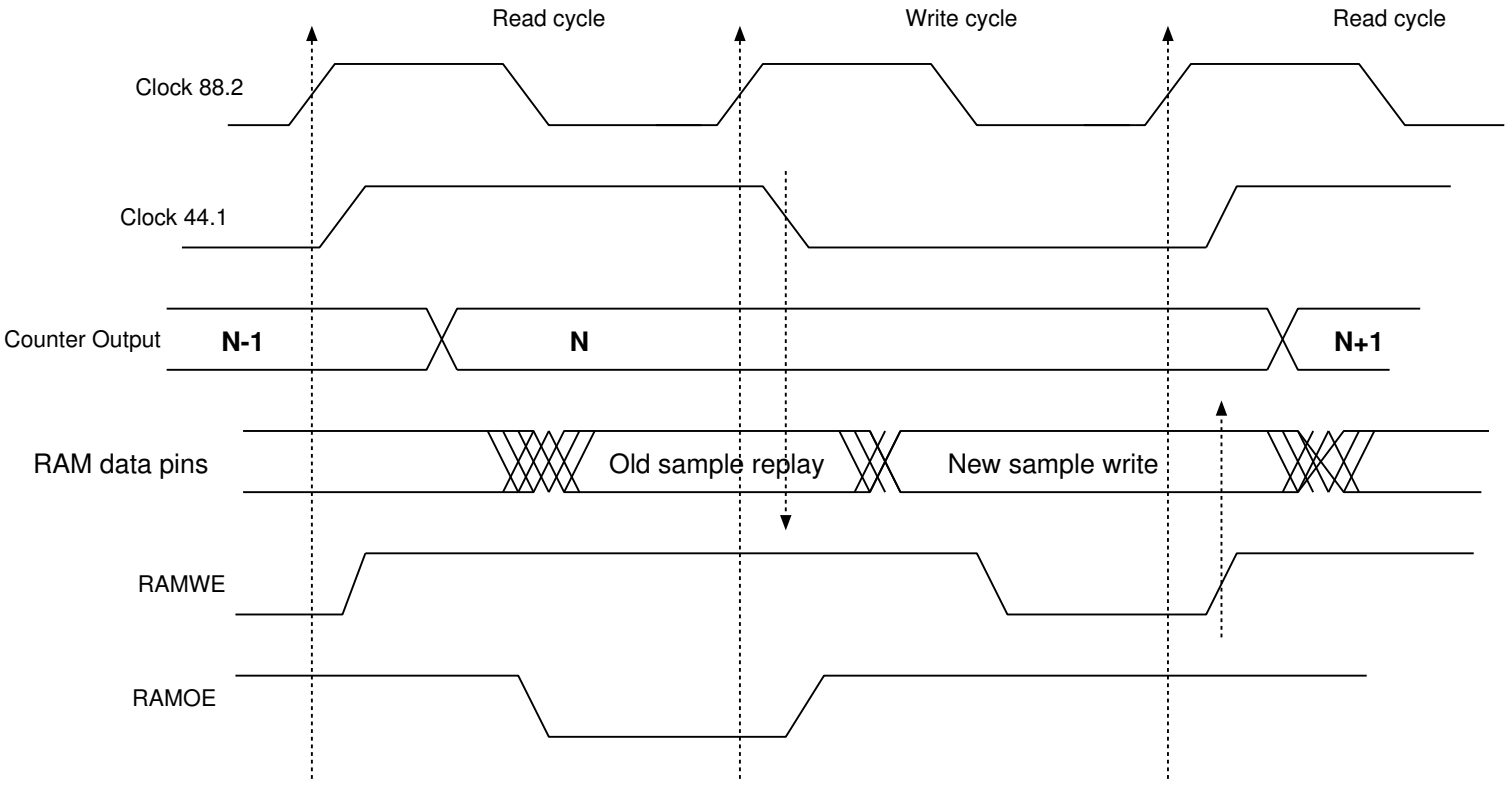


Use of a ROM as a function look-up table

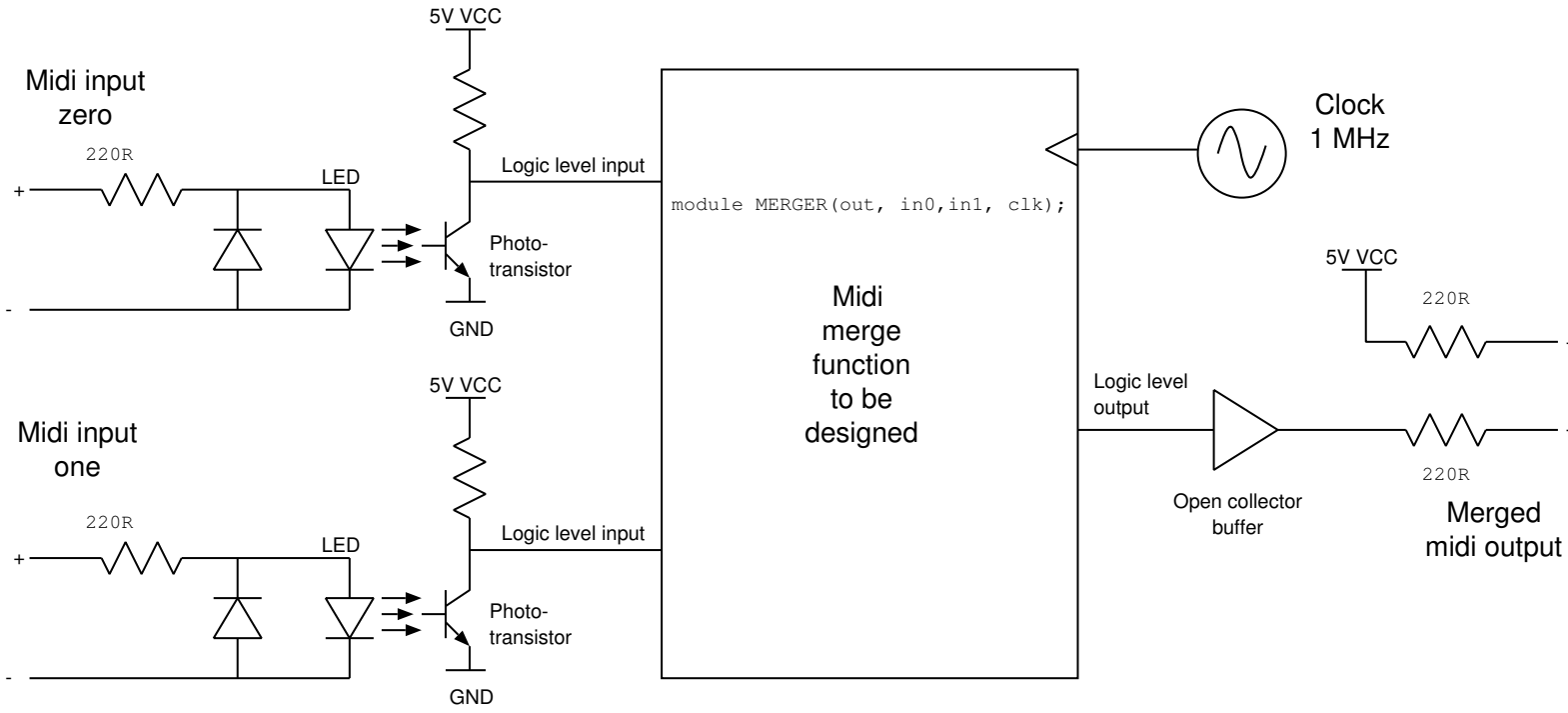


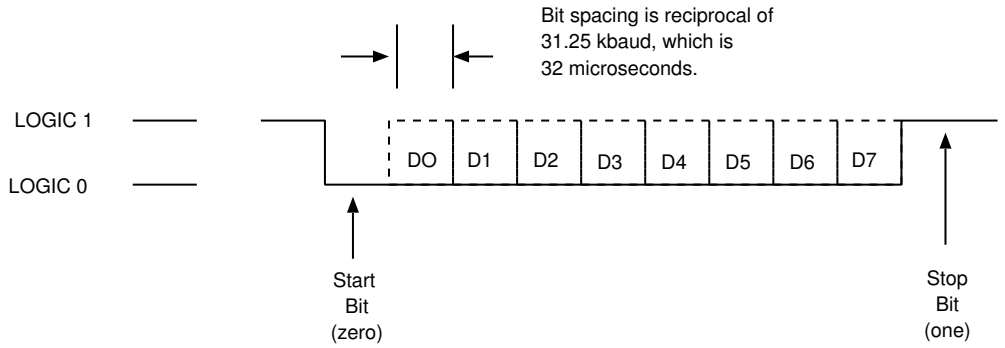
Use of an SRAM to make the delay required for an echo unit





Merge unit block diagram

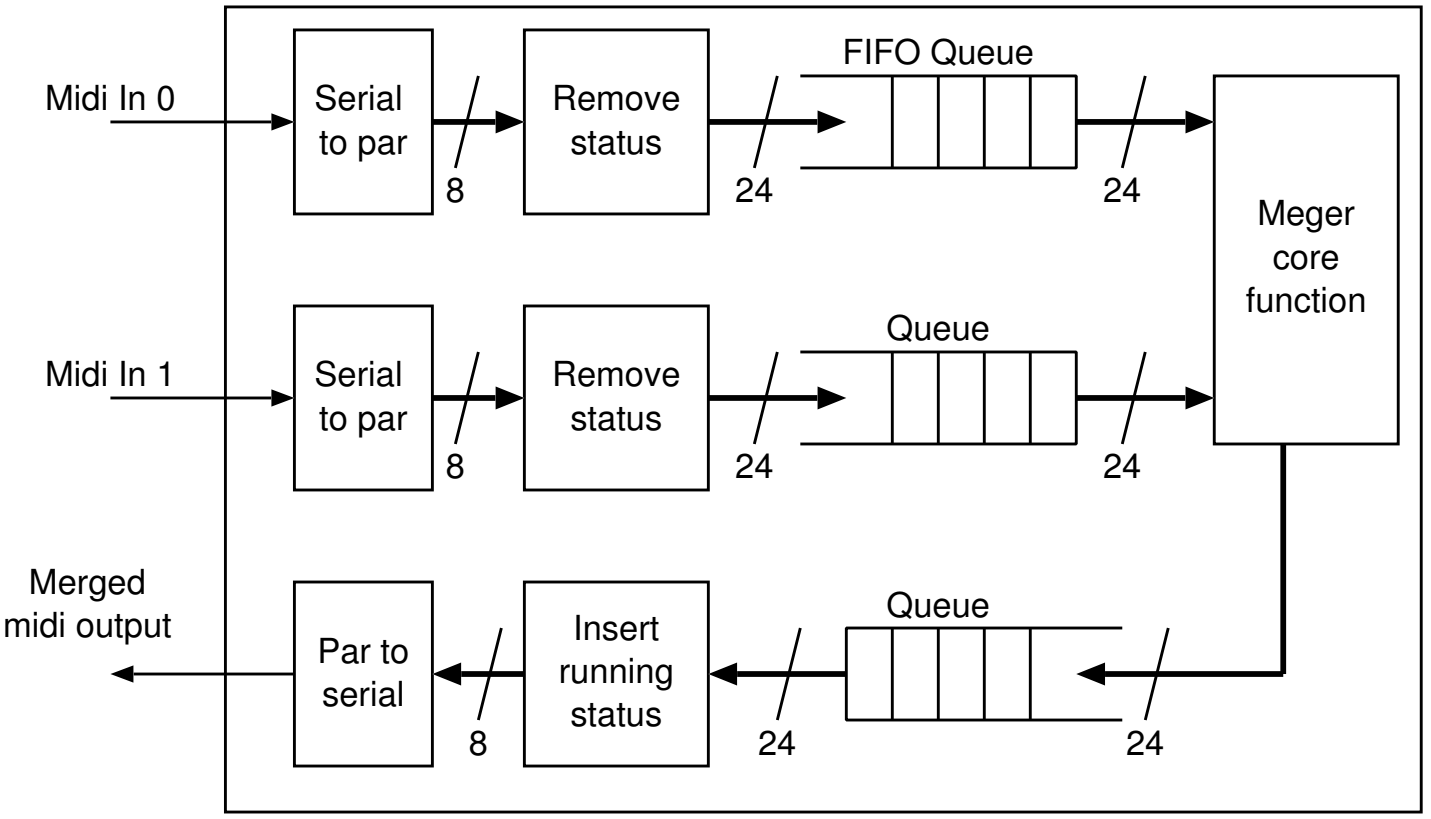




MIDI serial data format

```
9n kk vv        (note on)
8n kk vv        (note off)
9n kk 00        (note off with zero velocity)
```

MIDI merge unit internal functional units



The serial to parallel converter:

```
input clk;
output [7:0] pardata;   output guard;
```

The running status remover:

```
input clk;
input guard_in;   input [7:0] pardata_in;
output guard_out; output [23:0] pardata_out
```

For the FIFOs:

```
input clk;
input guard_in; input [7:0] pardata_in;
input read; output guard_out;  output [23:0] pardata_out;
input read; output guard_out;  output [23:0] pardata_out;
```

For the merge core unit:

```
input clk;
input guard_in0; input [23:0] pardata_in0; output read0;
input guard_in1; input [23:0] pardata_in1; output read1;
output guard_out; output [23:0] pardata_out;
input read; output guard_out;  output [23:0] pardata_out;
```

Status inserter / parallel to serial converter are reverse of reciprocal units