Structured Hardware Design
6L 1A Easter 2000.

Dr Ian Pratt

April 25, 2000

Preface

These five topics are given in six lectures to the part 1A 50 percent candidates in the Easter Term. This course concentrates on system-level design issues, but nonetheless, a good knowledge of the digital electronics course is required as background material.

Together with the reading below, the best preparation for this course is to look around at electronic equipment, from cell-phones, to toys, disco lights, to lifts to teletext televisions: ask yourself what components these items are made from, how many are sold and how were they designed ?

Bibliography (Book list)

Books related to the course, in approximate order of importance, are:

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.'* Benjamin Cummings ISBN 0 8053 2703 7

Texas Instruments. *'System 74 Series Logic Family.'* Texas Instruments.

'IEEE Transactions on Consumer Electronics' in the SPL.

Glossary of jargon and acronyms.

Here are some acronyms:

ALU	Arithemetic and logic unit.
ASIC	Application specific integrated circuit.
BICMOS	A process with both CMOS and bipolar transistors on one die.
CAD	Computer aided design.
CAE	Computer aided engineering.
CAM	Computer aided manufacture.
CLB	Configurable logic block.
CRC	Cyclic redundancy check (added to end of a data frame).
CMOS	Complementary metal oxide of silicon.
DRAM	Dynamic random access memory.
DMA	Direct memory access.
DSP	Digital signal processor/ing.
EDO	Extended Data Out for DRAMS.
ECL	Emitter coupled logic.
EDA	Electronic Design Automation (CAD tools).
EMC	Electromagnetic compatibility.
EMI	Electromagnetic ingress or i(m)missions (sic).
FET	Field effect transistor.
FSM	Finite state machine.
FPGA	Field-programmable gate array.
GaAs	Gallium Arsenide.
HDL	Hardware description language.
IDE	The bus used for connecting hard drives and CD roms
IEEE	Institure of electrical and electronic engineers.
IPR	Intellectual Property Rights.
IRDA	Infra-red data association
JTAG	Joint technical advisory group
LSI	Large scale integration (say about 5000 gates, 10000 transistors).
LCA	Logic cell array.
LCD	Liquid crystal display.
Mbps	Mega-bits per second
MBPS	Mega-bytes per second
MCM	Multi-chip module.
MSI	Medium scale integration (i.e. the larger ttl devices).
NRE	Non recurring engineering costs. i.e paid once per design.
OTP	One-time programmable
PAL	Programmable array logic.
PIO	Programmed Input and Output.
PCB	Printed ciruit board.
PLL	Phase-locked loop.
PIN	Personal identification number.
RTL	Register transfer level.
SRAM	Static RAM.
SIMM	Single in-line memory module.
SSI	Small scale integration (i.e. about 4 gates on a chip).
TTL	Transistor-transistor logic.
UART	Universal asynchronous recevier and transmitter.
USB	Universal serial bus.
Verilog	An HDL language in wide use (not an acronym).
VHDL	VHSIC Hardware Description Language.
VHSIC	Very High Speed Integrated Circuit.
VNL	Verilog netlist.
VCO	Voltage controlled oscillator.
VGA	Video graphics adaptor
VLSI	Very large scale integration (big chips).
XNF	Xilinx netlist format.

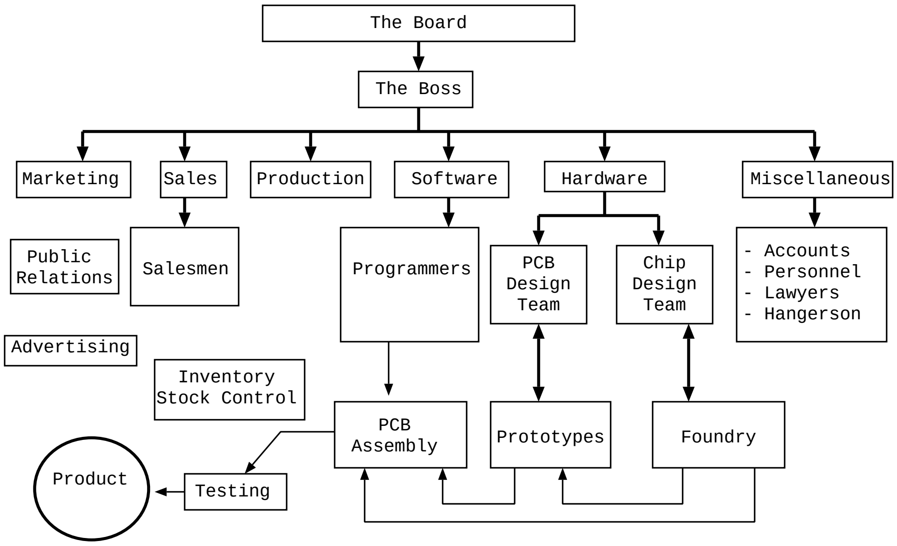


Figure 1: Structure of a small company with hardware products that use its own hardware, software and chips.

Introduction

A good hardware design works the first time. A structured approach to its design is required if the system is complicated. Simulation of the system behaviour before fabrication is also vital to help get it right. Multiple prototypes are normally tested in real situations before production runs commence. As Figure 1 shows, the final product relies on successful interworking between custom chips and the other parts on the circuit boards, together with the software.

Most designs today are held in a database which contains details of custom circuits, printed circuit boards, other chips, software, part numbers and so on. The database can help automate every part of the design process, except for the initial design decisions. These decisions define the basic structure of the product and they are what this course is about.

To create a new hardware design, as with software, it is important to use a top-down approach, starting with an accurate capture of the requirements specification and then decomposing the system into modules. As with software, the module must have well-defined functions and well-defined interfaces and the modules must be suitable for individual testing. However, two big differences are 1. that the modules in hardware are often made of different technology from one another, and 2. that the desireability of using pre-existing modules is much greater.

Integration of software and hardware is a key research area in EDA (Electronic Design Automation).

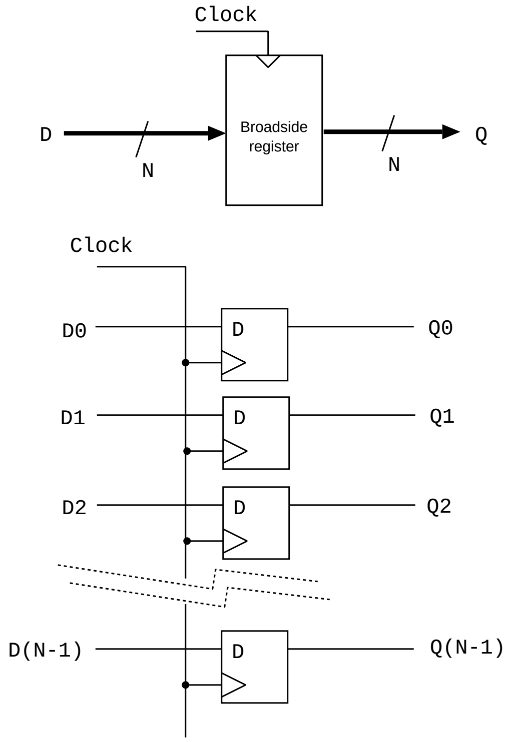


Figure 2: A broadside register: schematic symbol and internal structure.

1 Building Blocks in Hardware Systems.

Learners' Guide: What you should learn from this section is:

- Designs are made of modules.
- Modules are made of modules.
- Modules have easy to describe functions which could be tested in isolation from the rest of the system.
- Modules could be gates, collections of gates, chips, boards etc..
- Modules are designed to be re-usable.
- Examples of typical modules.

This section introduces a number of building blocks that are frequently used in hardware designs. A block may be an AND gate or it may be a PC. In times gone by, some blocks were refered to MSI (medium scale integration) and LSI (large scale integration) subsystems. Today, many of these blocks rarely exist as separate components: instead they are placed inside custom (or semi-custom) application-specific, integrated circuits (ASICs). Putting a PC inside a custom chip is not far off from being an everyday design step.

1.1 Busses and Broadside Components.

A bus is a number of signals that operate in parallel to carry a binary number. In the schematics, a thicker line is used, and the number next to the slash is the number of signals in the bus.

Figure 2 demonstrates the concept of a broadside component, in this case, a broadside set of D-types. **To form a broadside component, the building block is instantiated a number of**

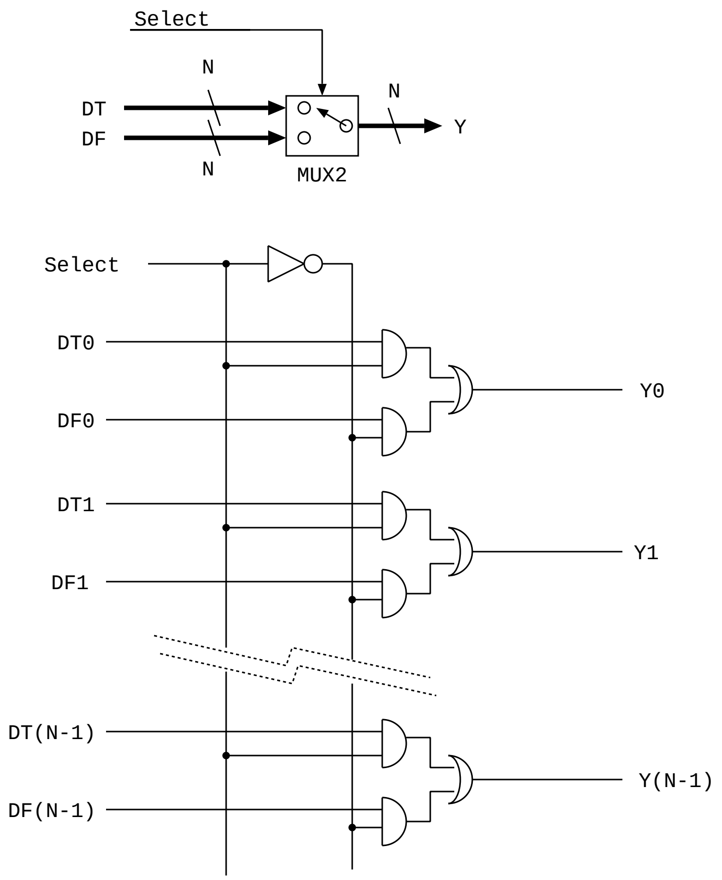


Figure 3: An N-bit broadside, two-to-one multiplexor.

times with the data inputs and outputs connected to the member lines of input and output busses. The control connections are paralleled and driven from a single external source. A broadside set of flip-flops is a *register*.

Figure 3 shows the schematic and a suitable circuit implementation for a broadside, two-input multiplexor. Most systems will regard a two input multiplexor as a fundamental building block and implement it at the transistor level, rather than from gates.

Figure 4 shows the schematic symbol for a dual-port register file. This device internally contains a number of broadside registers and multiplexors, but they are grouped into the file for convenient access. Such a file is a key part of all computers. The illustrated dual-port file has two reading ports and one writing port, so might be considered a tripple-ported file. On the positive edge of the clock, the data on the write port is stored in the addressed register. For reading, at any time, like an SRAM, the read address may be changed and the output data will change shortly afterwards.

Exercise: Sketch the circuitry that would turn a collection of standard broadside registers into a dual-port file. You will have to use one of the clock enabling techniques defined in section 2.2.

1.2 Read Only Memories

Figure 5 shows the logic symbol for a read only memory (ROM). The contents are non-volatile and are placed in the ROM during manufacture using a fabrication mask or with a special programming step in the field. **There are various technologies for storing data in a non-volatile way on ICs: they acheive roughly the same result.** The techniques for storing field programmed information are:

- W-Sn fuse: One time programmable by melting selected fuses. This is an old technology

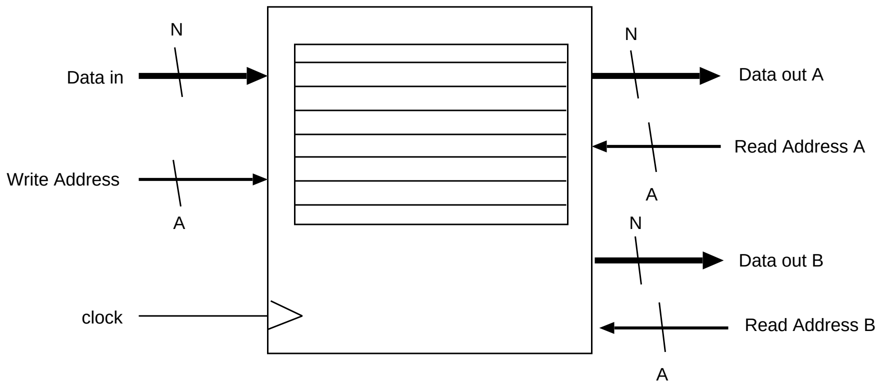


Figure 4: Dual port register file.

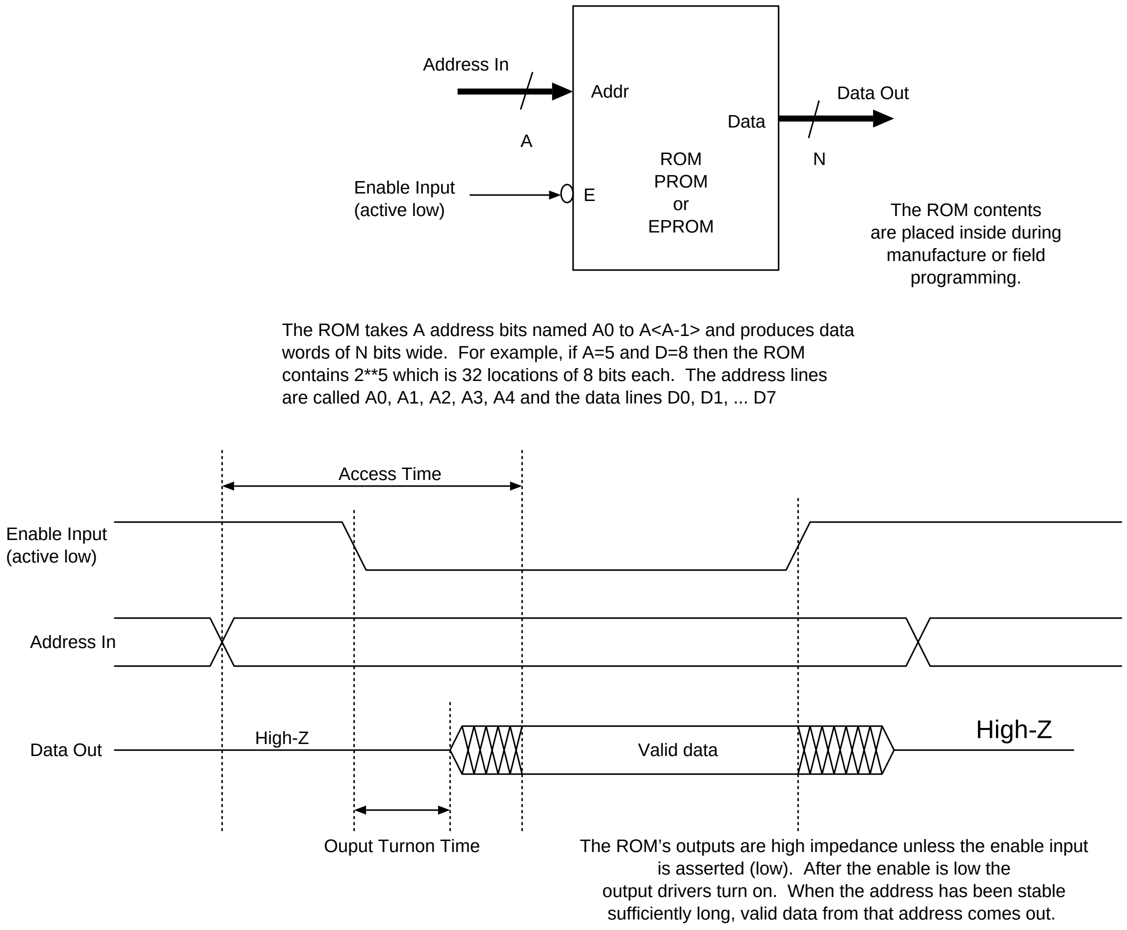


Figure 5: Read Only Memory (ROM).

which is not very reliable or dense. When used for ROM gives a PROM.

- Antifuse: One time programmable, high density process. Used on one contemporary family of FPGAs. Not used in ROMs today.
- Floating gate technology with ultraviolet erase. A static electrical charge is stored on the floating gate and is put there using a supervoltage that causes tunneling or other breakdowns in the insulator. Discharge is achieved using intense ultraviolet illumination to make the charge leak away. When used for a ROM, gives an EPROM.

  A varient on EPROM is an OTP-EPROM (one-time-programmable) which the same device packaged in a windowless package at much lower cost.
- Electrically reprogrammable floating gate devices. Again charge is stored on the gate, but electrical techniques used for programming can be applied for erasing. Used widely in personal organisers and (Flash) memory cards and in most modern PAL devices. Often, erase of individual bits is impossible: only block erase is supported.

Mask programmed ROMs of several megabytes may cost sub 50 pence each in quantities of tens of thousands. Flash memory may be more than 100 times more expensive. The choice of which technology to use depends on the number of units that are going to be made and the expected number of reprogramming cycles required. There is always a testing problem with one time programmable devices!

1.3 RAM memories

Figure 6 shows a random access memory or RAM. It is as equally random access as a ROM, so is a misnomer. RAM is normally volatile, meaning it loses its contents when power is disconnected. Static RAM (SRAM) normally uses one flip-flop per bit stored. A transparent latch suffices, so it is not as complicated as having a D-type per bit. A possible internal structure for an SRAM is shown in Figure 7. However this would lead to a long, skinny chip. In practice, a square array of storage elements is used, meaning that the on-chip tri-state bus is much wider than the external data bus. The on-chip bus will have width approximately equal to the square root of the number of bits stored in the device. Modern SRAM devices have an access time of 20 nanoseconds and have capacities of about half a megabyte. Price is about six pounds per megabyte, but depends greatly on the access time required. Second-level cache chips for PCs are SRAMs and produced in high volume, so are currently a good way to purchase SRAM, even for other designs or uses.

A property of the SRAM, shared with the ROM, is that there is no clock and, during read, the output is essentially a combinatorial logic function of the address input value.

When writing, the address must not change and the data present on the data pins at the back edge of the (active low) write pulse is the data actually written.

Figure 8 shows the schematic symbol for a dynamic RAM. DRAMs store their data not in flip-flops, but in small capacitors. The value stored leaks away over time unless refreshed. It is also destroyed by readout: after readout, the device always writes back the value internally, removing the job from external circuitry, but meaning that the external logic must not start a new cycle immediately. Therefore the cycle time and access times of DRAM are different. Typically, today, DRAM cycle and access times are 60 and 120 nanoseconds and capacity is 2 to 4 megabyte.

All DRAMs are accessed by applying the address in two halves. First a row address is presented and RAS taken low, then a column address is applied, and CAS is taken low. These row and column addresses typically correspond to the actual physical topology of the rectangular array of bits on the device. In external view, some DRAMs are one bit wide and others are four.

DRAM is much cheaper than SRAM owing to the smaller silicon area needed per bit. Smaller packages are also possible, owing to the multiplexed address bus.

Refresh cycles do not transfer data in or out of the chip, but must be done sufficiently often anyway. A typical specification is that 512 refresh cycles are executed every 4 milliseconds. Since DRAM consumes most of its power when being refreshed, a designer who puts all of the refresh cycles in

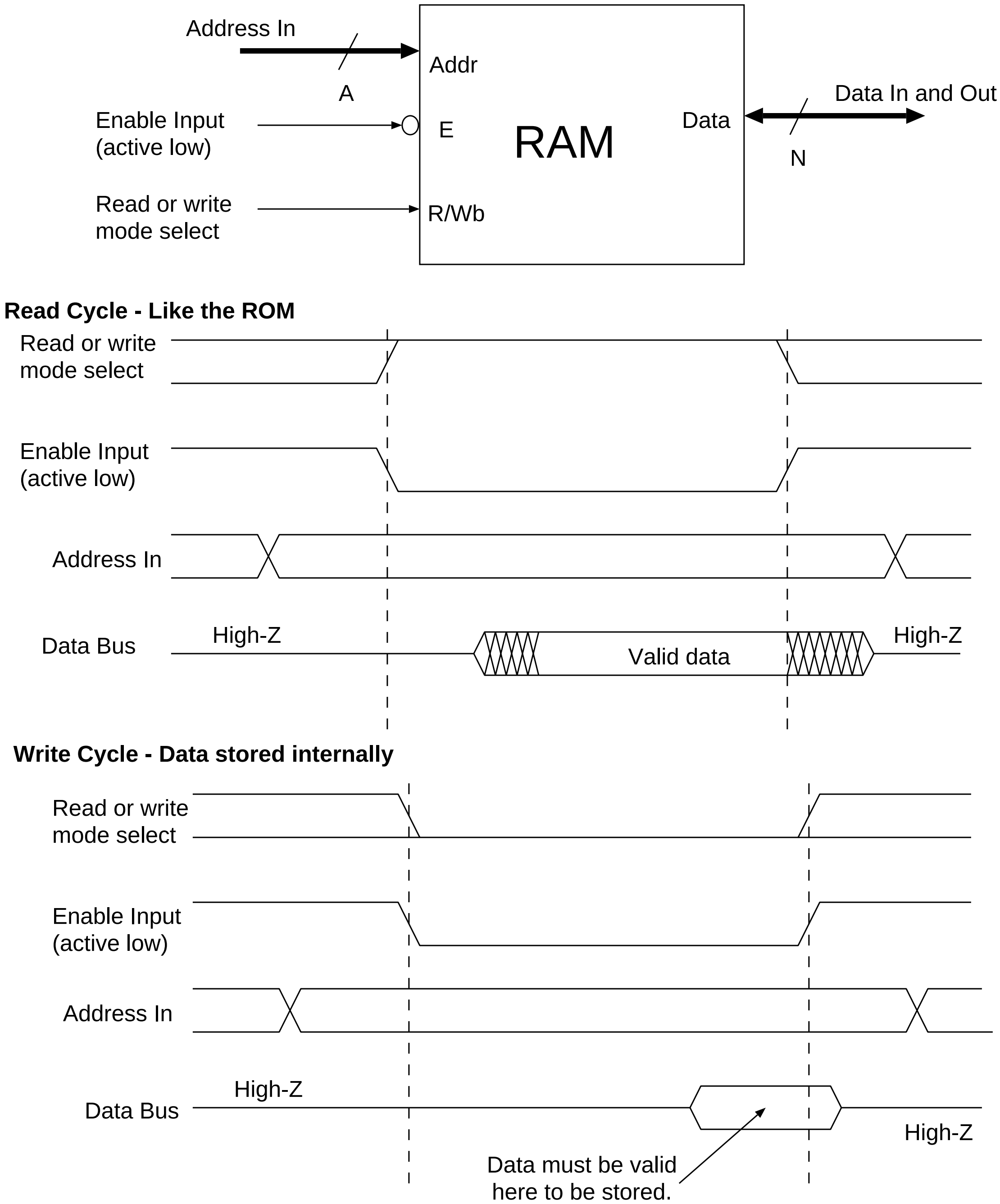


Figure 6: Read and Write Memory (RAM).

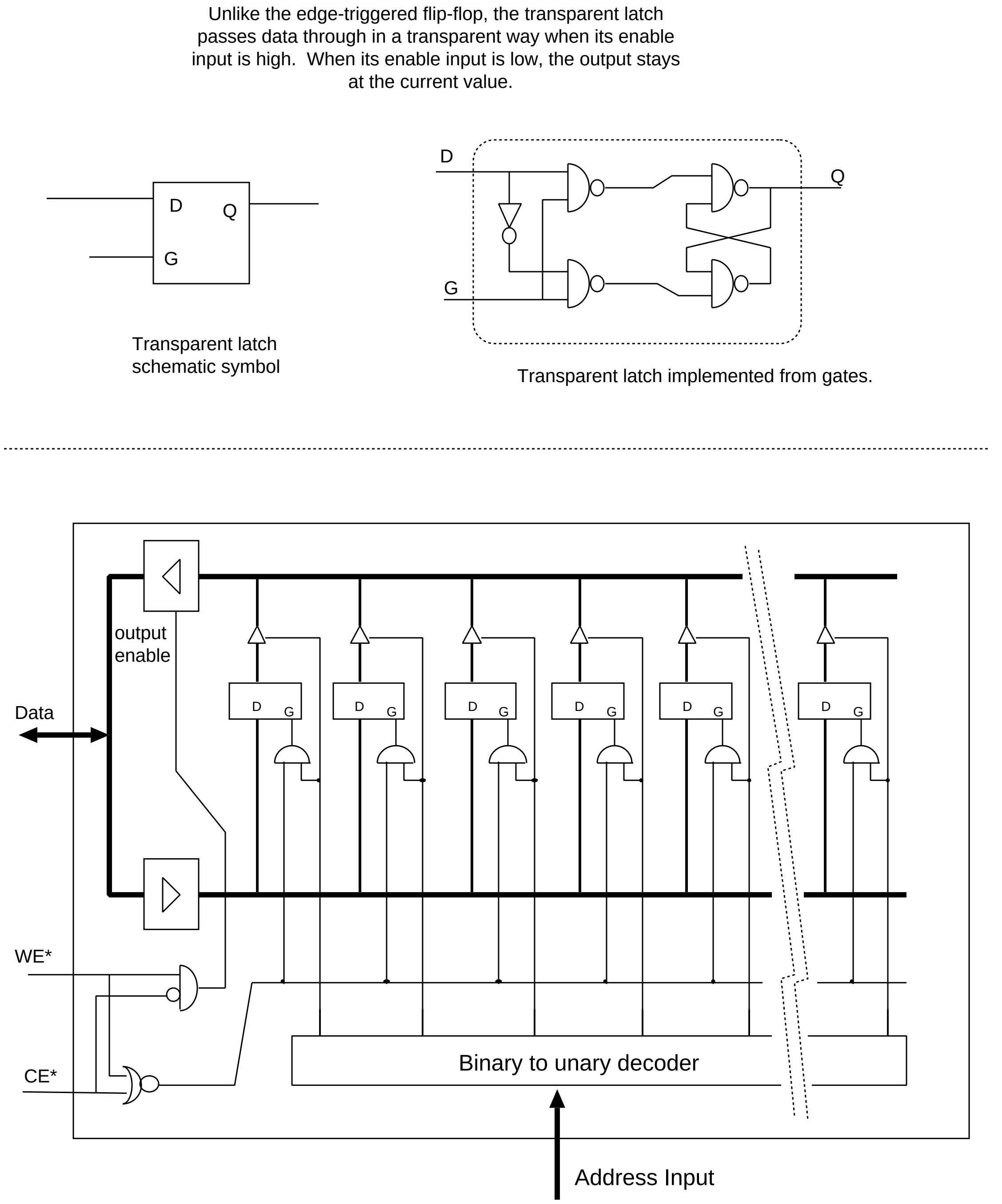
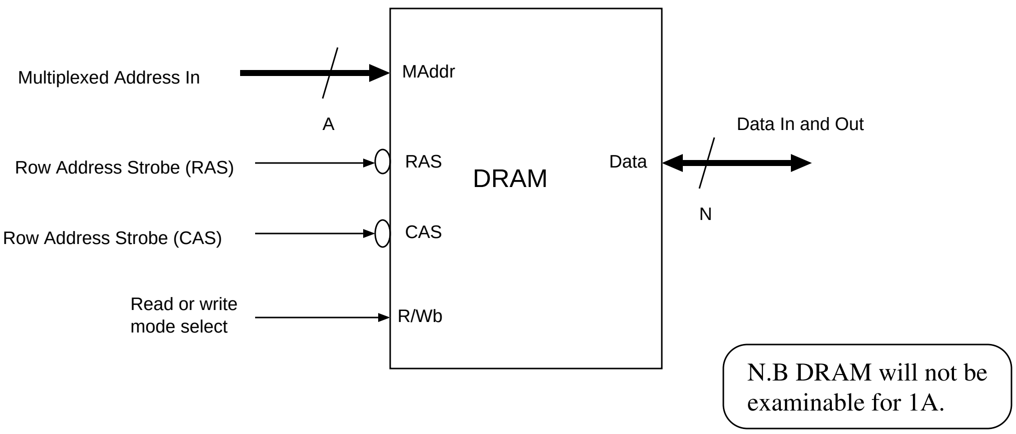
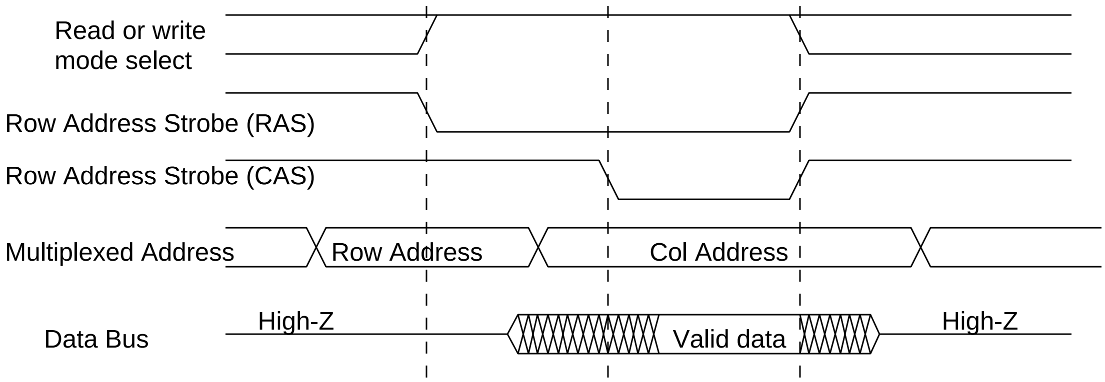


Figure 7: Schematic and possible implementation of a transparent latch and a naive implementation of an SRAM using broadside transparent latches.



Read Cycle (write is similar)

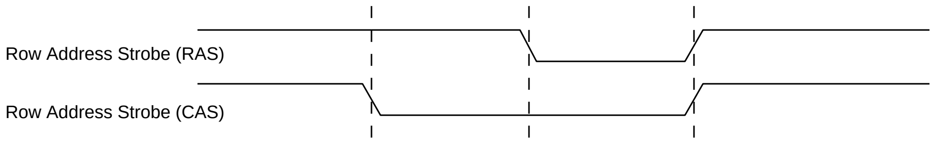


A DRAM has a multiplexed address bus and the address is presented in two halves, known as row and column addresses. So the capacity is 4**A x D. A 4 Mbit DRAM might have A=10 and D=4.

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.

Figure 8: Dynamic RAM (DRAM).

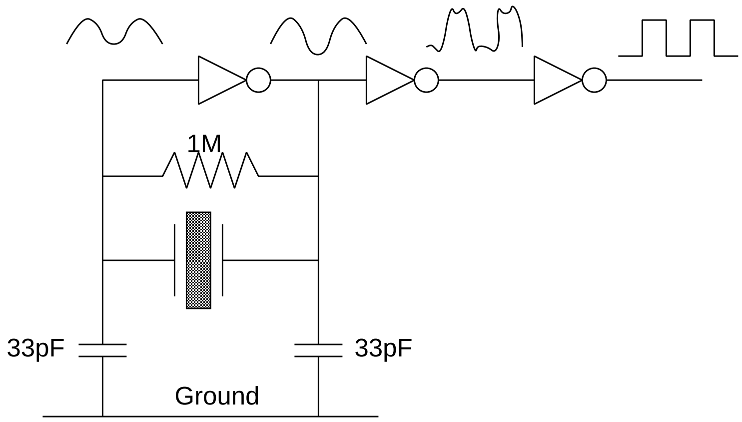


Figure 9: A crystal oscillator clock source.

one batch at the end of every 4 millisecond period makes the design of power supplies much more difficult.

Modern EDO (extended data out) DRAMs guarantee that the data will be valid on the output for a period after CAS is deasserted, thereby easing system timings and helping the designer.

DRAMs are often packed on SIMMs (single in line memory modules). Capcities of SIMMs are around 16 to 32 Mbyte with prices of tens of pounds per SIMM.

High performance systems today use synchronous DRAMs (SDRAM) which have a clock input incluce broadside registers on the address or data connects. This gives a pipeline delay (section 2.6) but allows faster operation (e.g. 100 Mwords per second).

Exercise: When would you use SRAM and when DRAM ?
Exercise: Sketch the circuitry that could be put around a bank of DRAM to make it appear like an SRAM with a single, non-multiplexed address bus and similar control signals.

1.4 Miscellaneous Components

Figure 9 shows a circuit frequently used as a clock source. **A carefully cut slither of quartz crystal is used as the timing reference.** Quartz is pizoelectric, so can be made to change shape by applying an AC signal across metal coatings applied to its sides. Also, as it changes shape, it generates an equivalent electrical signal. Since the speed of sound in quartz is very high, a slice about 100 microns thick will resonate at several MHz. The details of the circuit are beyond the scope of this course, but suffice to note that the sine waves generated by the crystal become good quality square waves, suitable for use as a clock, after passing through a few inverters. A crystal might cost one pound to add to a design, but is accurate to about 50 parts per million in frequency, so is highly useful.

Figure 10 shows a cheaper circuit that serves as a clock when accuracies of only a few percent are needed. The first inverted must be a special Schmiddt inverter with abrubt switching and hysteresis for the circuit to work. The capacitor is the only component that is hard to make on an integrated circuit, so a single pin on the device serves to connect the external capacitor. The crystal oscillator always needs two pins.

Figure 11 shows the typical arrangement of clock distribution inside a large, modern IC. The clock is actually provided from a lower-frequency external reference and multiplied up internally with a phase-locked loop. The multiplication factor may be higher than the value of 2 shown: for instance it might be 10. **The clock is brought on to the chip at a lower frequency than required since it is easier to handle lower frequency signals on PCBs and the lower frequency is one that can be directly generated by a crystal.** Skew in the delivery to the various parts of the device is minimised by using a balanced clock distribution tree. Inverters are used instead of buffers to minimise pulse shrinkage (duty-cycle distortion).

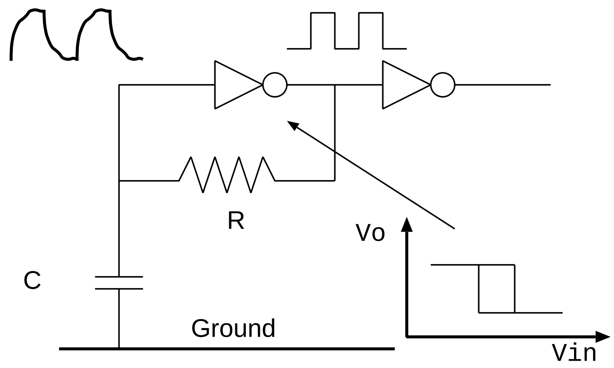


Figure 10: An RC oscillator clock source.

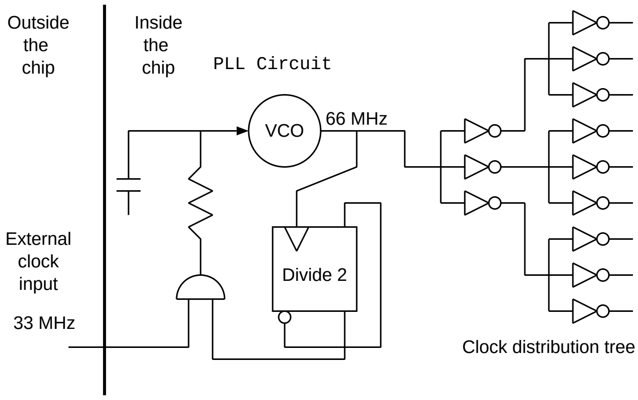


Figure 11: Typical clock multiplication and distribution system for a large IC.

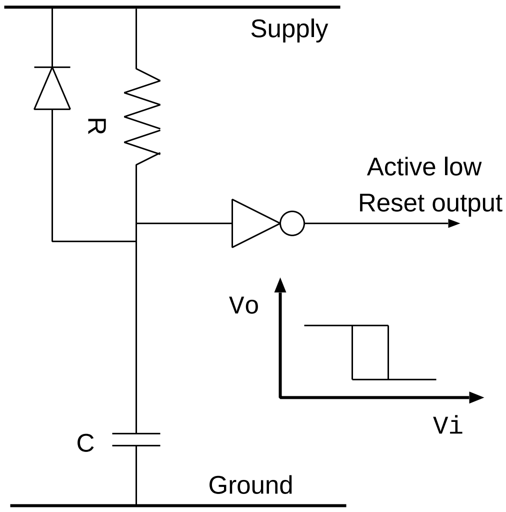


Figure 12: A circuit often used as a power up reset.

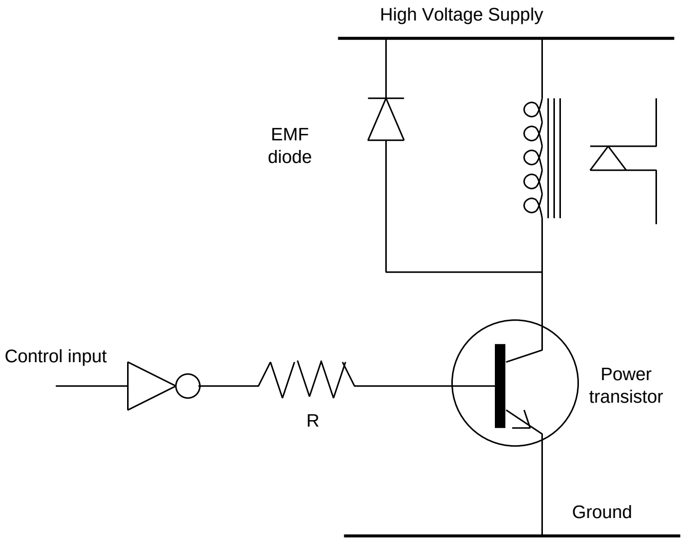


Figure 13: A typical structure used to drive a heavy current or high-voltage load.

Figure 12 shows a circuit that is often used to generate a reset signal when a system is switched on. When power is applied, the capacitor is initially discharged, so has zero volts across it. Therefore the output is a logic one. After a while, the resistor charges the capacitor such that the input to the invertor crosses the switching threshold of the Schmiddt input and the output from the circuit is deasserted. Typical delay times used are about half a second, which is long enough for the power supply to become stable and for oscillators to settle down. The diode discharges the capacitor into the supply rail when power is removed. On most devices, a reset switch is also fitted, which discharges the capacitor when pressed, therefore starting the device as from power up. **The reset signal is normally arranged to reset every flip-flop in every chip** (except for static RAM cells).

Figure 13 shows a typical structure used to drive a high power load from a logic signal. The semiconductor processing used to fabricate logic devices may be too expensive to waste on large transistors or may not be able to tolerate the higher voltages required to control the load. Therefore, external driver transistors in individual packages or in power driver ICs are used. Typical applications are motor, loudspeaker, solenoid, relay, and print head driving.

Figure 14 shows the circuit typically used to get a clean signal out of a mechanical switch. In such a switch, when one contact hits another, it bounces off with a supersonic ping, which means that the circuit is made and broken a few thousand times for each switching operation executed. Using a two-pole switch and an RS latch formed from a pair of cross-coupled NAND gates, a clean output is generated. The systems works on the basis that the *first* time the commuting contact hits the stator contact, the latch changes state, and stays there until the switch is moved all the way back to the other side, where the reverse happens.

1.5 ALU and ALU circuits

The arithmetic and logic unit (ALU) is a fundamental block of all computers: it is the basis of the integer execution unit. Figure 15 shows the normal schematic for an ALU. The ALU is combinatorial, but is shown connected to a flag register, which does have a clock input. For a detailed example, look at the 74181 in the Texas System 74 databook.

The illustrated ALU has two N-bit inputs and a 4 bit function code input. The output is also N-bit. The function code determines what function of the two inputs is computed. Typical functions are add, add with carry, bitwise AND and OR, subtract and identity functions of the two inputs. The instruction set for the ARM microprocessor contains 16 data manipulation instructions which are a good example of the 16 most useful functions that an ALU can perform.

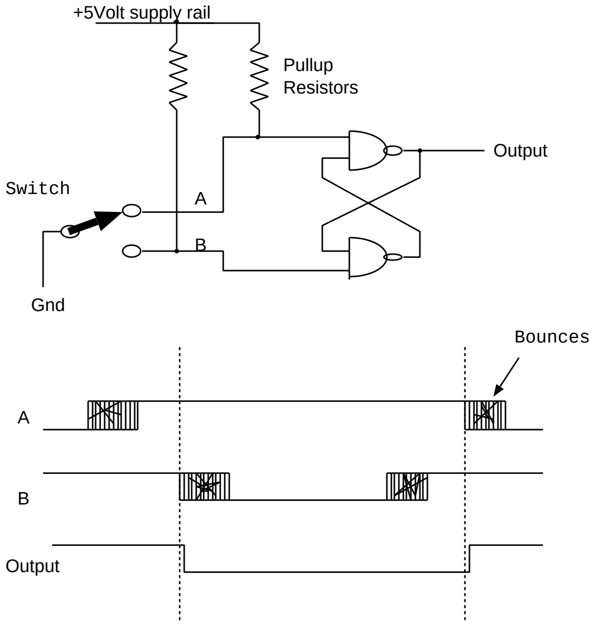


Figure 14: A debouncer circuit for a two-pole switch.

Like all combinatorial logic functions, the ALU is guaranteed to compute its output within a fixed time interval from the last input change. The value of this delay is typically dependent on the carry chain speed inside the ALU and it will set the maximul clock frequency of a simple microprocessor.

Question: Suggest why ALU's typically only support addition and subtraction as arithmetic operations and not multiplication or division ?

Figure 16 is a nonsense circuit designed to show the interaction of an ALU and a register file. **These two units together form the heart of the execution unit of a computer.** In the circuit, the addressed register is both read and written at the same time. Even though the address is incremented each time, it is important to understand that the old value read is updated in the ALU and written back to the same location. The only possible interference between one register and the next is through the carry flip-flop, which will allow the output of one ALU operation to affect the behaviour of the next.

Two function generates are illustrated. These are just combinatorial blocks and could be considered as ROMs. To achieve some particular function these could be hardwired or programmed. In very early computers plugboards were used for such functions.

If the ALU function code generator is programmed to specify 'A+B' when the input is one and 'B+Cin' otherwise, and the A input function generator is programmed to produce a constant value of 1, we will have programmed up a very large counter. **Register zero is incremented each time and the others only when a carry is generated in the previous addition.** Will this composite counter complete its cycle this universe lifetime ?

1.6 Microprocessor Circuits

Figure 17 shows the schematic symbol and internal block diagram for a microprocessor. The internal block diagram is beyond the scope of this course, since it is coupled with a subject known as programming, but the external view is fair game.

The main features of the microprocessor are that it has a clock and reset input and it has an address bus output and a bi-directional data bus. The microprocessor generates read and write cycles on its busses and these are essentially identical in form to the waves shown in figure 6 for the SRAM.

The microprocessor places addresses on the address bus and sets the read/write signal to show

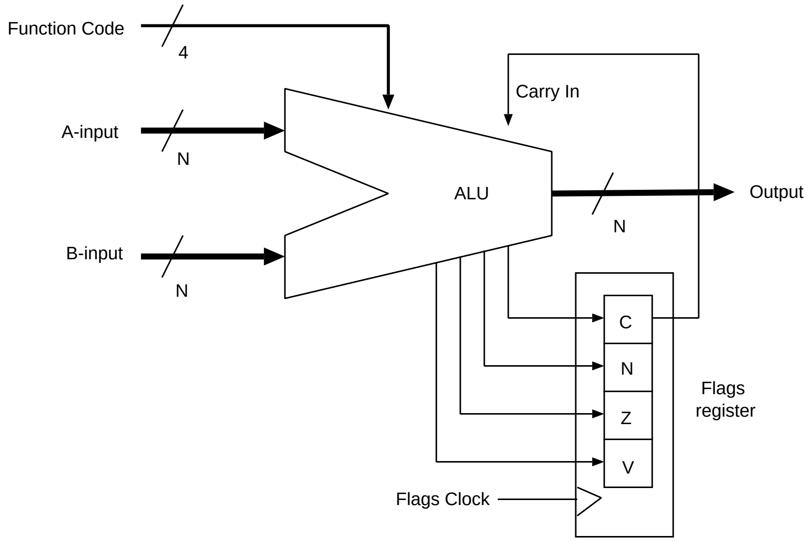


Figure 15: Schematic symbol for an ALU, connected to a flags register.

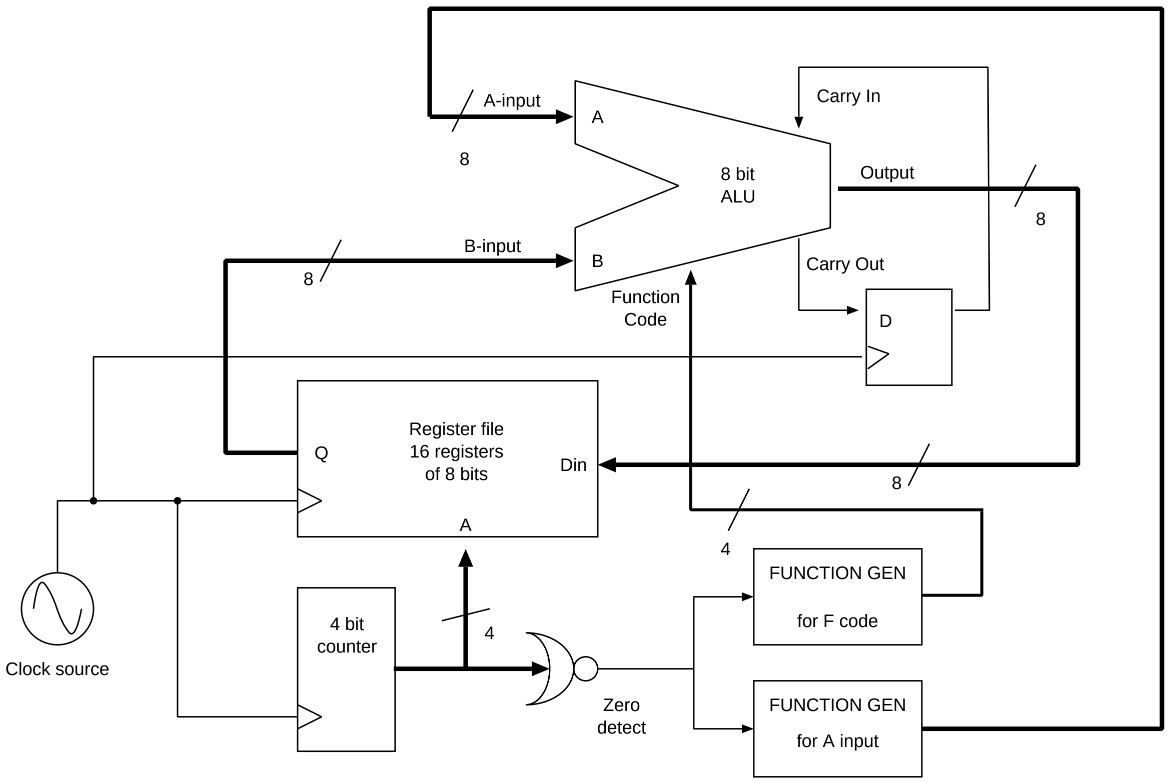


Figure 16: Example structure using an ALU and register file.

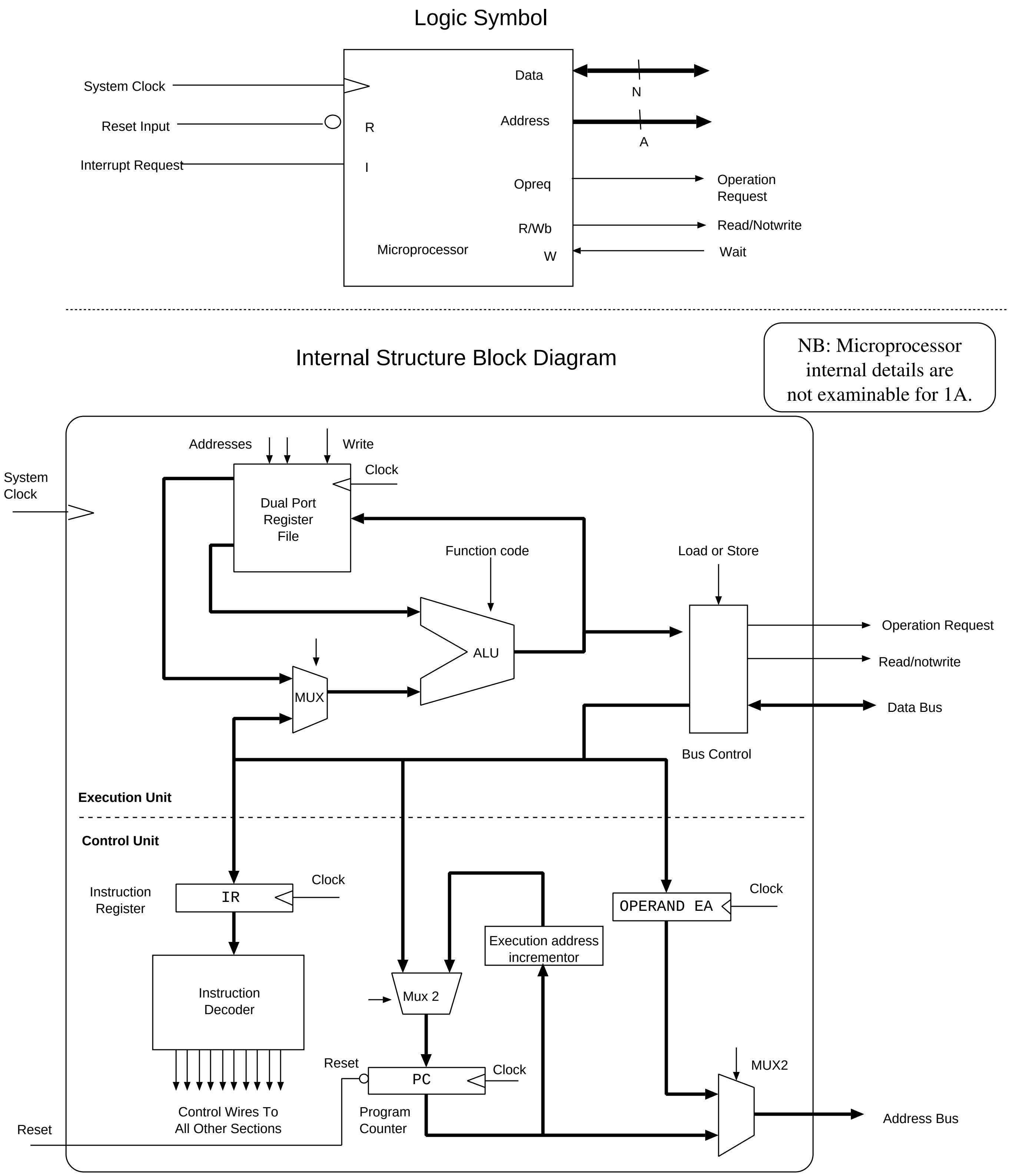


Figure 17: A microprocessor logic symbol and simplified internal structure.

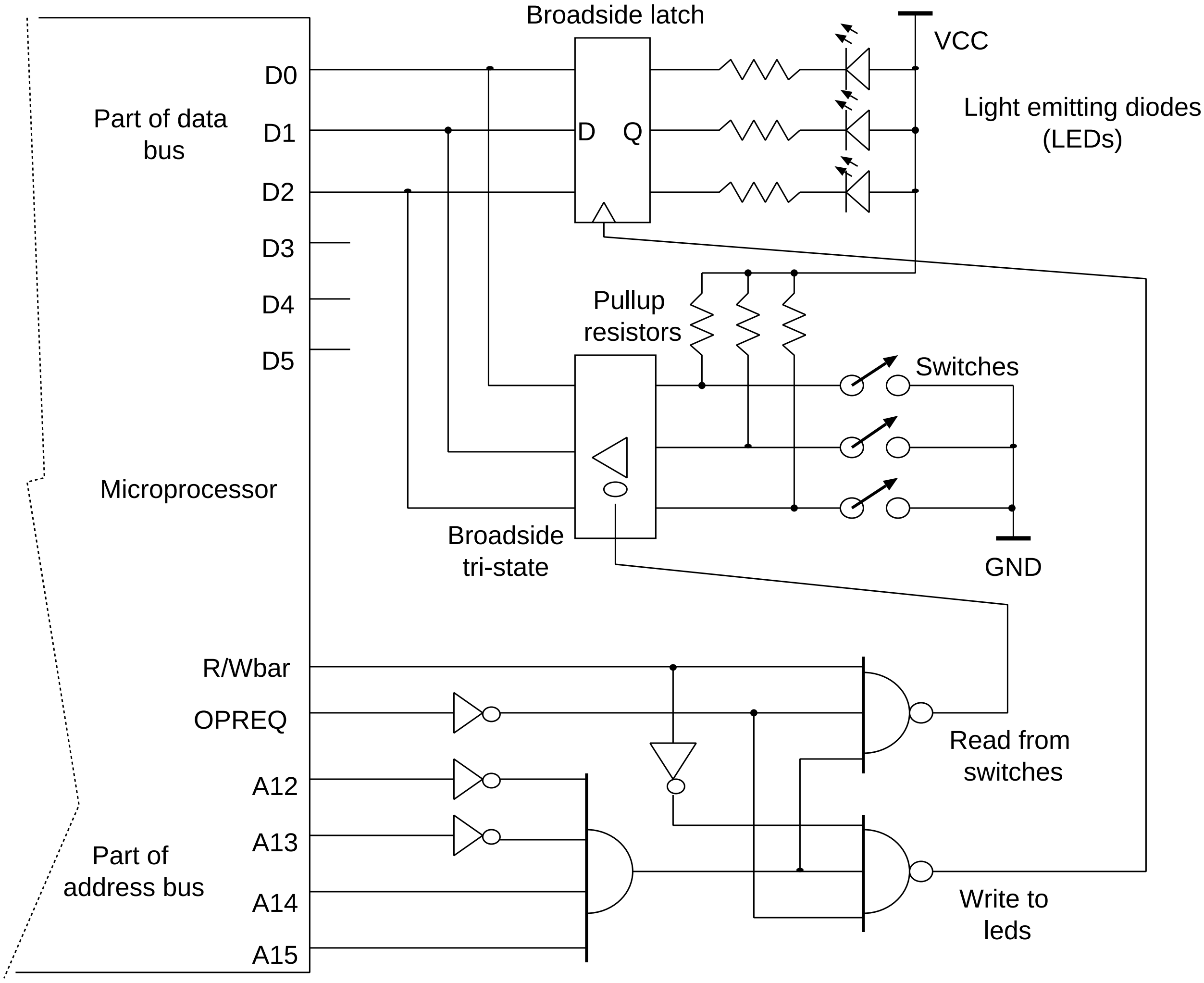


Figure 18: Example of memory address decode and simple LED and switch interfacing for programmed IO (PIO) to a microprocessor.

the direction of the intended transfer on the data bus. It then raises its operation request output (opreq) and either places the data to be written on the data bus (for a write) or expects the addressed device to drive the data bus with its information (for a read).

The behaviour of the microprocessor is to fetch an instruction from the reset locataion and execute it. If the instruction is a jump, then the next instruction may not be the one following it. Executing an instruction either modifies internal registers in the microprocessor or loads or stores data to the devices connected on the address and data bus.

Figure 18 shows how to connect some LEDs and switches to a microprocessor for simple programmed input and output. Only the low order data bits are used, so when reading from the switches, random values may appear on the higher order bits. Conversely, when writing to the LED registers, the higher order bits are ignored.

Exercise: If the address bus is 16 bits, how many addresses are consumed by the switches and leds. If the data bus is 8 bits, what percentage of addressable bits have been wasted and is this a sensible design ?

Figure 19 is the circuit of small, microprocessor based computer. The microprocessor can address 2^{16} locations of eight bits (65536 bytes) and some of these are filled in with devices.

Exercise: What would happen if the microprocessor tried to write to the read only memory ?

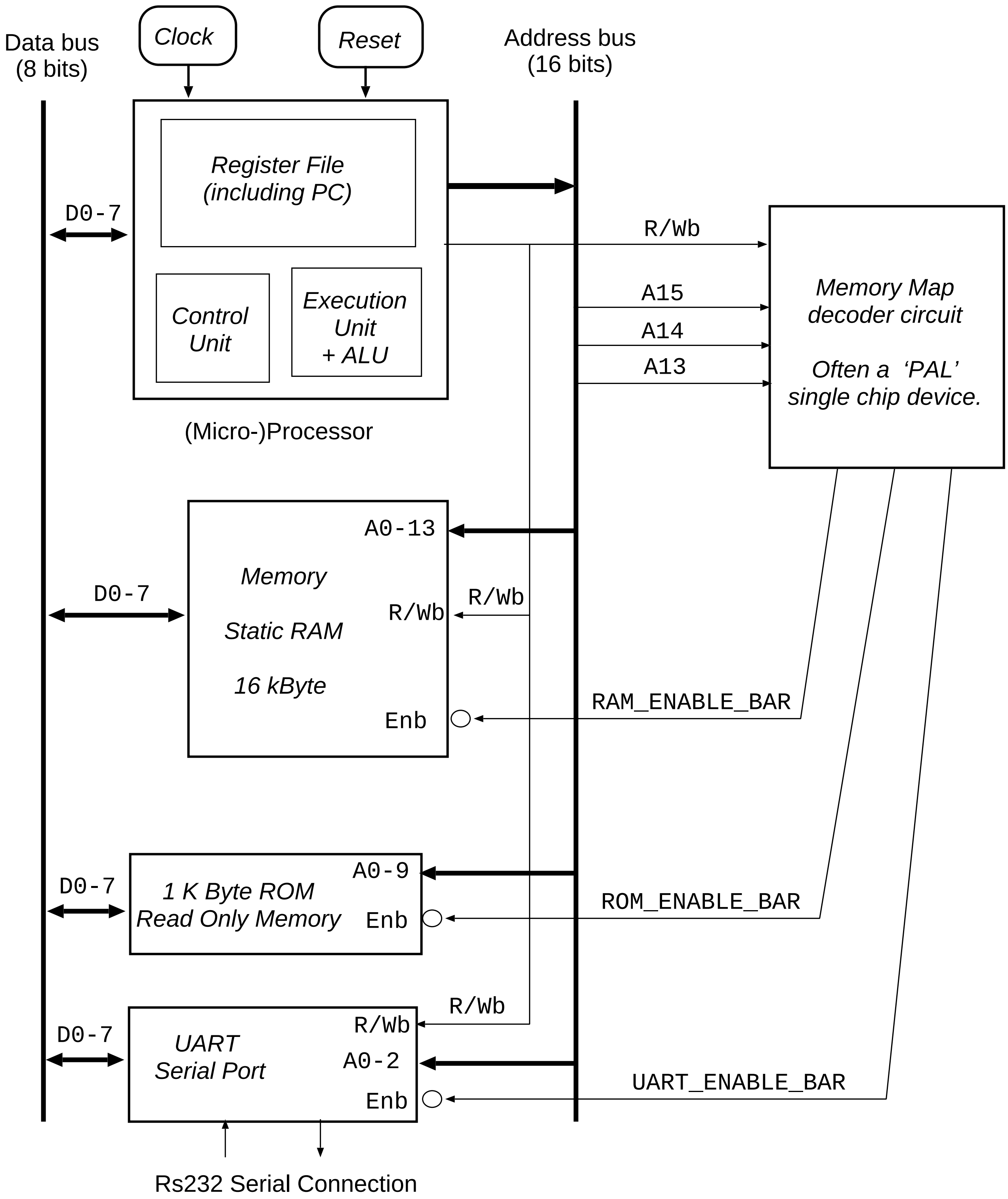


Figure 19: A small computer (A=16, D=8).

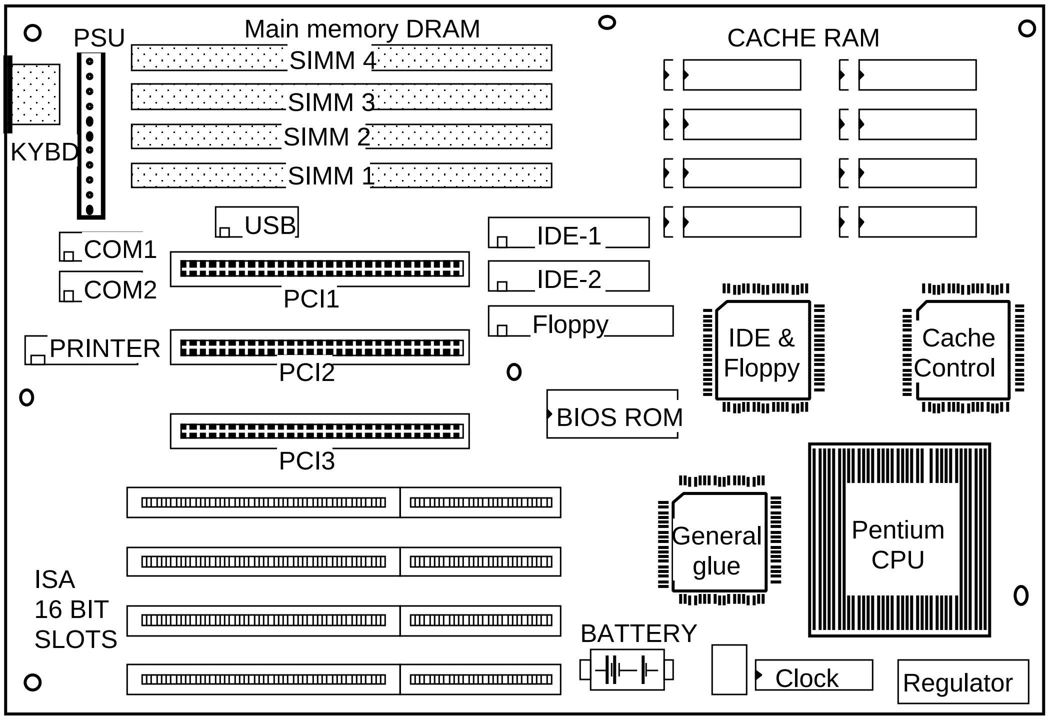


Figure 20: PC motherboard, 1997 vintage.

1.7 The PC as a component.

The PC motherboard has evolved in a backwards-compatible way for 20 years. The mounting holes and connectors and architecture are largely unchanged, and the changes which are not back-compatible have been controlled and rare. Today, the PC motherboard offers exceptional value for money: 300 MHz processor, 32 Mbyte RAM and the logic for many useful interfaces (keyboard, USB, VGA, Floppy, IDE, IRDA, etc.) all for under 200 pounds. A typical PC motherboard is shown in figure 20. In addition, a huge variety of hardware devices is available to plug in.

Smaller, off-the-shelf, euro-boards containing a PC have dimensions 100 by 160 millimeters and can easily slot into a rack-oriented hardware design. Putting a PC on a custom chip is a possibility for the near future, especially if memory and CPU levels are relaxed a 3 to 4 years' worth of performance gains.

It is easy to think of a PC as something that always has a screen and a keyboard and runs Microsoft code. However, the PC motherboard is a general purpose computing platform that, given the correct software, is just as happy providing, for instance, the processor inside an airport arrivals VDU cluster. In this example, a video display board and a network board would have to be fitted.

1.8 Printed Circuit Boards

Nearly all designs require the fabrication of new printed circuit boards. A cheap PCB has one layer of wiring. Most digital boards have 4 layers, with the internal pair just used for power. Some high-tech boards need more layers of wiring. See examples in lectures.

2 Joining Modules Together and Clocking.

Learners' Guide: What you should learn from this section is:

- Real systems are made of interconnected modules and have more than one clock.
- Distributing a single clock reliably requires care.
- When signals have to cross from one clock domain to another the design should accomodate that there will be skew and metastability.
- The maximum clock rate is normally set by the longest path of logic.

2.1 Connecting Boards Together.

When boards are interconnected, some of the main issues are:

- How much data in bits per second needs to flow?
- Will the connection be synchronous or asynchronous?
- Is flow-control needed to limit the rate of data flow?
- How long do the wires or cables have to reach?
- Is it desireable to carry power with the cables ?
- Is hot-pluggable mode needed ?
- Is the interconnection topology fixed at design time or does the system need to adapt to various configurations?
- Should we use an existing standard or design a new system ?

Major electronics issues when connecting boards together is whether the cables will give off interference and whether the electronics at the ends will be damaged when people with a large static charge touch the exposed pins.

The bits stored in a data cable is given by the signalling rate multiplied by the length divided by 200 m/gmus.

2.1.1 Parallel Port

With the parallel port, data is transferred at a rate dependent on the slower of the sender and the receiver, since the transmitter is not allowed to strobe the next byte of data when the busy signal is asserted.

Modern parallel ports support bidirectional transfer of data, but originally they were just designed for printer connection.

The reach is potentially dominated by skew of the parallel signals, since they must all be valid at the receiver before the strobe signal. Flow control is provided by the busy signal.

Exercise: Figure 21 shows the signals involved in a parallel port, as frequently used to connect printers to computers. The basic operation is that the master places a byte of data on the data bus and then strobes the strobe signal low until acknowledge is asserted by the device. The master must not present a new byte to the printer while the printer has put busy high. Design the parallel port interface logic by drawing on the ideas shown in figure 18. To do this, it is sufficient if the busy signal appears to the processor like one of the switch inputs and the data output is rather like eight LEDs. Considering that the processor really does not care about the acknowledge signal or when the strobe signal is deasserted, you might care to use an RS latch to help the processor generate the strobe signal.

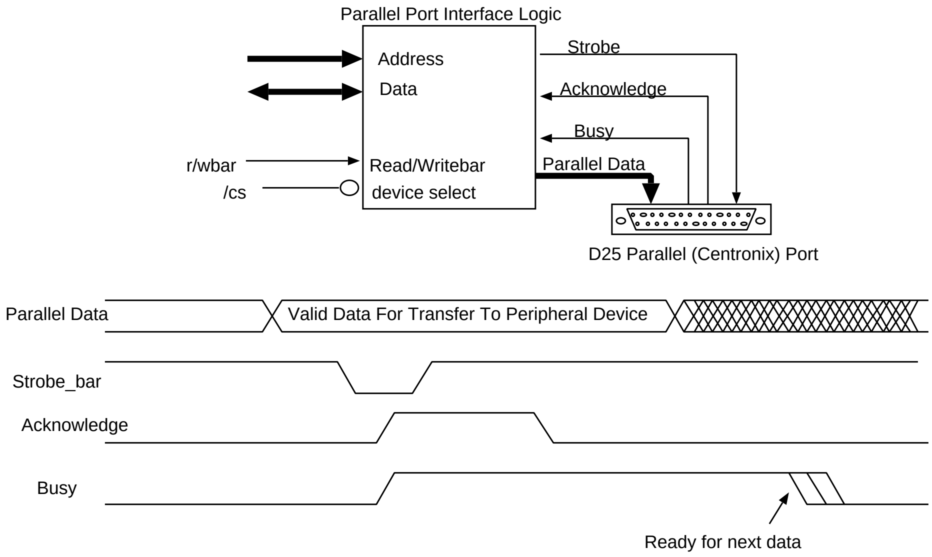


Figure 21: PC or Centronics Parallel Printer Port.

2.1.2 Serial Port

The serial port uses one pin for each direction, whereas the parallel port sends only (in its basic form) and uses 8 data pins to transfer a byte.

With the serial port, the receiving device must be at the same baud rate as the transmitter, to make sense of the data. With only one signal, there is no skew problem, but the maximum transition rate on the data wires is much faster. **The baud rate is the number of signalling events per second on the line.** In a binary system, the baud rate is the bit rate. A serial port with a start and stop bit per byte achieves 80 percent of its bit rate as useful throughput.

2.1.3 PS2 and Keyboard port

The original 1977 IBM PC introduced a connector that has become widely used for PC keyboards and mice. It is a point-to-point connection that carries power, although the total current available is limited by a fuse or similar protection device. The connection is bidirectional and byte-oriented. A pair of open collector lines are used for clock and data. When one end wants to send data to the other, it pulls down the data line, forming a start bit, the other end then clocks the data line until a byte has been received.

The PS/2 port is unsuitable for high bit rates or long cables since the cable length must not be longer than a clock wavelength.

2.1.4 USB port

The Universal Serial Bus uses similar electrical technology to the PS/2 port, but the protocol for using the two wires is different. Small hubs, perhaps integrated into the back of a keyboard, allow a desk-top network to be made. Addressing and topology determination are supported and the data rate is 12 Mbps, or 1.5 for slow devices.

2.2 Clocking and Synchronous FSM Combination.

This section looks at aspects of clock signals and considers the how, when and why of multiple clocks in a system. The decisions relating to clocks are fundamental to any hardware design and are normally taken very early in the process.

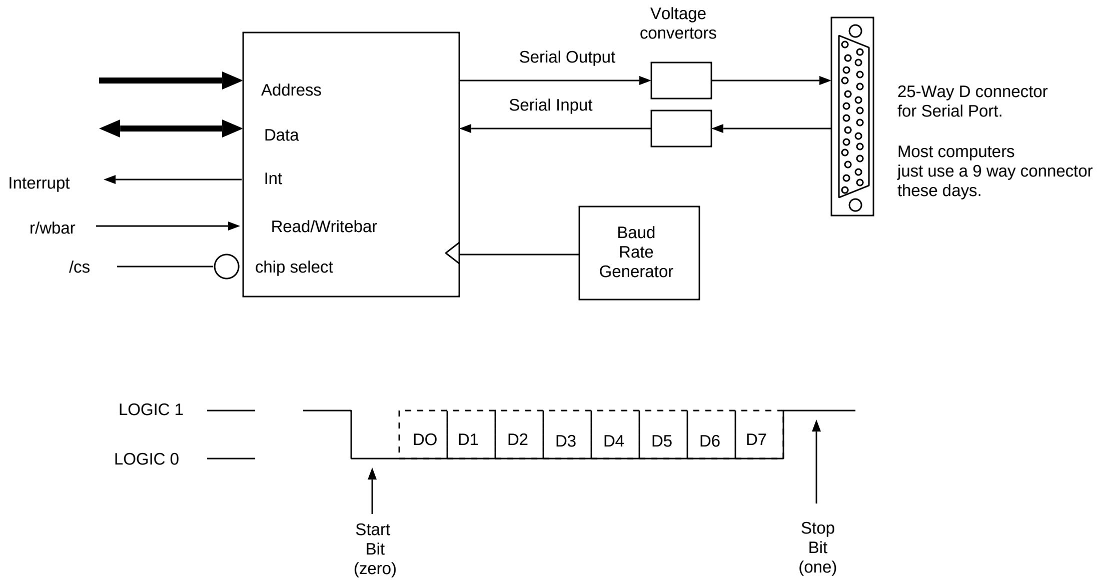


Figure 22: Serial port or UART arrangement

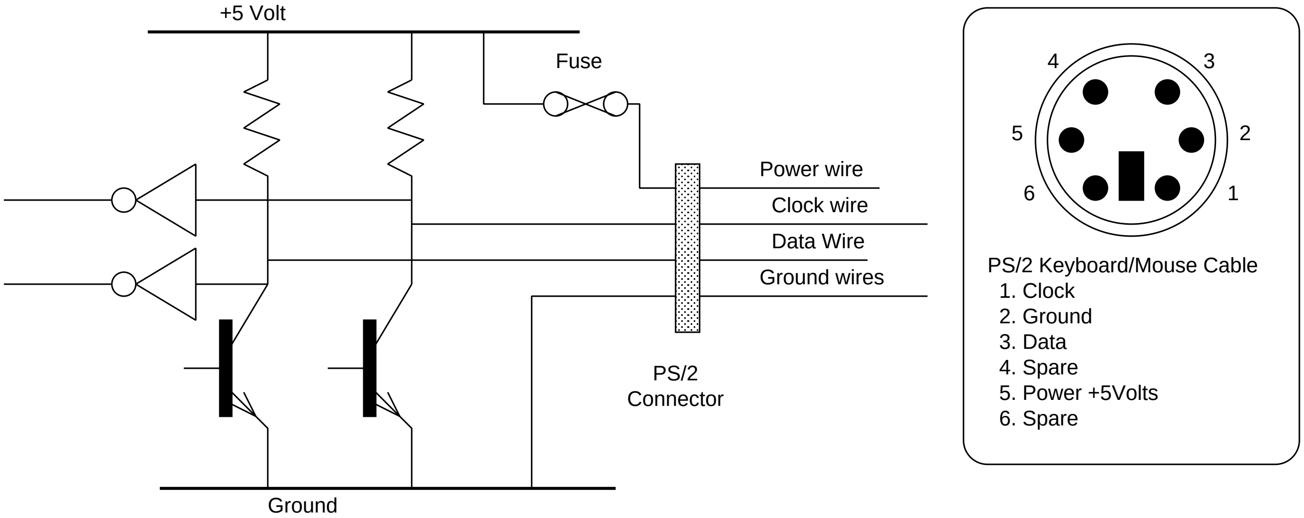


Figure 23: Keyboard and/or PS/2 port.

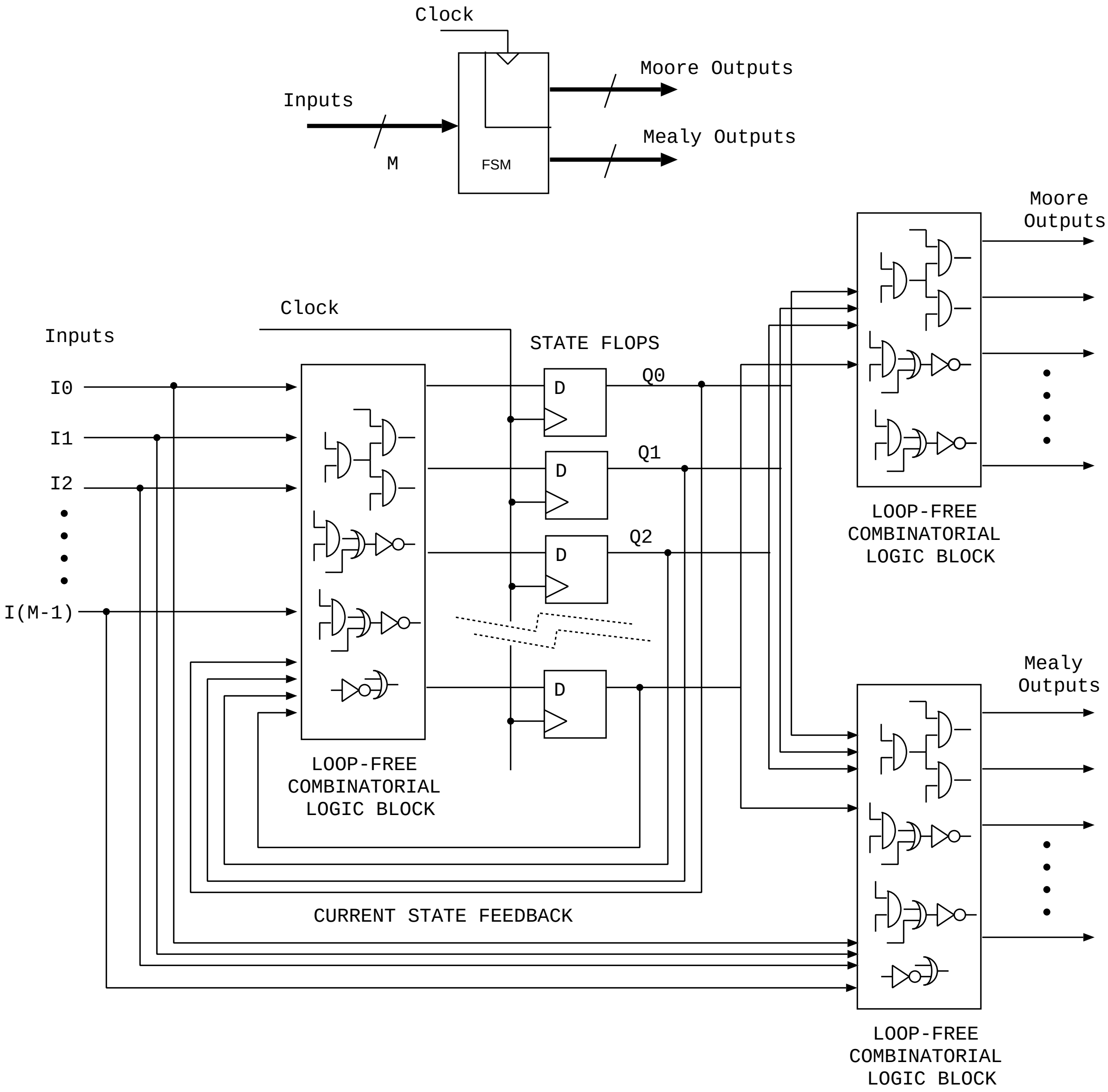


Figure 24: Canonical synchronous, finite-state machine.

2.3 Finite State Machines.

Figure 24 shows the schematic symbol and canonical circuit for a finite-state machine. It is a mantra among mature hardware designers to consider everything as a collection of finite-state machines. Beginner hardware designers tend not to have this approach, and instead think nothing of treating the clock input to a flip-flop as suitable for wiring to anything convenient. They do not get far like this, since contemporary CAD tools, design techniques and formal methods will stop providing any help and circut bugs will creep up everywhere. (An exception is that there are certain *licensed* asynchronous hardware designers, like Dr Simon Moore, who often don't even have clocks in their circuits.)

The FSM has inputs and a clock. It has two types of outputs. Moore outputs depend only on the current state. Mealy outputs may also depend on the current inputs.

Figure 25 shows a D-type flip-flop and its three significant timing specifications. The flip-flop input must not change at the same time as the positive edge of the clock: indeed it must be stable for a period before (the set-up time) and remain stable a period after (the hold time). The output will change after a propagation delay time from the positive edge of the clock. The negative edge of

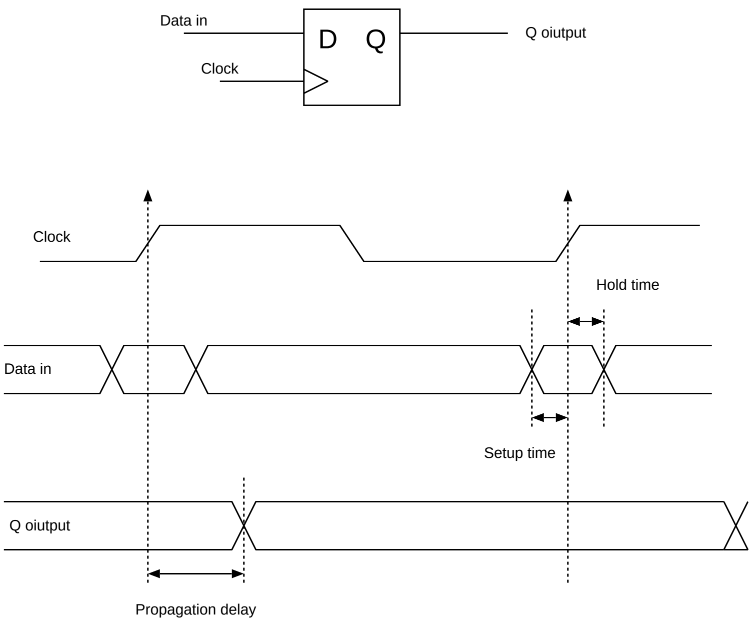


Figure 25: Timing specifications for an edge-triggered D-type flip-flop.

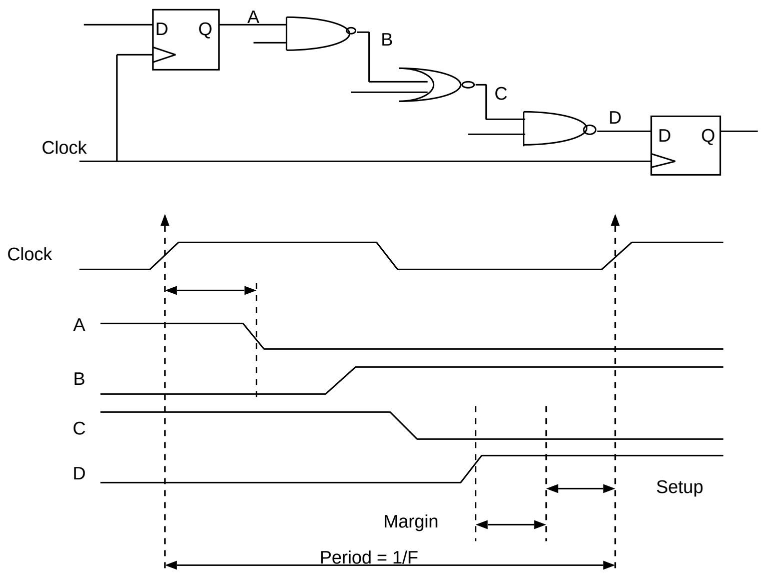


Figure 26: Typical nature of a critical path.

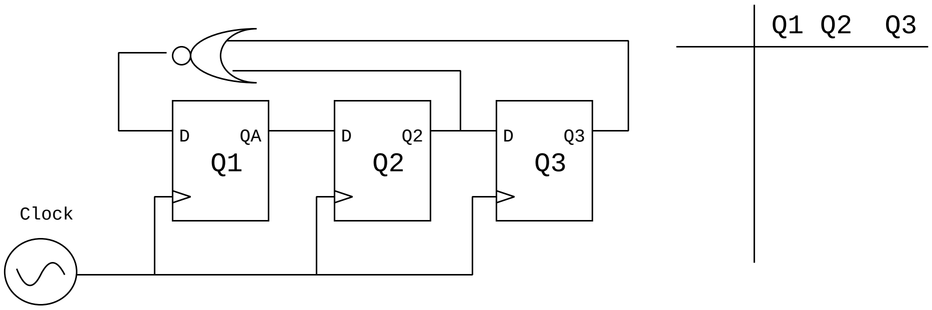


Figure 27: A counter that does not use binary.

the clock has no effect. The J and K inputs of JKs and the T inputs of Ts have the same set and hold requirements.

In a synchronous FSM, the maximum clock frequency is normally determined by the longest path of logic which ends at the input of a flip-flop. This path is know as the *critical path* and is illustrated in figure 26.

2.4 Speed Increase with Johnson Counter

A *Johnson* counter is based around a shift register. Figure 27 shows a Johnson counter that divides by five. The significant features of Johnson counters actually become clearer for larger counters, but they are twofold: 1. the division ratio is restricted to twice the number of flip-flops (e.g. 5 flip-flops can give a divide by 10 at maximum) and 2. the number of logic gates is always low, and so the counter is fast.

Exercise: Design a Johnson Counter that also has a control input that makes it divide by six or seven. Optimise your design for two features: 1. maximum clock speed, 2. that the ratio input can be changed at any time without causing a hazard. Note: clearly, metastability cannot be avoided, but by being hazard free, we mean that the state path taken will always be one or the other of the two desired paths, whatever happens. Why might the second feature be important in practice (hint: consider derived clocks) ?

2.5 Speed Increase with One Hot Coding

In *One Hot* coding, at all times, the flip-flops in a counter or FSM are arranged to be all zero except for one that is one. This clearly minimises the number of gates required to decode the current state and can greatly reduce the number of gates that determine the next state.

2.6 Speed Increase using Pipelineing

Figure 28 shows a circuit before and after the addition of a pipelining stage. The pipeline stage may have a fewer or greater number of flops as the input or output. The benefit of the pipeline stage is that a complex combinatorial logic function is split into two halves, thereby reducing the maximum length of unclocked logic leading to the inputs of the output flip-flops. The disadvantage of the pipeline stage is that data takes longer to be processed. This disadvantage is not often a problem: for instance, if the data has just come down a link in a network, then the additional delay will be just as if the link were a few meters longer. More than one stage of pipelining can be inserted if needed.

Some modern logic synthesiser tools will insert pipeline delays where possible, if helpful. Others will back-propagate them if they are instantiated at the outputs.

Exercise Sketch the circuit of a large adder with ripple carry and then modify it to include a pipeline stage. Note, the output signals should always reflect the result of a single addition in any one clock cycle. Explain or estimate the effect of the pipeline delay on the maximum clock frequency.

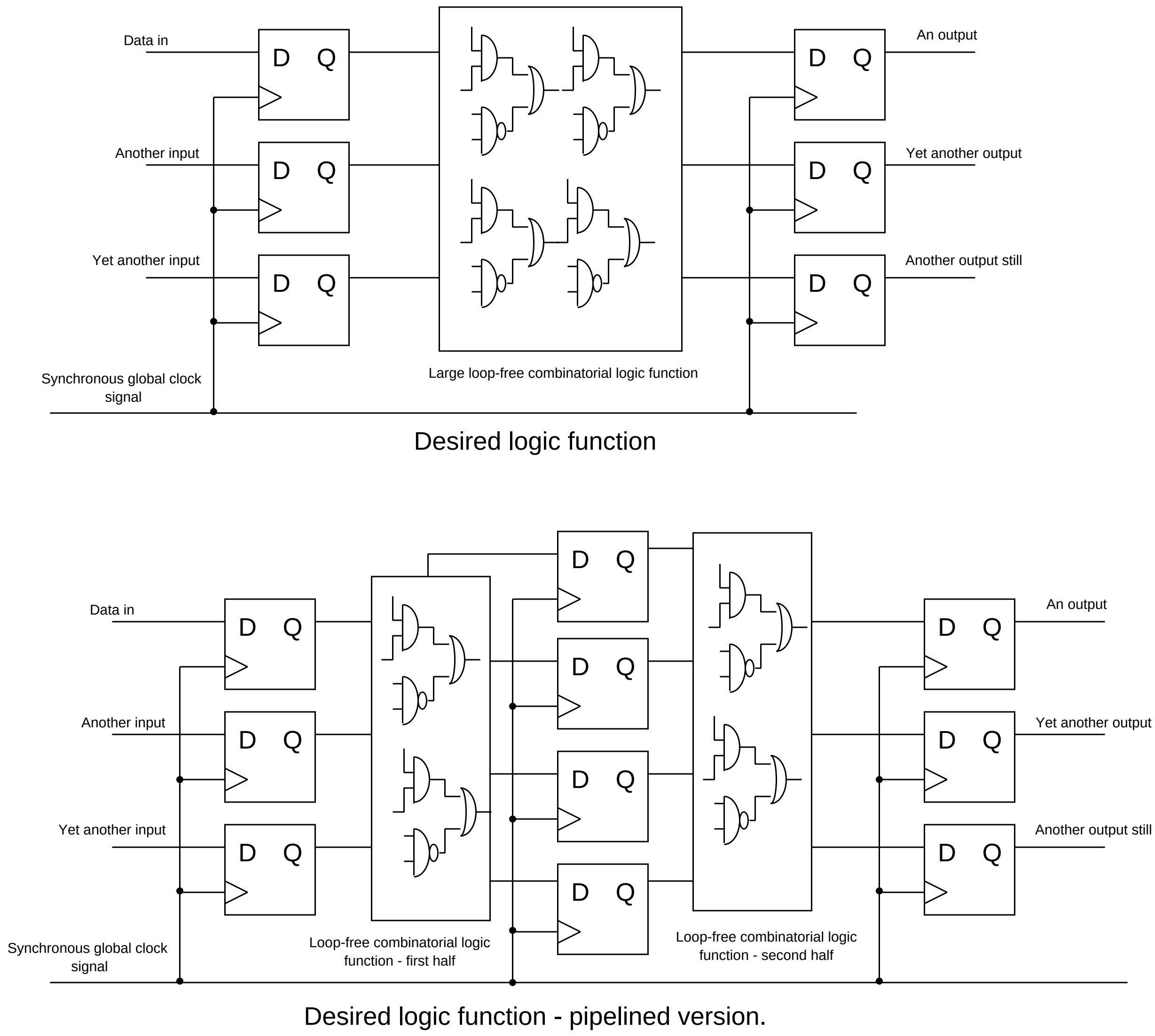


Figure 28: Example of introducing pipelining to increase throughput, but also delay.

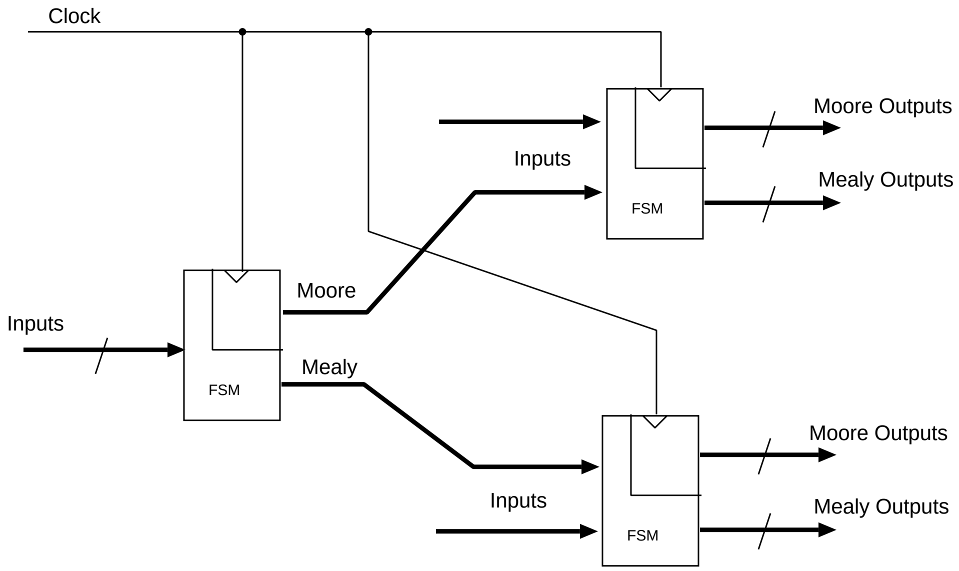


Figure 29: Cascading FSMs to produce larger circuits.

2.7 Derived Clocks.

Figure 29 shows a concatenation of finite-state machines running from the same clock. Of course, canonicly, these together simply form one larger FSM. However, the decomposed view is vital in practice, since the density of wiring between FSMs will be lower than within the FSM and the functionality, authorship, history and even ownership of the IPR (intellectual property rights) of the FSMs will be different.

If we feed a Mealy output into another FSM, delays accumulate, reducing maximum clock frequency. This does not happen with Moore outputs. Moore outputs should be used if possible, but for some designs, a Mealy output is needed in order to reduce processing delay of the unit as a whole.

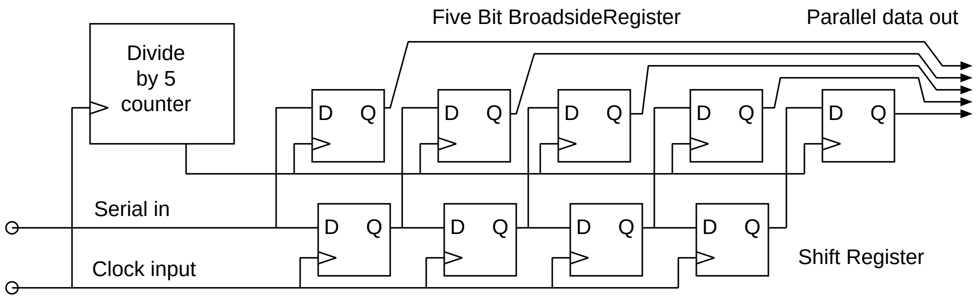


Figure 30: An example that uses (badly) a derived clock: a serial-to-parallel convertor.

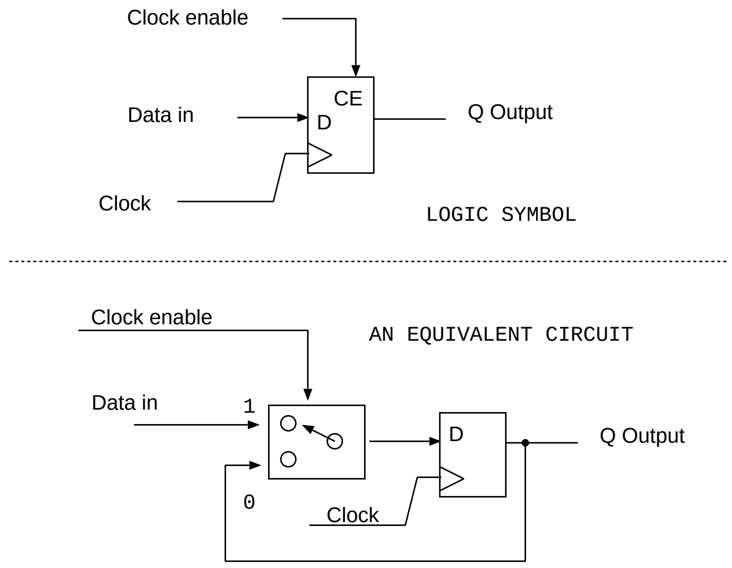


Figure 31: A D-type with clock enable and the equivalent circuit.

Figure 30 shows a typical example of the use of a derived clock. Power consumption is increased in parts of a circuit where fast clocks are used, and so using lower frequency clocks where needed is an important design consideration.

The serial-to-parallel convertor can be redrawn as a pair of FSMs as shown in general in figure 35. It is then clear that the design rule about two Moore outputs from the master not changing at once cannot be applied. Therefore what can we do ? This is not easy. Real hardware often is not easy.

Exercise: Give the schematic for a reciprocal parallel-to-serial convertor. Does combining the two convertors result in the identity function ? If not, why not. Note: they can be combined in both orders.

2.8 Gated and guarded clocks

Sometimes we do not wish a flip-flop to change its value every clock cycle. Clearly, J-K device support this by putting both inputs low. For subsystem FSMs or broadside D-type registers, we need the concept of the clock enable.

Figure 31 shows a D-type with a clock enable input and the equivalent circuit. The complexity of the additional multiplexor is not present in a real implementation, since it can be adsorbed into the circuit which makes the D-type.

Figure 32 shows the safe way to gate a clock. The clock is OR-ed with the negated enable signal. In this clock gating method, the derived clock is kept normally high, except when a clock pulse is

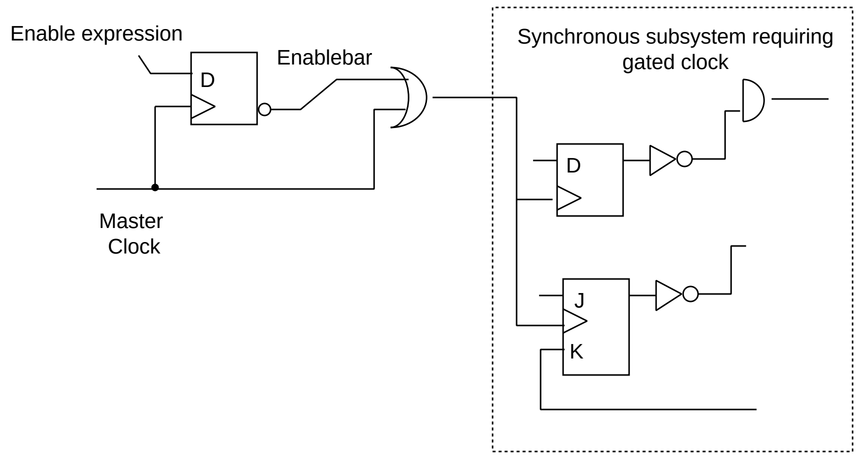


Figure 32: A 'safe' alternative to using clock enables ?

to be let through. Other arrangements for gating clocks tend to produce glitches in the derived clock.

Gating clocks should be avoided normally. If flip-flops have clock enable inputs these should be used instead. Providing your own multiplexors before the flops is also a good alternative. Clocks do have to be gated at times though, most commonly in generating a write strobe to an SRAM (figure 6).

Exercise: Design the logic to place in front of a J-K flip flop that turns it into a D-type with clock enable input.

Given that from Digital Electronics you know that a J-K can be equally as simple as a D-type to implement from transistors or gates, it is reasonable to assume that a clock enable D-type is also as simple. Therefore why should we ever use gated clocks instead of clock enable inputs ? The answer is that if we have a large number of flip-flops that are all to share the same clock-enable expression, we would have two penalties of using clock enable: 1. we have to route the clock enable signal to all of them, and 2. we have to make the flip-flops from logic which is capable of being clocked at every opportunity. We use clock-enable inputs where single or smaller numbers of flip-flops are to be enabled in unison, such as in a broadside register with clock enable. For single flip-flops, the clock enable is best considered as just one aspect of the FSM that the flop is part of.

2.9 Clock and Data Skews

Skew is when a signal that is intended to arrive at several places at the same time actually arrives at various times.

Figure 33 shows a shift register circuit with clock skew. Skew is the enemy of synchronous, finite-state machines, since in the worst scenario, the output of one flop may have changed, causing a change in the input to another flop, before that second flop has been clocked itself: this destroys the FSM behaviour.

If there is considerable delay in both the data and the clock, and these delays are random, then the shift register is not likely to work well. An example of such a system is where the shift register is actually a communications medium snaking around a number of circuit boards in a system. The delays are in the interconnecting cables. Typically, the logic at each station is not simply a D-type, but is logic which can read and change the data as it passes and the data may be a bus instead of a single wire. In this case, **for low-speeds, the solution is to invert the clock at each hop.**

Exercise: Draw a timing diagram for the inverting-the-clock solution and work out the maximum frequency of operation. Use the following figures: conductor delay random between 4 and 9 nanoseconds, flip-flop set-up time 2 nanoseconds, hold time 1 nanoseconds and propagation time 5 nanoseconds. Note: in practice you would not run such a system close to the maximum you have just calculated.

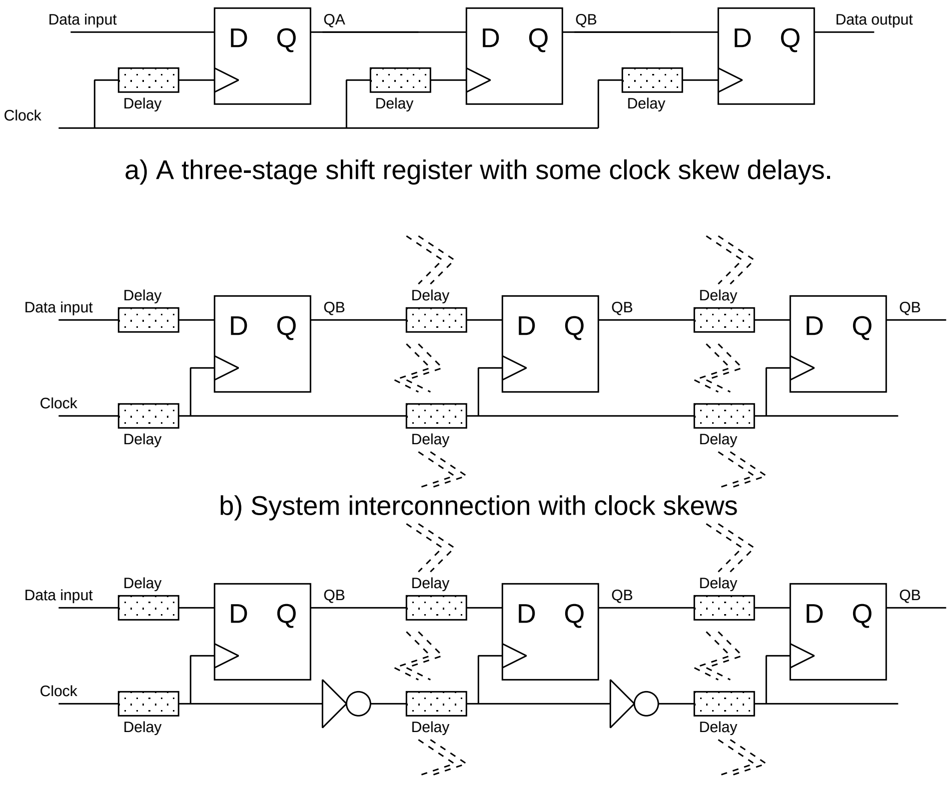


Figure 33: Synchronous systems, but with clock skew.

2.10 Crossing Asynchronous Boundaries

Figure 34 shows two clock domains with signals crossing between them. There may be signals in the reverse direction between the clock domains, but these would not demonstrate anything further that is significant.

The first thing to note is that for a single signal between the domains, there is always the possibility of metastability and failure of the whole system. However, with careful flip-flop design, a designer can reduce this to less than the probability of winning the lottery every week for a year: so this is not a real concern. Since there is more than one clock in the world, we do have to build these circuits in reality.

The main thing to note is that there will be skew between the lines and in the characteristics of the individual flip-flops. Therefore it is impossible to be sure of capturing a consistent view of them at the receiver if they are all changed at once or changed freely by the source. Therefore a protocol is required. A suitable protocol is to denote one of the signals as a *guard* or *qualifier*. Only if the guard has its active value is it implied that the data on the remaining signals is valid. The source must execute or implement a protocol that simply delays the assertion of the guard with respect to the other wires such that the other wires are sure to be seen as settled if the guard is seen to be true.

A good way to help avoid metastability is to use two flip-flops in tandem (as shown for the guard). Even if the first oscillates a while after being violated, it is most unlikely that it is still oscillating a clock cycle later when the second flip-flop is used. The additional delay in the second flip-flop will result in the guard being presented to the receiver logic later than the signal lines, thus ensuring that they are stable.

Exercise: Is it possible to use changes of state of a guard signal to indicate that the data on the remaining wires is valid? If so, sketch an example circuit and explain the benefits. What assumptions regarding the relative clock frequencies in use have you made, if any ?

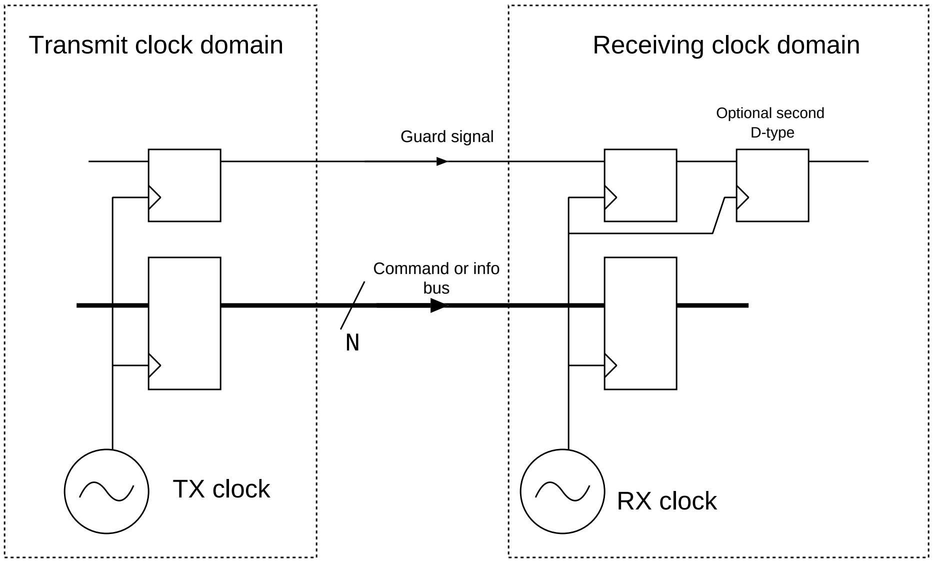


Figure 34: Use of a guard to cross an asynchronous boundary

2.11 FSM clocks derived from another FSM

Figure 35 shows a FSM whose outputs are used to clock another FSM. Examples of this are common, as in the serial to parallel convertor shown in figure 30 and the CFR two-chip decomposition shown in section 4.4. Such structures must be used carefully. The Mealy output clocking a slave FSM is probably a bad idea. The Moore output used to clock a slave is normally fine, but the logic function generating it must be free from hazards, otherwise the clock will glitch. A function of one state variable is best, and state assignment should have considered that.

The non-clock signals between the master and slave FSM also need attention. The Moore output from the slave to the master will not cause hazards, but will tend to restrict the maximum frequency of safe clocking for the system as a whole. The Moore output from the master to the slave needs careful attention to hazards, since it must never change at the same time as the slave clock signal.

Exercise (long): Consider an FSM or system which generates a set of output signals that have to be carried via an intermediate subsystem which uses a separate (asynchronous), faster clock, and then delivered to a receiving system that must be clocked with a transported version of the original clock. An example of this would be where a company has fire alarm systems on two sites which are interconnected on a high speed data link owned by a telephone company, where the link is also carrying many other services. Sketch the basics of the circuits involved.

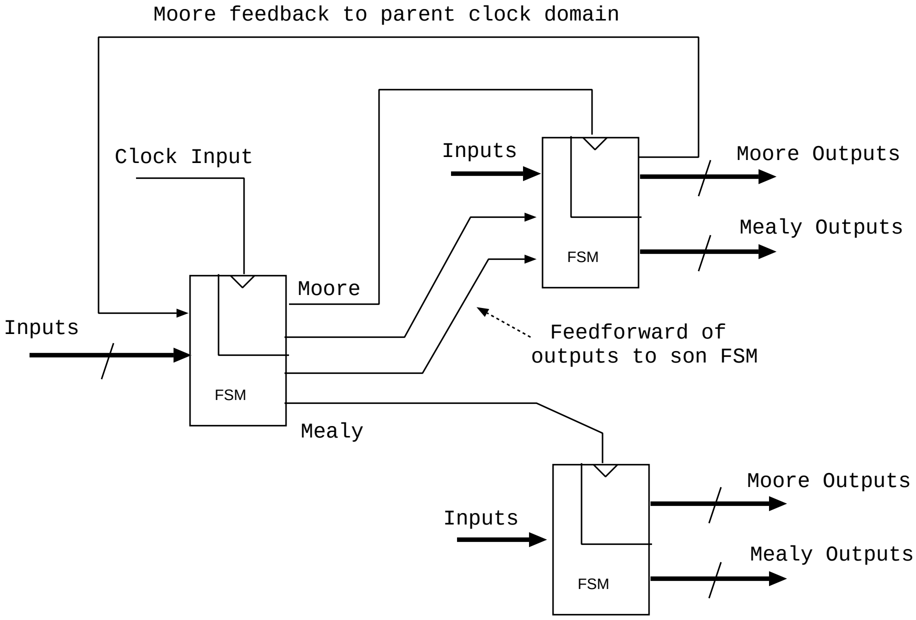


Figure 35: Example paths between FSMs with derived clocks

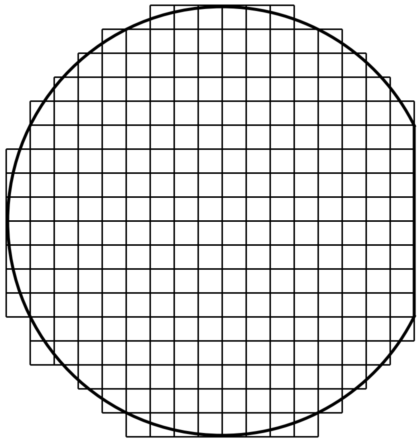


Figure 36: A wafer outline showing where it will be diced (Chips are not always square).

3 Technology for making Systems

Learners' Guide: What you should learn from this section is:

- Most designs for volume manufacture require new chips to be made.
- Field-programmable devices are used when volume does not dicate new chips or for prototypes.
- Systems are getting faster for a given power consumption, mainly due to shrinking physical feature size.
- It is sensible to implement systems with chips up to 1 centimetre on a side.
- Because of shrinking, modules which were previously complete chips now occupy just part of a new chip.

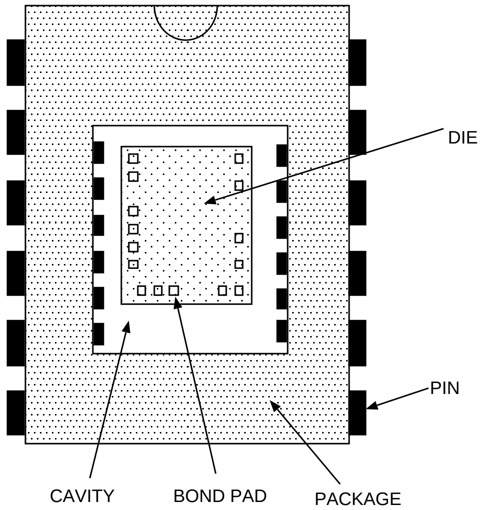


Figure 37: A chip in its package, ready for bond wires.

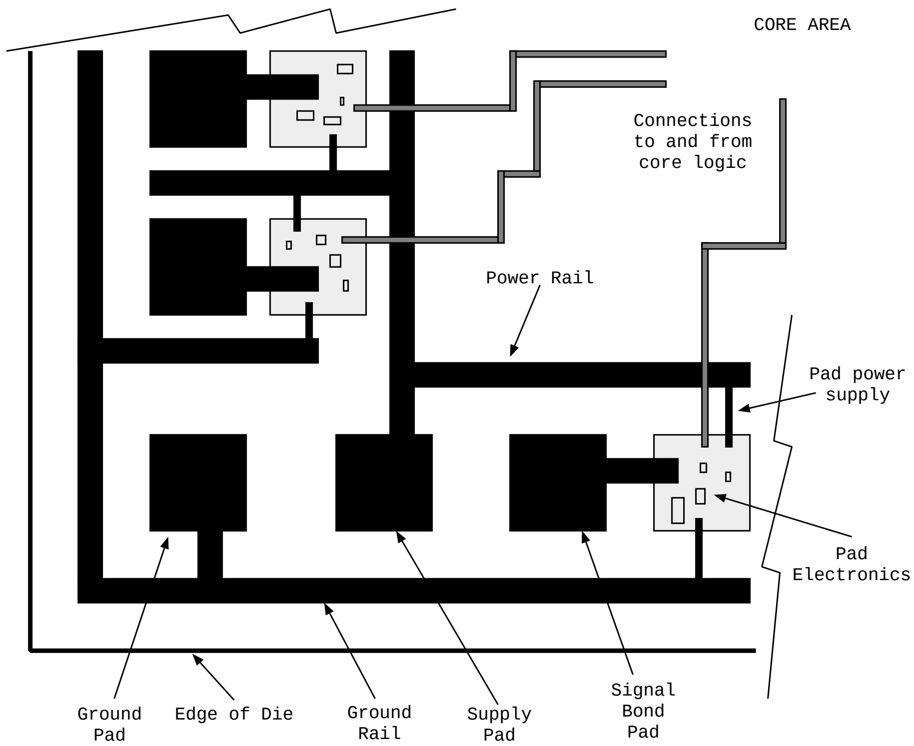


Figure 38: The corner of a typical chip showing IO and power pads.

The cost of a system depends on the cost of its parts. There are many ways to design a system and typically we want to minimise cost. Therefore, before you can design a system, you need to be familiar with a vocabulary of technologies and pre-existing parts. You also need to know how to produce new integrated circuits for your application design. Application-specific integrated circuits (ASICs) and make up a large fraction of the IC market by revenue.

3.1 Basics of ICs.

Note: this section is a briefest of introductions to VLSI.

Chips are made of Si or GaAs with layers of Al on top for signal wiring. Chips are made on a rectangualr grid on a wafer of the substrate material. Wafers are about 0.3 mm thick and from 3 to 8 inches in diameter. Most of the wafer depth is only present for handling reasons and the active portion of the chip lies in the top few microns of the wafer or is built up above the top surface.

Wafers are processed using ion implantation and photo-resistive etching at elevated temperatures in the presence of dopant and etchant gasses. Each photo-resist is first exposed to an ultraviolet or X-ray source under a mask generated by the CAD tools. Between 7 and 25 masks might be used, depending on the technology. The masks are stepped over the wafer to form the rectangular array of chips. Masked areas are left with an intact layer of resist and then remain unaffected by the next processing step, hence successive masks build up complex three dimensional structures forming the active components and conductors. Typical chips are today made using 0.25 to 0.5 micron feature sizes. The smaller a transistor the faster it operates.

A wafer contains many chips which are diced after wafer testing. The chip is then packaged to ease handling and to dissipate heat.

A typical chip is 4 to 100 square millimetres.

Wires are connected to chips at bond pads. Bond pads are about 100 microns square and 150 microns apart or half this size (linear dimension) in modern devices. The corner of a typical device is shown in Figure 38. Most chips have the shown power and ground 'rings' around the outside with the bond pads in between. These rings provide low resistance power supplies to the pad logic. The pad logic is buffer circuitry to provide high-power drive to off-chip loads and to protect the internal circuitry from external static discharges.

Figure 37 shows the chip carrier cavity containing a die. Connections are made from the conductors in the package to the bond pads around the chip's perimeter using cold-welded gold wires.

3.2 Will an ASIC be core or pad bound ?

A chip may be either:

- **Pad bound** if it has many more pins than natural perimeter, or
- **Core bound** if it has sufficient active core area to dominate.

A pad bound chip is wasteful of area since we have to use a bigger die to get the pads on, but the body of the die is not fully occupied with gates.

Generally **we end up with separate fabrication processes for each of the following subsystems:**

- memory,
- random logic,
- analogue circuits and analogue to/from digital conversion,
- high speed (e.g. clock rate greater than about 100 MHz).

Also, we must group together functions which can be implemented in the same technology, so they can most easily be put on one chip. This is hard if the functions do not otherwise have much to do with each other: we might prefer to put the two sections of logic further apart.

3.3 Chip cost versus area

The cost of a chip divides into two parts: NRE and per-device cost.

NRE — non-recurring expenditure. NRE is paid once, regardless of how many devices are made. It consists of:

- design cost (labour, computer time, management overheads),
- mask making cost.

Per device — recurring expenditure consists of:

- cost of basic silicon wafer,
- cost of processing a wafer,
- cost of testing devices,
- cost of device packages and packaging,
- cost of further testing, possibly under stressful conditions.

The per device cost is influenced by the yield — the fraction of working dice.

The fraction of wafers where at least some of the die work is the 'wafer yield' and is typically close to 100 percent for a mature fabrication process.

The fraction of die which work on a wafer (often simply the 'yield') depends on wafer impurity density and device size. Die yield goes down with chip area.

The fraction of devices which pass wafer probe (i.e. before the wafer is diced) and fail post packaging tests is very low. However, full testing of analog sections or other lengthy operations are typically skipped at the wafer probe stage.

3.4 Example of CMOS die cost

A six inch diameter wafer has area $(3.14r^2) = 18000$ mm^2.

A chip has area A, which can be anything between 2 to 200 mm^2 (including scoring lines).

Dies per wafer is $18000/A$

Processed wafer cost might be 5000 pounds. (Actually it's probably about one third this cost to the silicon foundry, but there are overheads and profit to add on.)

Probability of working = wafer yield $\times$ die yield (assume wafer yield is 1.0 or else included in the wafer cost).

Assume 99.5 percent of square millimetres are defect free. Die yield is then

$$P(\text{All A squares work}) = 0.995^A$$

cost of working dice is

$$\frac{5000}{\frac{18000}{A}0.995^A} \text{ pounds each.}$$

Here is a program to estimate the cost of a working die given a six inch wafer with a processing cost of 5000 pounds and a probability of a square millimetre being defect free of 99.55 percent. Its output is shown in table 1.

```
#include <math.h>
display(double area)
{
   double dies = 18000.0 / area;
   double yield = dies * pow(0.995,  area);
   double cost = 5000.0 / yield;
   printf("%.0f  %.0f %.0f %.2f\n", area, dies, yield, cost);
```

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

Table 1: Output from the die cost program.

```
}
int main()
 {
    double area = 2.0; /* area in square mm */
    while (area < 500.0)
     {
       display(area);
       area = trunc(area * 1.5);
     }
  }
```

Large dies sometimes use redundant design such that one, two or three faults can be tolerated. Laser or ion beam trimming is used to patch out faulty sections (of say a RAM array) and substitute previously redundant sections.

3.5 Classes of Integrated Circuits

Figure 39 presents a taxonomy of chip design approaches. The top-level division is between standard parts, ASICs and field-programmable parts. **Where a standard part is not available, the choice between full-custom and semi-custom and field-programmable approaches has to be made, depending on performance and cost requirements.**

3.5.1 Standard Parts

A *standard part* is essentially any chip that a chip manufacturer is prepared to sell to someone else along with a datasheet. The design may actually previously have been an ASIC for a specific customer which is now on general release. However, most standard parts are general purpose logic, memory and microprocessor devices. These are normally full custom designs designed in-house by the chip manufacturer to make the most of in-house fabrication line, perhaps using optimisations not made available to others who use the line as a foundry. Other standard parts include graphics controllers, LAN controllers, bus interface devices, and miscellaneous useful chips.

3.5.2 Masked ASICs.

A masked ASIC (application specific integrated circuit) is a device manufactured for a customer involving a set of masks where at least some of the masks are used only for that device. These devices include full-custom and semi-custom ASICs.

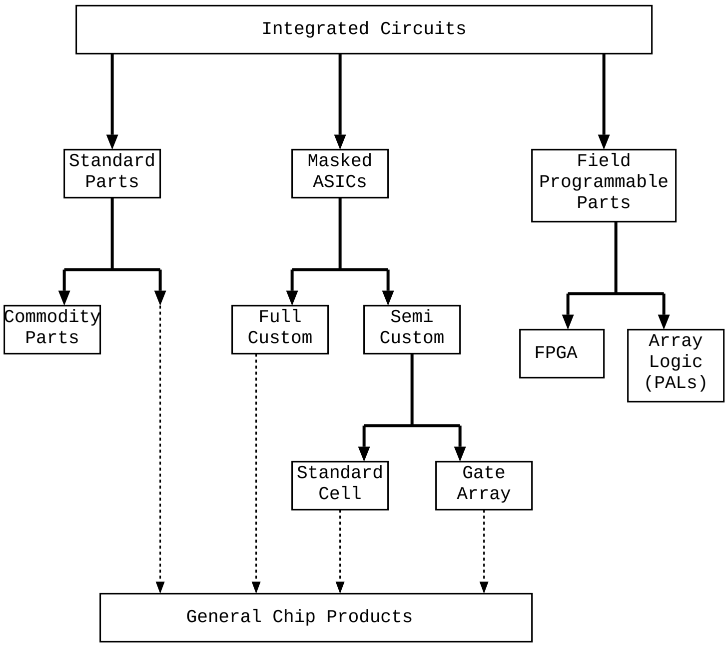


Figure 39: A taxonomy of integrated circuits.

A full-custom chip (or part of a chip) has had detailed manual design effort expended on its circuits and the position of each transistor and section of interconnect. This allows an optimum of speed and density and power consumption.

Full-custom design is used for devices which will be produced in very large quantities: e.g. millions of parts where the design cost is justified. Full-custom design is also used when required for performance reasons. Microprocessors, memories and digital signal processing devices are primary users of full-custom design.

In semi-custom design, each cell has a fixed design and is repeated each time it is used, both within a chip and across many devices which have used the library. This simplifies design, but drive power of the cell is not optimised for each instance.

Semi-custom is achieved using a library of logic cells and is used for general-purpose VLSI design. Figure 49 shows a cell from the data book for a standard cell library. This device has twice the 'normal' drive power, which indicates one of the compromises implicit in standard cell over full-custom, which is that the size (driving power) of transistors used in a cell is not tuned on a per-instance basis.

There are two types of semi-custom devices:

- standard cell (for high volume)
- gate array (for volume ¡ 5K devices).

In standard cell designs, cells from the library can freely be placed anywhere on the device and the number of IO pads and the size of the die can be freely chosen. Clearly this requires that all of the masks used for a device are unique to that device and cannot be used again. (Mask making is one of the largest costs in chip design).

In gate array designs, the silicon vendor offers a range of chip sizes. Each size of chip has a fixed layout and the location of each transistor, resistor and IO pad is common to every design that uses that size. A gate array is configured for a particular design by wiring up the transistors (and other components) in the desired way. Many transistors will be unused. The wiring up is, as usual, done with the top two or three layers of metal wiring. Therefore only two or three masks need be made to make a new design.

Silicon vendors will typically have stocks of wafers which have had the bottom 15 to 20 process steps made and are only awaiting final metalisation to be turned into usable devices. Hence design time is low. Risk is also low since the finished chip has only a small NRE cost and gate array technology is intrinsically conservative and hence reliable.

The disadvantage of gate arrays is their intrinsic low density of active silicon.

Standard cell designs use a set of well-proven logic cells on the chip, much in the way that previous generations of standard logic have been used as board-level products, such as Texas Instruments' System 74.

A variation on the semi-custom approach is to include full-custom macrocells such as processor cores in fixed positions on the wafer.

Exercise: Ignoring the title, determine from the data sheet whether the cell in figure 49 is for standard cell or gate array use ? What other information about the cell is needed to prepare an audit of resources used in a design which has used this cell ? The audit refered to is typically a report generated by the CAD tools which gives summary information.

3.5.3 Field-programmable logic

About 25 to 40 percent of chip sale revenue now comes from field programmable logic devices. These are chips which can be programmed electronicly on the user's site to provide the desired function. The Xilinx FPGA parts used in the Part 1B E+A classes are one of the most important examples of field-programmable logic.

Field-programmable devices may be volatile (need programming every time after power up), reprogrammable or one-time programable. This relates to how the programming information is stored inside the devices, which can be in RAM cells or in any of the ways mentioned for ROM devices in section 1.2.

3.5.4 FPGAs

An FPGA (field-programmable gate array) consists of an array of configurable logic blocks (CLBs), as shown in Figure 40. Not shown is that the device also contains a good deal of hidden logic used just for programming it. Some pins are also dedicated to programming. Such FPGA devices have been popular since about 1990.

Each CLB (Figure 41) typically contains one or two flip-flops, and has a few (five shown) general purpose inputs, some special purpose inputs (only a clock is shown) and two outputs. The illustrated CLB is of the look-up table type, where the logic inputs index a small section of preconfigured RAM memory which implements the desired logic function. For five inputs and one output, a 32 by 1 SRAM is needed. Some FPGA families now give the designer write access to this SRAM, thereby greatly increasing the amount of storage available to the designer. However, it is still an expensive way to buy memory.

All CLBs within a FPGA generally have the same structure, but FPGAs are available with lower and higher functionality CLBs. The best size of CLB is not yet clear. Modern designs of FPGA have a heirarchy of CLB internconnection patterns, giving CLB clusters within clusters.

An FPGA is very like a mask-programmed gate array to use. The design flow and CAD tools are virtually identical. The expenditure before the designer has the first device in her hands might be 1000 times lower. The cost of further devices is at least 10 times higher than mask-programmed devices, owing to the programming cost and wasted die area devoted to the programming activities. Clearly there is a crossover production volume point!

FPGAs also tend to be quite slow, owing to the signals passing through hidden logic used only for configuration.

Often a designer or company will build prototypes and early production units using FPGAs and then use a drop-in mask-programmed equivalent once the design is mature and sales volumes pick up.

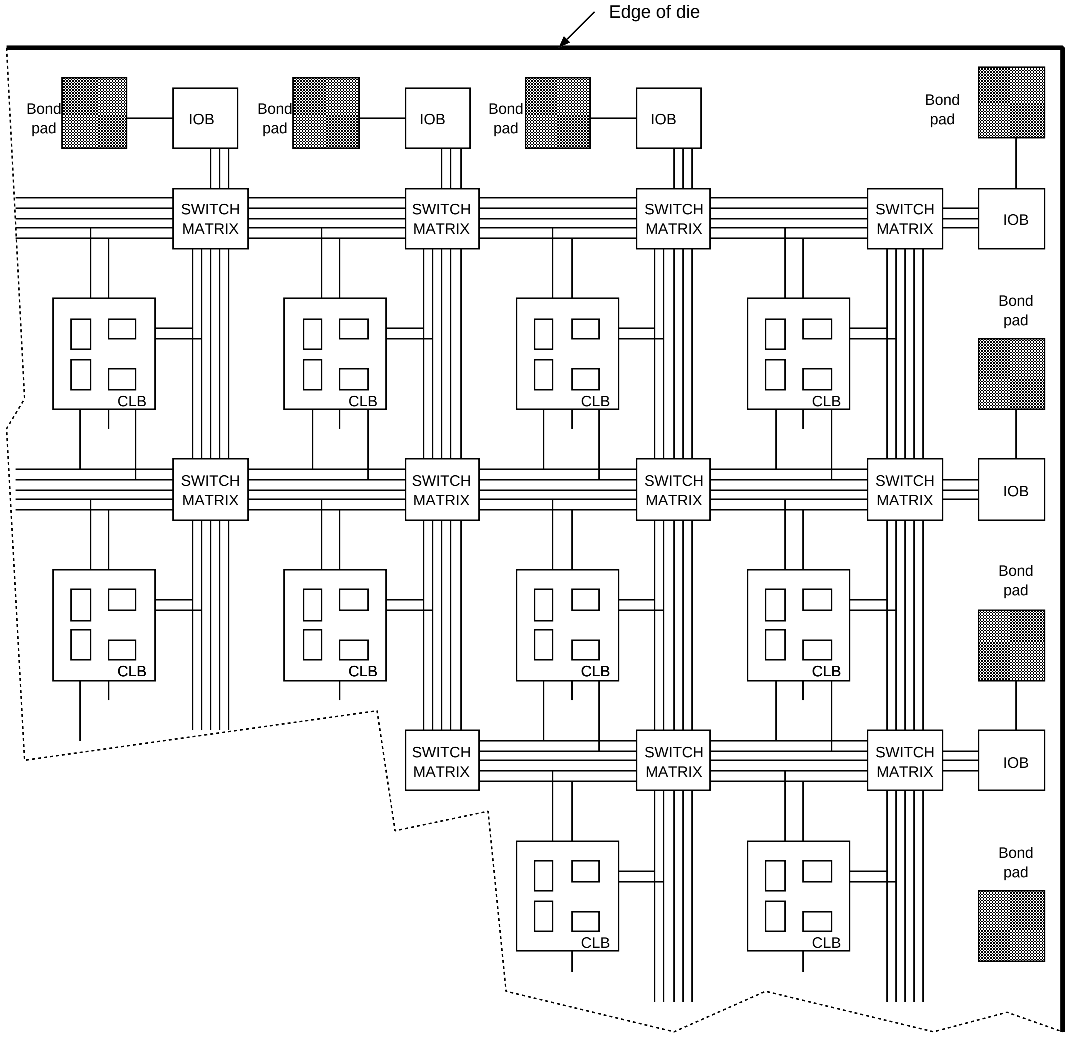


Figure 40: Field-programmable gate array structure, showing IO blocks around the edge, interconnection matrix blocks and configurable logic blocks.

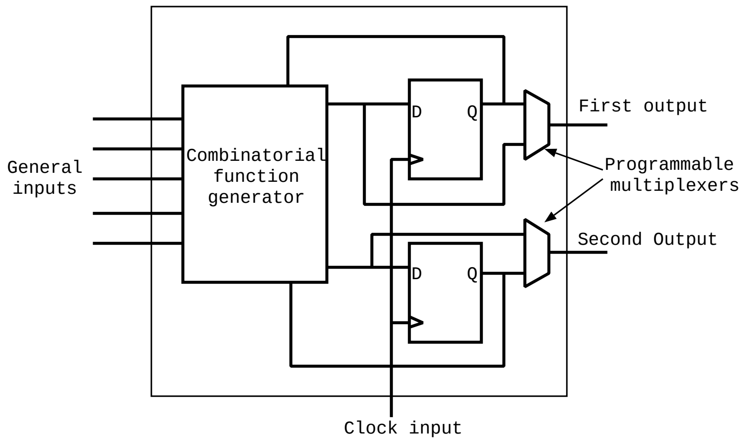


Figure 41: A configurable logic block for a look-up-table based FPGA.

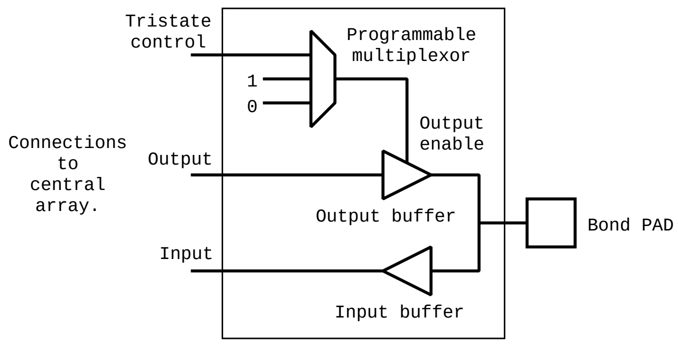


Figure 42: A simple IO block for an FPGA.

Exercise: Attempt to sketch the logic needed inside one of the programmable switch matrix boxes. You should find this is a vast amount of logic and therefore understand why FPGAs are so expensive for their functionality. (If you know how to use pass transistors, this will help your design).

3.5.5 PALs

A PAL is programmable array logic device. Figure 43 shows a typical device. Such devices have been popular since about 1985. The illustrated device has 8 product terms per logic function, and so can support functions of medium complexity. Such devices are very widely used and can feature high speed operation with clock rates of above 100 MHz. They are really just highly structured gate arrays. Every logic function must be multiplied out into sum-of-products form and hence is achieved in just two gate delays.

Programmable *macrocells* (Figure 44) enable the output functions to be either registered or combinatorial. Small devices (e.g. with up to 10 macrocells) offer one clock input; larger devices with up to about 100 macrocells are also available, and generally offer several clock options. Often some macrocells are not actually associated with a pin, providing a so called *buried state* flip-flop.

Figure 45 shows part of a PAL description for the PAL showin in figure 43, as entered by a designer in a typical PAL language, and part of the fuse map that would be generated by the PAL compiler. Each product line has seven groups of four fuses and produces the logical AND of all of the signals with intact fuses. An 'x' denotes an intact fuse and all of the fuses are left intact on an unused product lines in order to prevent the line ever generating a logical one (a gets ANDed with abar etc.). The fuse map is loaded into a programming machine (in a file format known as JEDEC), an unused PAL is placed in the machine's socket and the machine programs the fuses in the PAL accordingly.

Exercise: How large a binary counter can the illustrated device implement ? (Hint: are there sufficient product terms for the most-significant bit ?)

PALs achieve their speed by being highly structured. Their applicability is restricted to small finite state machines and other *glue logic* applications.

3.6 Delay-power product

Aside from the selection of design methodologies just discussed, the designer must choose the basic circuit technology to use:

- silicon or GaAs
- bipolar or unipolar
- saturating or non-saturating.

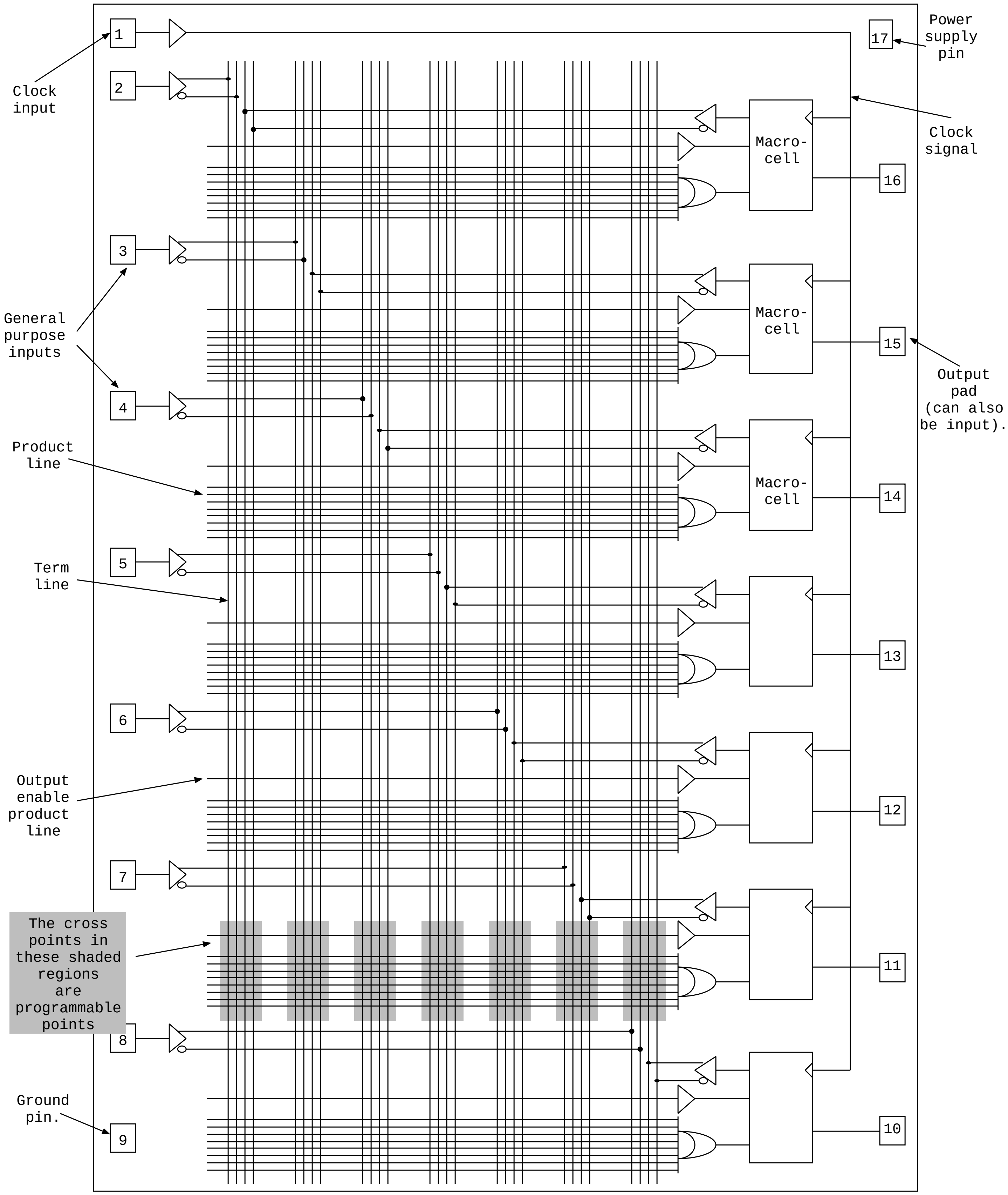


Figure 43: A typical PAL with 7 inputs and 7 I/Os.

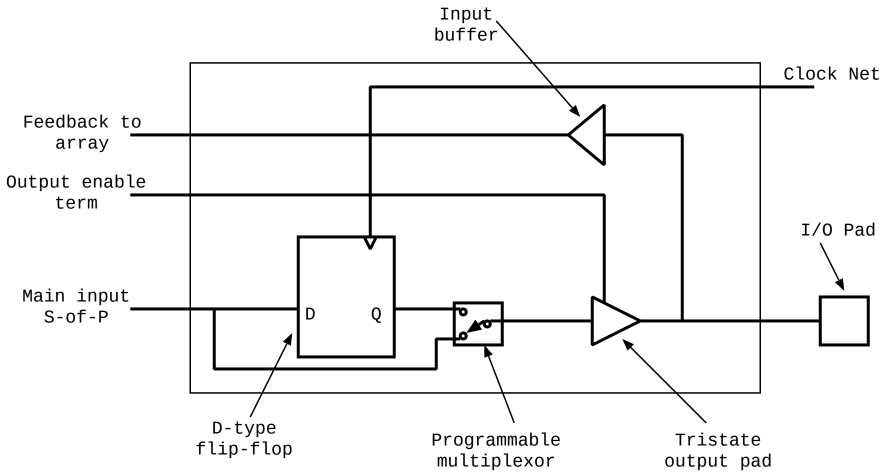


Figure 44: Contents of the PAL macrocell.

```
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)
```

Figure 45: Example programming of a PAL showing only fuses for the top macrocell.

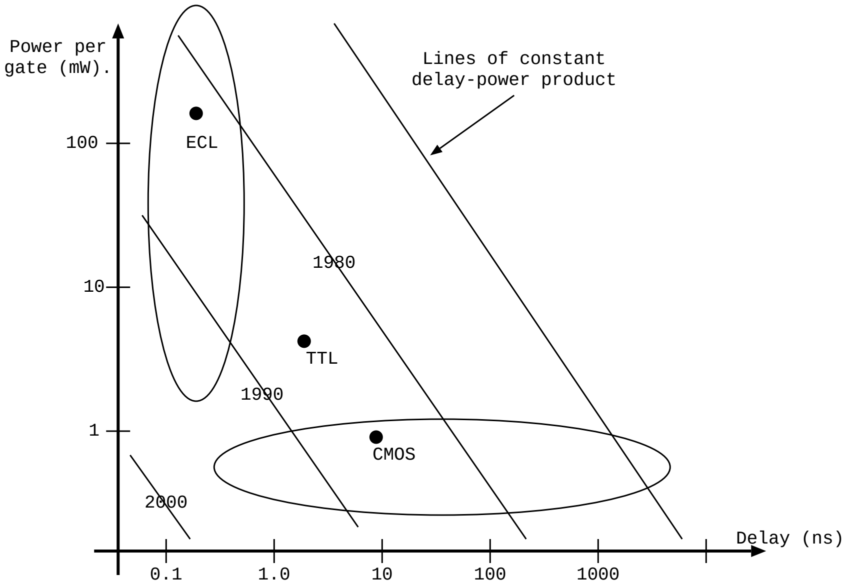


Figure 46: Delay-power style of technology comparison chart.

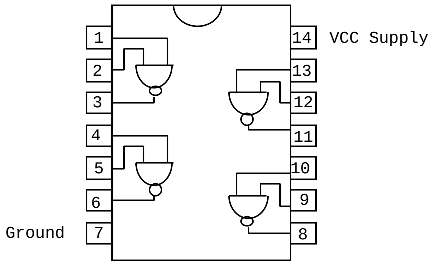


Figure 47: A SSI quad NAND gate in the 7400 pinout.

As the size of transistors on chips is reduced with ever improving fabrication equipment, the number of devices per unit area increases and the delay-power product (power consumption multiplied by propagation delay) decreases.

We here consider the main three circuit technologies of the 70's and 80's using a quad AND gate chip SSI chip for example (shown generically in Figure 47). In the 90's, CMOS is dominating most application areas.

technology	device	propagation delay (ns)	power (mW)	product (pJ)
CMOS	74hc00	7 ns	1 mW	7
TTL	74f00	3.4 ns	5 mW	17
ECL	sp92701	0.8 ns	200 mW	160

The CMOS gate has the property that it consumes virtually no power when outputs are not changing.

The ECL gate is an old technology, but it is still the fastest.

Gates in the core of an IC tend to be one order better in speed or power when compared with the SSI gate used in the example (they have reduced drive requirement).

All of these aspects have to be considered by the system designer and it is therefore a skilled job.

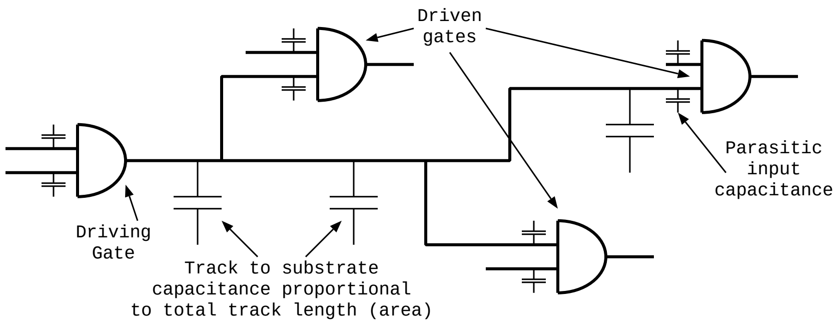


Figure 48: Logic net with tracking and input load capacitances.

3.7 Fanout and delay estimations.

Figure 48 shows a typical net, driven by a single source. To change the voltage on the net, the source must overcome the stray capacitance and input loads. The fanout of a gate is the number of devices that its output feeds. The term *fanout* is also sometimes used for the maximum number of inputs to other gates a given gate is allowed to feed, and forms part of the design rules for the technology.

The speed of the output stage of a gate, in terms of its propagation delay, decreases with output load. Normally, the dominant aspect of output load is capacitance, and this is the sum of:

1. the capacitance proportional to the area of the output conductor
2. the sum of the input capacitances of the devices fed.

To estimate the delay from the input to a gate, through the internal electronics of a gate, through its output structure and down the conductor to the input of the next gate, we must add three things:

1. the internal delay of the gate, termed the intrinsic delay
2. the reduction in speed of the output stage, owing to the fanout/loading, termed the derating delay,
3. the propagation delay down the conductor.

The propagation delay down a conductor obeys standard transmission line formula and depends on the distributed capacitance, inductance and resistance of the conductor material and adjacent insulators. For circuit board traces, resistance can be neglected and the delay is just the *speed of light* in the circuit board material: about 7 inches per nanosecond, or 200 metres per microsecond.

On a chip, the speed of light can be neglected because chips are physically small, but the resistance of the aluminum conductors is sufficiently large to have an effect for critical applications, such as master clock signals.[1]

The upshot of this is that on a chip, we can mostly neglect the third aspect of delay, whereas on a circuit board, we need to model certain critical conductors as components in themseleves. These have a simple delay model, whose value can be set by post routing back annotation.

The first two aspects of delay may easily be absorbed into the model for a component. We can use the following formula

$$\text{device delay} = (\text{intrinsic delay}) + (\text{output load} \times \text{derating factor}).$$

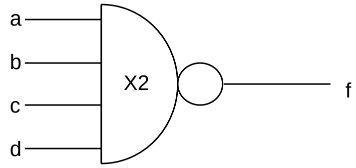
[1]Older technologies used polysilicon interconnections which had significant resistance and so the different gates connected to a polysilicon net would experience different arrival times of a signal at their inputs.

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

Figure 49: An example cell from a manufacturer's cell library.

Technology	Maximum clock speed	Maximum gate count	Maximum pin count
GaAs bipolar	50 GHz	50	5
GaAs fet	3 GHz	300	100
Si ECL	3 GHz	5000	300
Si CMOS	400 MHz	1 E6	500
Within Si CMOS			
Full-custom	400 MHz	1 E6	500
Standard cell	200 MHz	40000	500
Gate array	100 MHz	20000	500
PAL	75 MHz	500	68
FPGA	35 MHz	4000	200

Table 2: Approximate limits on contemporary technology (1977).

The intrinsic speed of a device is given in its data sheet, as is the derating factor and the loading factor of its inputs. Typical values for a standard cell array are shown in Figure 49. The output load is the sum of a track dependent part and the fanout dependent part.

The track dependent part is a library constant times the track area.

The load dependent part is the sum of the input loads of all of the devices being fed.

Exercise: How true is the above model of signal delay when we use field-programmable gate arrays with high effective track resistance ? The conductors on FPGAs consist of many sections of real metalic conductor interconnected by the user-programmed connection points. We then have considerable resistances at points along each conductor's path. How would you estimate the various delays for the staggered arrival of a signal on each part of the net ?

3.8 Comparative view of digital logic technologies

Table 2 gives some randomish numbers for what can be achieved with today's chips. These numbers are only accurate to about half an order of magnitude. Technology is always advancing!

For CMOS technology, the number of transistors on a chip and the clock rate both seem to double every three years.

4 Hardware, Software and Design Partition.

Learners' Guide: What you should learn from this section is:

- The functions of a system can be expressed in a programming language or similar form and this can be compiled fully to hardware or left partly as software
- Chosing what to do in hardware and what to do in software is a key decision. Hardware gives speed (throughput) but software supports complexity.
- Partitioning of logic over chips or processors is motivated by speed, technology and module reuse.

A hardware system can be designed using only hardware or hardware with an embedded processor and software in ROM. Perhaps several processors are to be used. A design also must be partitioned into standard components and application specific integrated circuits (ASICs). The choice of which parts of the design are realised in what technology is the design partition problem.

4.1 Embedded Systems

A processor which is permanently connected to a single ROM which contains all of the software that the processor will ever execute is called an *embedded processor*. Here we consider when to use an embedded processor, and then, when using one, the designer has to chose whether to design a new one or use an existing one.

The main difference between a hardware solution and a software solution is the degree of parallelism. Processors typically execute one instruction at a time, reusing the same hardware components again and again. Hardware solutions tend to have dedicated circuits for each function. If most of the hardware is likely to be idle for most of the time, a processor is prefered, but if a processor cannot achieve the throughput required, increased parallelism using hardware is prefered.

Complex functions normally require a processor, but CAD tools are evolving, allowing complex functions to be expressed alogrithmicly but implemented in logic gates.

High speed processing normally requires dedicated hardware. For instance, consider the error correction performed by a CD player to overcome dirt and scratches. When CD players first came out, this error correction was done with dedicated hardware, but today, microprocessors have increased in speed, and so the function can be done using the processor which is already there to provide other complex functions (e.g. track skip). However, on the latest, 24-40 speed CD ROM drives for computers, the error correction must be done 24-40 times faster, and so dedicated hardware is re-introduced.

Processors can be placed on an ASIC.

Processors give flexibility: if the design is likely to be changed in minor ways to form new models, or to provide field upgrades, then using a processor and providing a new software release is a good technique.

Standard processor chips can be very cheap for a given performance. This is because a very great deal of effort is put into their design and they are sold in large quantites, thereby reducing price. Part of the cost of any product is in testing it. A standard processor not only comes with its own test qualification programme, it is able to execute software to help test the rest of the system.

The decision to design and use a custom processor for an application should not be taken lightly. It can be useful, however, when the application is very simple or highly specialised. Examples are signal processing, where a particular algortihm needs to be executed at high speed to track a missile or assist computer vision.

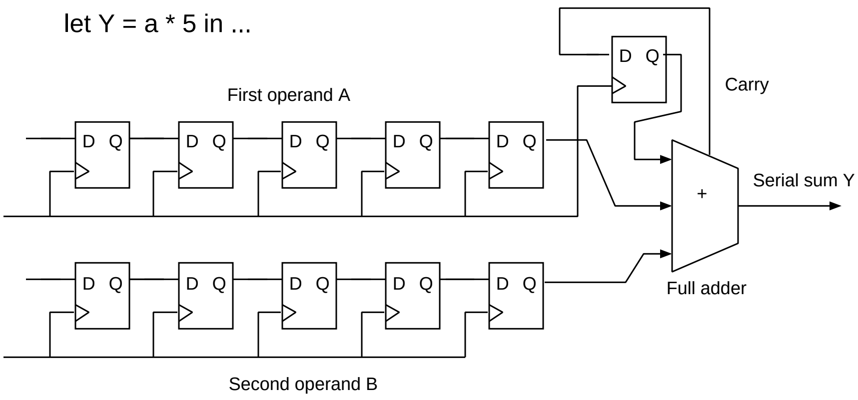


Figure 50: Addition of two integers serially, l.s.b. first

4.2 Logic Synthesis and Collapsing

If one considers an embedded processor connected to a ROM, it may be viewed as one large FSM. Since for any given piece of software, the ROM is unlikely to be full and there are likely to be resources in the processor that are not used by that software, the application of a good quality logic minimiser to the system, while it is in the design database, could trim it greatly. In most real designs, this will not be helpful: for instance, **the advantages of full-custom applied to the processor core will be lost.** Infact, the minimisation function may be too complex for most algorithms to tackle on today's computers.

However, there are cases that are practical. For instance, many digitial signal processing applications do use this approach. A processing algorithm typically consists of multiple stages with names such as pre-emphasis, equalisation, coefficient adaptation, fft, deconvolution and reconstruction. Each of these steps normally has to be done as fast as possible and so parallelism through multiple instances of ALU and register hardware is needed. However, there can be data dependencies in that one step cannot be done until the previous is completed, or in that two steps take the same amount of processing as one other step. In these cases, **we desire to reuse the same hardware for multiple steps.** The Cathederal DSP compiler is the most famous tool for helping design such circuits. It performs folding of the algorithm to reuse hardware from one step in the next.

4.2.1 Hardwired, bit-serial algorithms.

Bit-serial arithmetic is also a key approach in custom signal processing. The adder shown in figure 50 requires only about 10 gates to add any number of bits. It operartes l.s.b. first so that the carry from one digit is available for the next digit. The clock speed for the calculation does not need to be reduced if longer words must be consumed, but the pipeline delay does go up. Multiplication is also easy in bit serial form, as shown for multiplication by a constant in figure 51. Consider the same functions implemented in naive, broadside implementations.

A battery operated walkman CD ROM player may implement a great deal of DSP. The printing on the unit may advertise '8 times oversampling D-to-A bitstream convertor and MASH bit mapping'. The processing power required for this is as much a 486 style processor can produce, **but through the use of bit serial arithmetic and hard-wired algorithms, it takes very little power or silicon area.**

The problem with bit-serial processing for the human designer is that it is very tricky to work out where individual bits are at any one time, and the optimisations that are possible cannot readily be seen by the human. Therefore, good quality CAD tools are vital in the design of such systems.

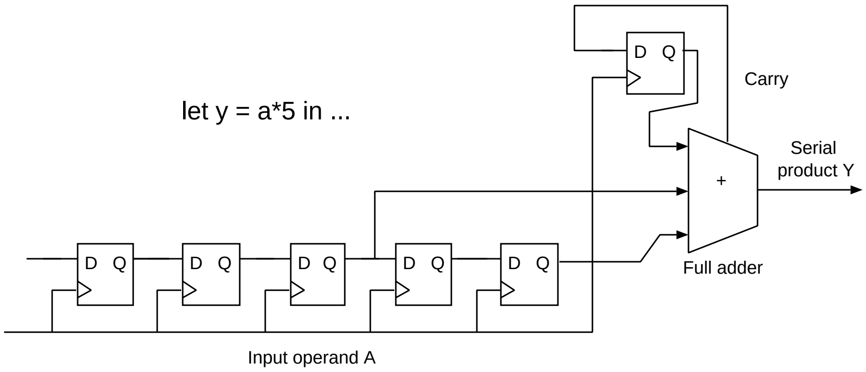


Figure 51: Bit-serial multiplication of an integer by a hardwired constant

4.3 ASICs

The cost of developing an ASIC has to be compared with the cost of using an existing part. The existing part may not perform the required function exactly, requiring either a design specification change, if possible, and/or some additional *glue logic* to adapt the part to the application.

More than one ASIC may be needed under any of the following conditions:

- application specific functions are physically distant
- application specific functions require different technologies
- application specific functions are just too big for one ASIC
- it is desired to split the cost and risk or reuse part of the system later on.

Factors to consider on a per chip basis:

- pad count limitation (pad density limit of 15 per mm)
- power consmption limitation (powers above 5 Watts need special attention)
- gate count limitation (above 100 kgates is big for CMOS)
- speed of operation — influences choice of technology
- will it be core or pad bound ?
- special considerations :
  - special static or dynamic RAM needs
  - mixed with analogue parts ?
  - high power handling outputs for load control: e.g. motors.
- availability of a developed module for future reuse

Power dissipation is generally proportional to:

Power $\propto$ (clock speed) $\times$ (number of gates) $\times$ (supply voltage squared).

Various technologies can be positioned in the speed-power plane, as shown in Figure 46 and as time goes by, things get better.

4.4 Partitioning example: The Cambridge Fast Ring two chip set.

Two devices were developed for the CFR local-area network, illustrating the almost classical design partition required in high-speed networking. They were never given grander names than the *ECL* chip and the *CMOS* chip. The block diagram for an adaptor card is shown in Figure 52.

The ECL chip clocks at 100 MHz and contains the minimal amount of logic that needs to clock at the full network clock rate. Its functions are:

- implement serial transmission modulator and demodulator
- convert from 1 bit wide to 8 bits wide and the other way around
- perform reception byte alignment (when instructed by logic in the CMOS chip).

Other features:

- ECL logic can support analogue line receivers at low additional cost so can receive the incoming signal directly on to the chip.
- ECL logic has high output power if required (1 volt into 25 ohms) and so can drive outgoing twisted pair lines directly.

The CMOS chip clocks at one-eigth the rate and handles the complex logic functions:

- CRC generation
- full/empty bit protocol
- minipacket storage in on-chip RAM
- host processor interface
- ring monitoring and maintenance functions.

The ECL chip has at least 50 times the power consumption of the CMOS chip. The CMOS chip has more than 50 times the gates of the ECL chip.

Two standard parts are used to augment the CFR set: the DRAM chip incorporates a dense memory array which could not have been achieved for anywhere near the same cost onboard the CMOS chip and the VCO (Voltage Controlled Oscillator) device used for clock recovery was left off the ECL chip since it was a difficult-to-design analogue component where the risk of having it on the chip was not desired.

PALs are used to 'glue' the network interface itself to a particular host system bus. Only the glue logic needs to be redesigned when a new machine is to be fitted with the chipset. PALs have a short design turn-around time since they are field-programmable.

4.5 Partitioning example: An external PC Modem.

Figure 53 shows the block diagram of a typical modem. The illustrated device is an external modem, meaning that it sits in a box beside the computer and has an RS-232 serial connection to the computer. It also requires its own power supply.

The device contains a few analog components which behave broadly like a standard telephone, but most of it is digital. A relay is used to connect the device to the line and its contacts mirror the 'off-hook' switch which is part of every telephone. It connects a transformer across the line. The relay and transformer provide isolation of the computer ground signal from the line voltages. Similarly the ringing detector often uses a optocoupler to provide isolation. *Clearly, these analog aspects of the design are particular to a modem and are designed by a telephone expert.*

Modems from the 1960's implemented everything in analog circuitry since microprocessors were not then possible. Today, two microprocessors are often used, but as processing power increases,

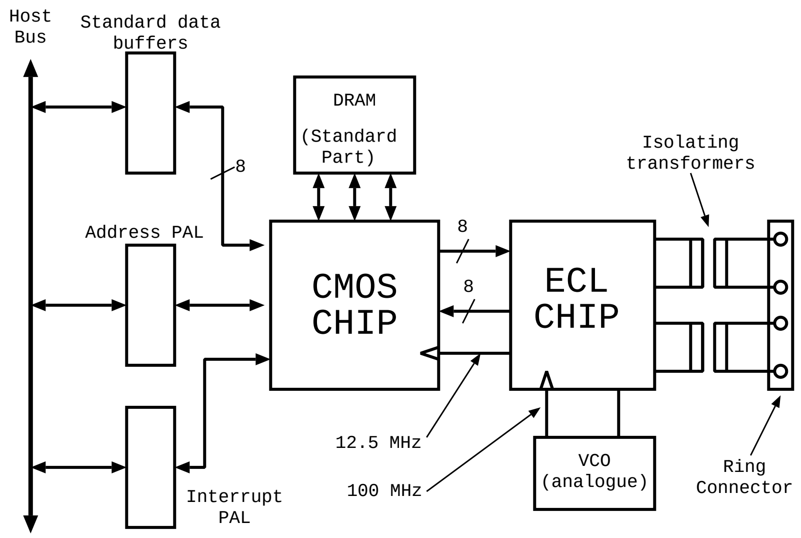


Figure 52: Example of a design partition — the adaptor card for the Cambridge Fast Ring.

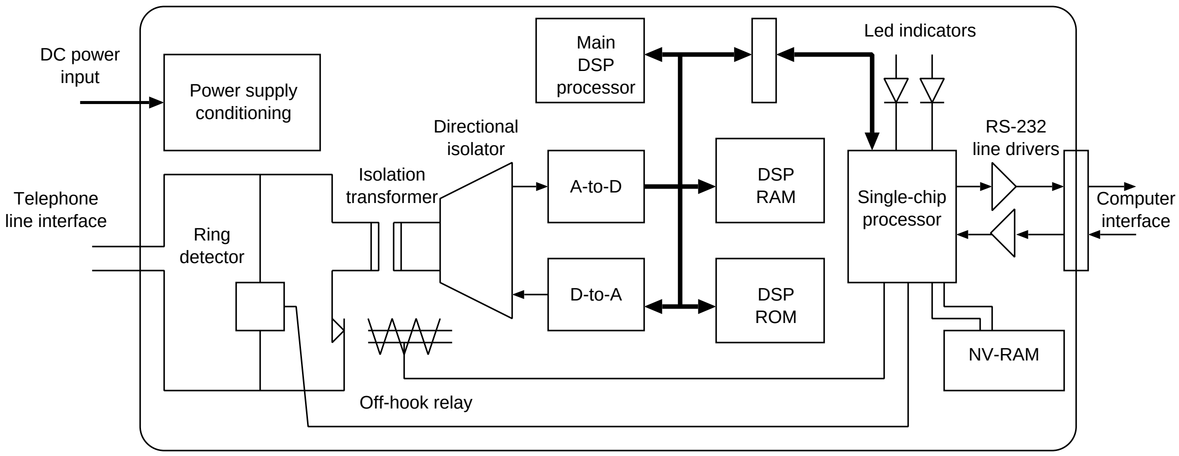


Figure 53: Example of a design partition — an external modem.

this can be reduced to one (or sometimes even none, if the main processor of a PC provides the functions needed).

The reason for two processors are interesting and will be discussed in lectures.

Note that the non-volatile RAM requires a special manufacturing processing step and so in not included as a resource on board the singe chip processor. Similarly, the RS-232 drivers need to handle voltages of +/- 12 volts and so these cannot be included on chip without increasing the cost of the rest of the chip by using a fabrication proceess which can handle these voltages. The NV-RAM is used to store the owner's settings, such as whether to answer an incoming call and what baud rate to attempt a first connection, etc..

4.6 Hardware-Software Codesign

Codesign is the name for an approach to system design where both the hardware and software components are entered in a common language and the selection of target architecture is specified in other ways, including automatically. New languages for codesign are a research area. Existing languages like C and Java can be used.

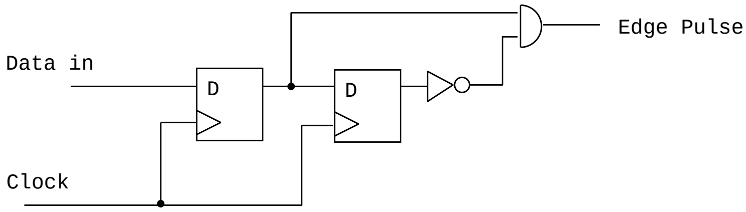


Figure 54: A simple edge detector

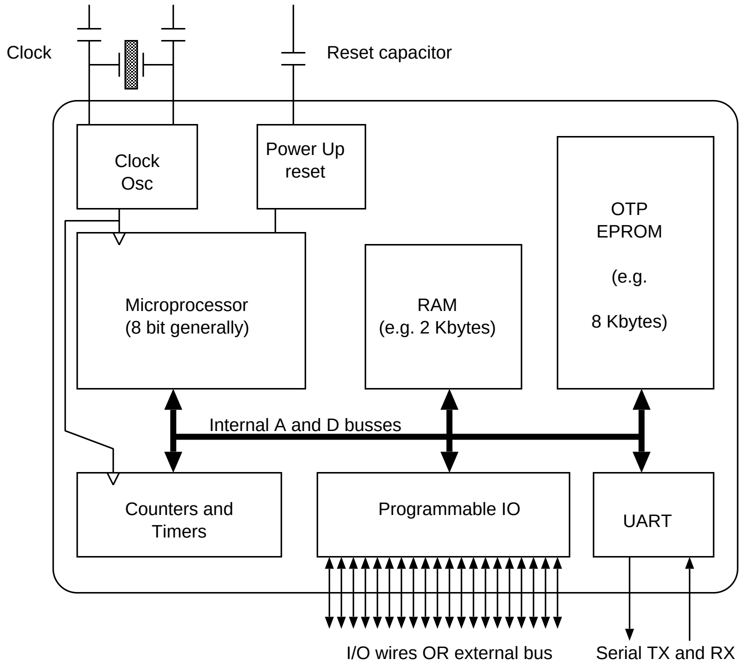


Figure 55: A typical single-chip microcomputer.

5 Example Circuit Structures.

Learners' Guide: What you should learn from this section is:

- Several further illustrations of design approaches.
- Consider the complexity of providing a user interface to itesm of hardware.
- A microprocessor often helps with control and the design of the user interface even if it is not really needed for the actual operation of a device.
- Whether to use custom integrated circuits depends on selling volume and availability of standard parts.

This section introduces some further, fundamental building blocks and includes examples of real products. The important points to note are the reasons for the design partition.

5.1 Single Chip Processor

Figure 55 shows the block diagram of a typical single-chip microcomputer. Such devices are available in 24, 40 or 80 pin packages and can cost only one or two pounds. The device contains the whole of a computer, requiring externally only a power supply and some clock and reset components. A number of programmable I/O pins are provided, but generally another basic mode

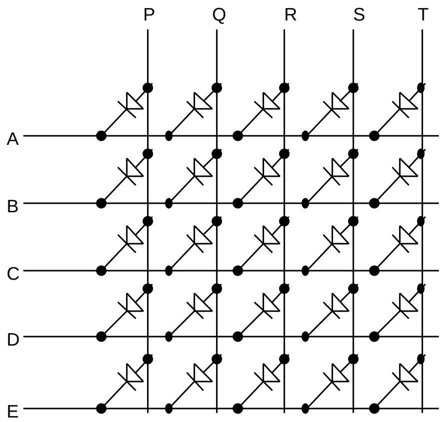


Figure 56: LEDs wired in a matrix to reduce external pin count

of operation is provided where 24 or so of them turn into the processor address and data busses for connection of external devices, such as further RAM or EPROM.

Programming is performed in all of the ways available for ROMs. Generally, uses will use windowed EPROM devices for development and prototypes and ship product with OTP devices.

A Universal asynchronous recevier and transmitter (UART) is normally always provided to help connect to RS-232 serial interfaces.

These devices are found everywhere today, from keyboards, clock radios, mice, telephones, car window winders, 'computer controlled' cassette recorders, cell phones and modems.

5.2 Scan multiplexing.

When a large number of leds or switches need to connected, as in a display or keyboard, the number of connections can normally reduced by connecting them in a matrix. Figure 56 shows that 25 LEDs can be connected to just ten signals.

The matrix gives a *scan-multiplexed* display or keyboard. In the display, one vertical column line can be driven to logic one at a time and a zero placed on the horizontal lines that should be illuminated in that column. A circuit to repeatedly read out the contents of a RAM and display it is shown in figure 58. Clearly, the desired LEDs are not all on at once, but by scanning the display faster than the human eye can detect flashes (about 50 Hz) and by using sufficiently large currents, so that the elements are brighter than would otherwise be required, this is overcome. The current is set by the value of a series current limiting resistor that is not shown.

For a scan-multiplexed keyboard, the switches take the place of the LEDs. Push-to-make, normally open switches must be used and the user should not press two at once. The scanning circuit must take one row line low in turn. Pull-up resistors keep the column lines at logic one unless a switch to a low row line is pressed.

Figure 57 shows the full PCB circuit of a typical infra-red remote controller. The PCB circuit of a toy electric organ would be identical, but with the IR diode replaced with a loudspeaker: clearly the chip would be different. A pocket calculator also has roughly the same circuit, except there is also a display.

Exercise (long): Sketch the full circuit of the chip in an infra-red remote control handset, as used for TVs and VCRs etc.. The device should have about 50 push keys. Output is through a pair of infra-red transmitting diodes which go on or off together. The design must include a ROM which can be programmed at chip manufacture with the desired sequence of pulses needed. The ROM must contain 16 bits for each key and these bits must be transmitted in turn through the IR diode using a 1 millisecond pulse of light for a zero and a 3 millisecond pulse for a one, with a 1 millisecond gap between each pulse.

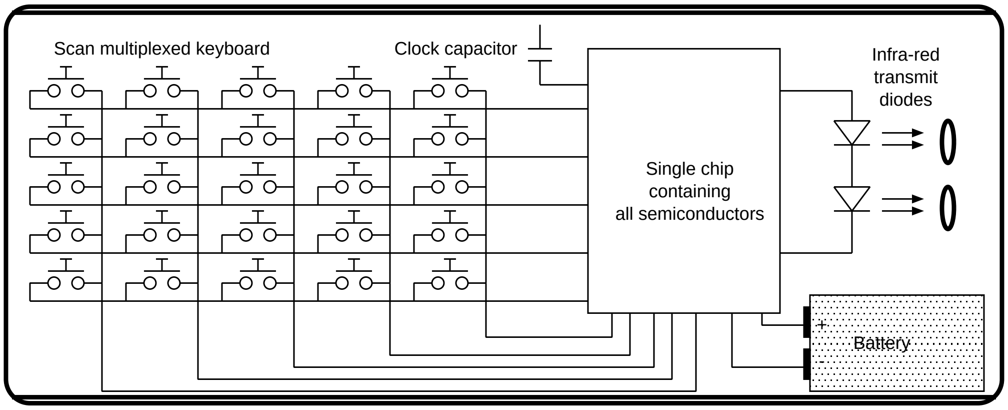


Figure 57: IR Handset Internal Circuit

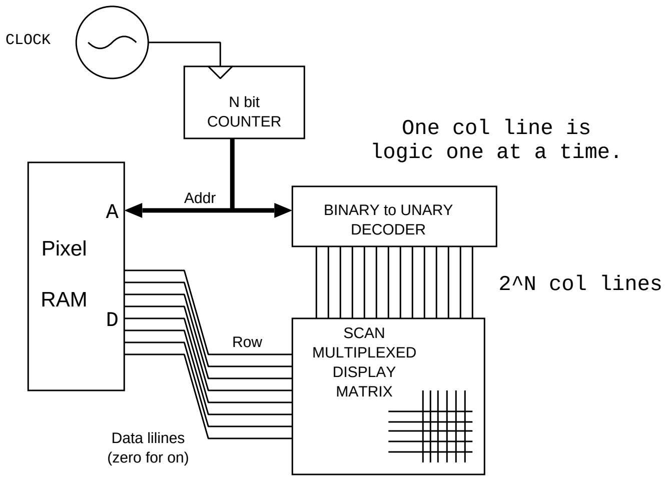


Figure 58: Scan multiplex logic for an LED pixel-mapped display

Exercise: For the IR handset, is it necessary for the clock to be running at all times or only when a key is pressed ? This design aspect will affect battery life greatly.

Figure 59 shows how a multiplexor can be added to allow update of the display RAM from a processor or similar. The display will briefly show the wrong data while being updated, but this may be imperceptible to the human. The compromise implied by such *psudo*-dualporting is accepted instead of having the cost penalty of truely dual-ported storage. (A second independent port to a RAM or register file greatly adds to silicon area).

Figure 60 shows the use of a ROM as a function look up table. All electric guitarists use a distorting amplifier and are pernickerty about the exact nature of the distortion. By using a ROM which contains a function appropriately distorted from the identity function, any desired distortion can be achieved.

The input A-to-D circuit samples the input signal at 44100 samples per second, which is the CD sampling frequency. The D-to-A converter on the output reconstructs an analogue signal.

The analog signals will vary both positively and negatively about zero. Two's complement representations are therefore normally used. Note that to convert from offset binary to two's complement, one just puts an inverter in the most significant bit.

Exercise: Sketch an alternative circuit using a microprocessor instead of the hardwired solution. What pros and cons do the two approaches have ? How does a processor help with the user interface?

Figure 61 shows the use of a static RAM to achieve a delay. The counter addresses each location in turn, and at each location, the old contents are read out before the new contents are put back.

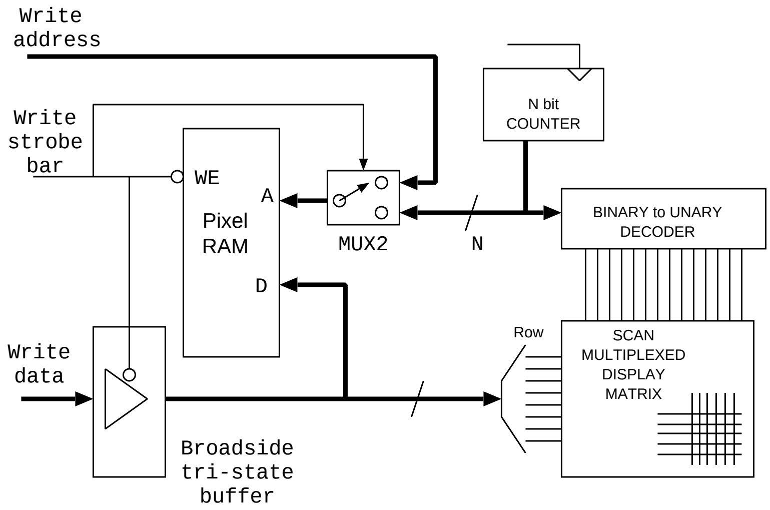


Figure 59: Addition of psudo dual-porting logic.

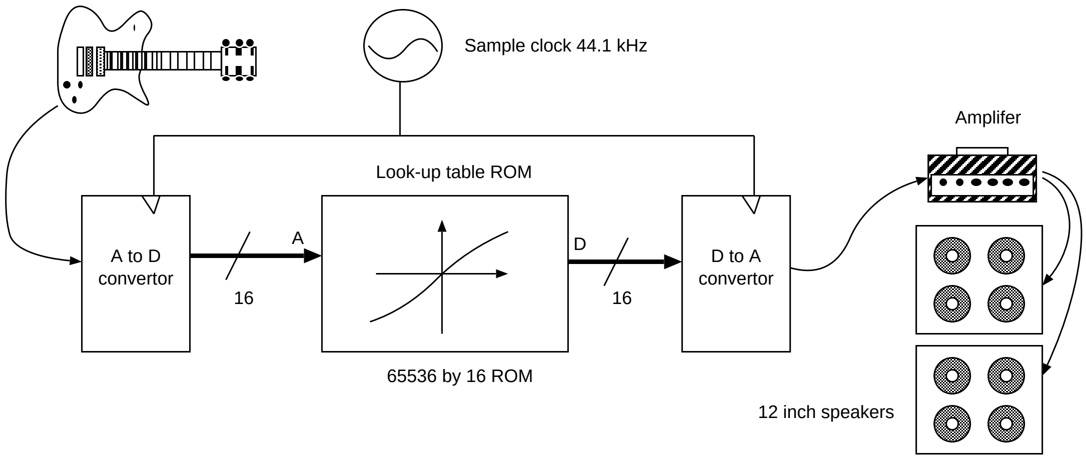


Figure 60: Use of a ROM as a function look-up table.

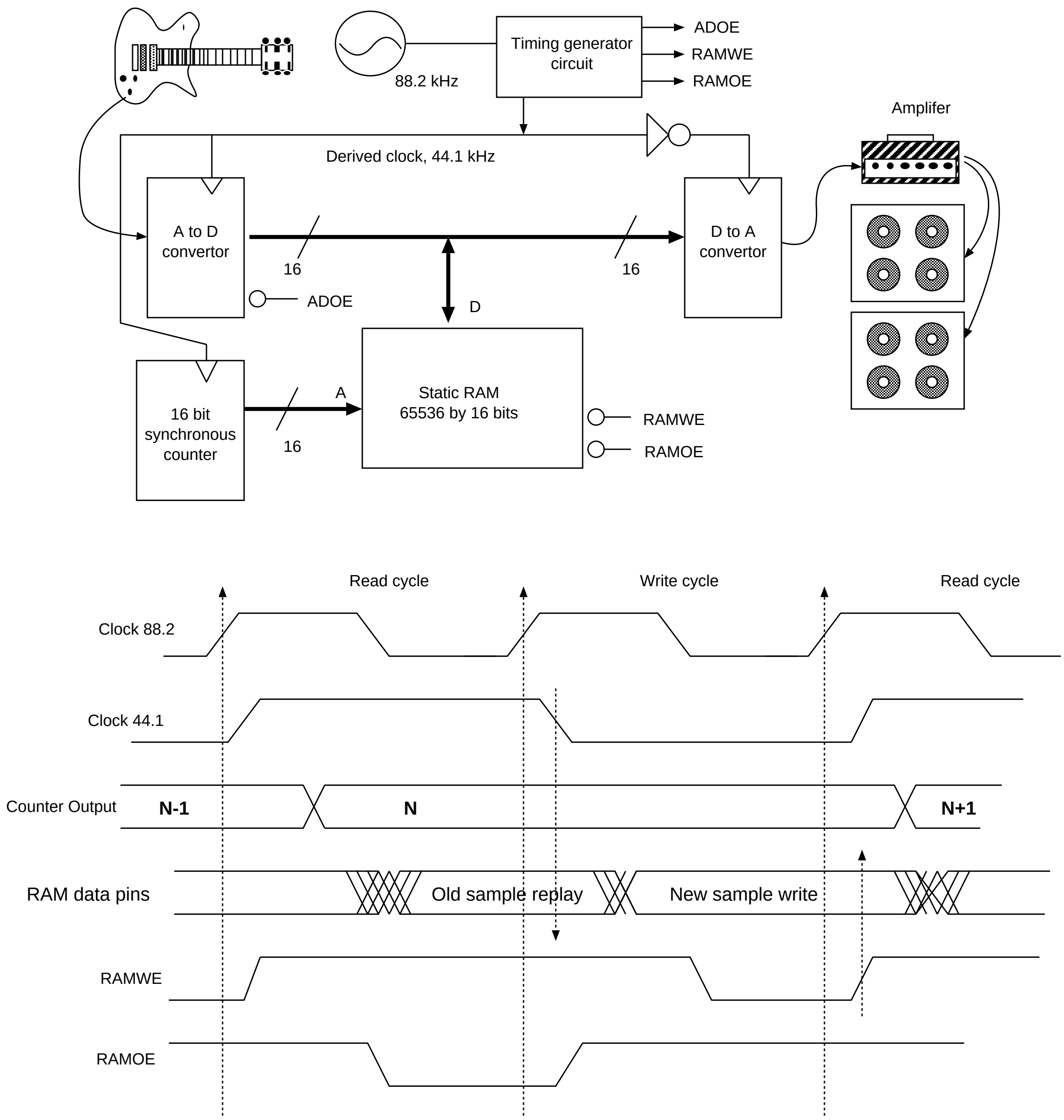


Figure 61: Use of an SRAM to make the delay required for an echo unit.

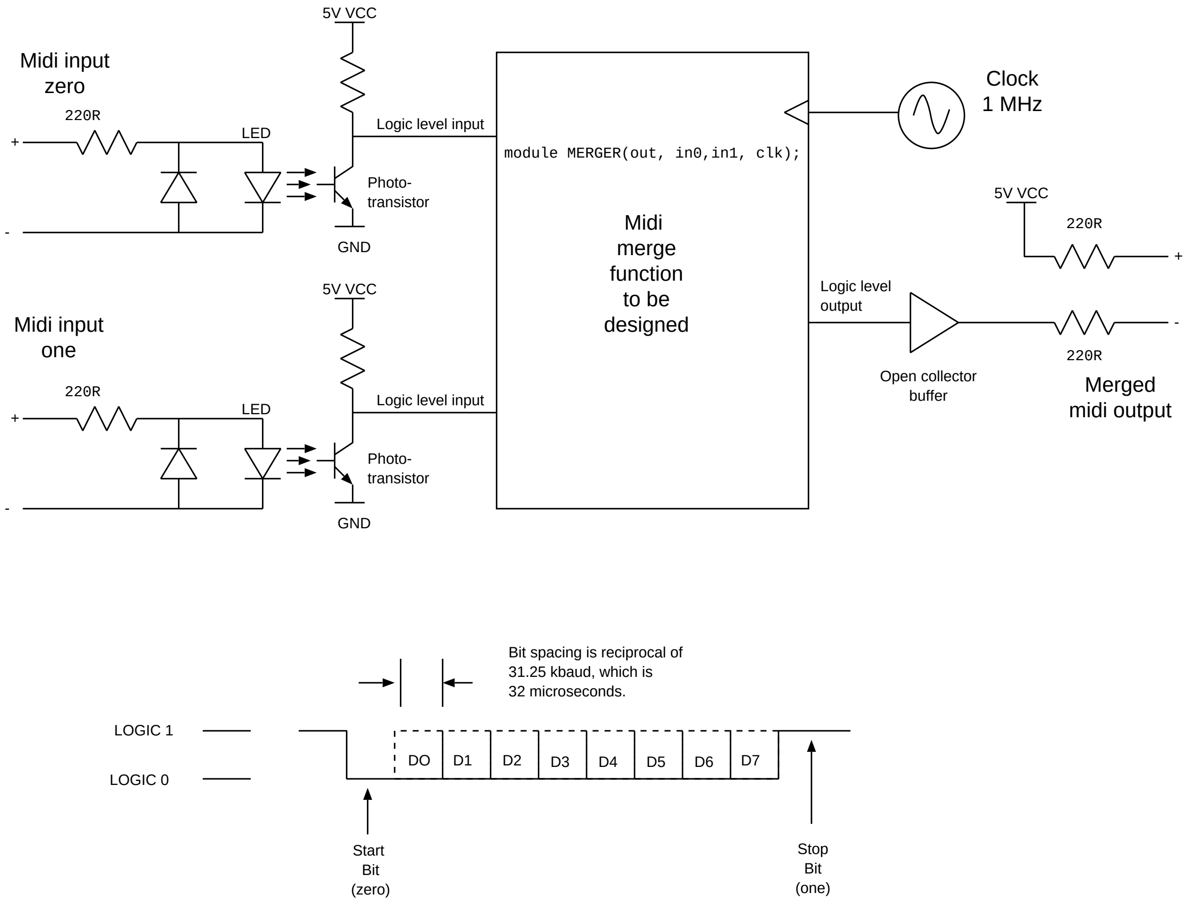


Figure 62: MIDI interface details and merge unit block diagram. MIDI serial data format.

Exercise: The circuit shown produces only a delay. For a true echo effect, the original signal must also appear at the output. To achieve multiple echos, the output must be fed back to the input at reduced amplitude. Introduce an adder and any additional logic to achieve these results. Use the fact that division by a constant power of two can be achieved by shifting right while leaving the sign bit unchanged.

5.3 Midi Design Example.

This section conatins a design example. The design example is a MIDI merge unit. *No part of this section are examinable.*

5.4 MIDI Merge Unit Specification

MIDI is the musical instrument digital interface and it is found on most electronic musical intruments. A MIDI connection is unidirectional and contains both control and real-time traffic. In this example, we will only concern ourselves with the real-time *note* data which is sent from a keyboard to a sound generator as the keyboard is played. A three-byte note command is sent for each key pressed or released on the keyboard. The three MIDI note commands we are interested in have the following formats (in hexadecimal)

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

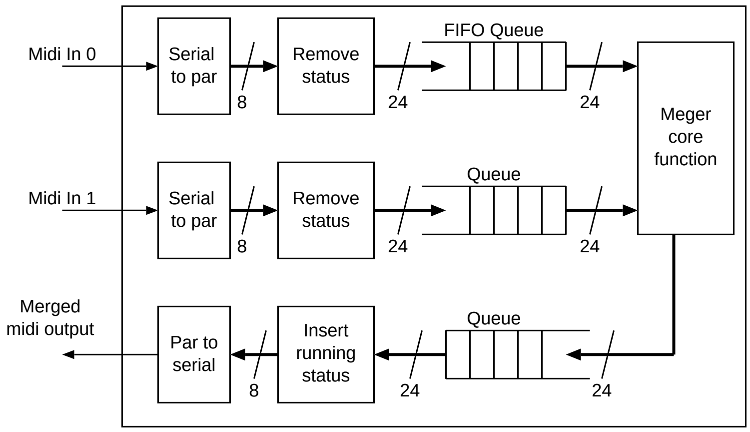


Figure 63: MIDI merge unit internal functional units.

where n is a channel number, kk is a note number and vv is a velocity. The values of n range from 0 to 15 and allow a single MIDI link to carry 16 virtual MIDI channels of data. The values of kk are in semitones in the range 0 to 127 with 60 being middle 'C'. The values of vv are from 1 to 127 where 127 is the loudest. Note that these parameters are all less than 80 hex. The first command (note on) is generated when a key is pressed and the second (note off) when it is released. The third is an alternative to the note off command when the release velocity is not used (which is the case for most keyboards). The benefit of the alternative accrues when *running status* is used. Running status is an optimisation of the MIDI protocol which brings the average number of bytes actually sent per note command down from three to closer to two. Since this is a real-time music playing protocol, and many note commands need to be sent when a large chord is played, reducing the number of bytes in this way is helpful, but it will impact our merge function. A status byte in MIDI is a MIDI byte in the range 80 to EF, and so this includes the note on and off commands. The rule for running status is that if a status byte needs to be sent and it is the same as the last one actually sent, then it does not need to be sent.

Figure 62 shows the desired configuration of the MIDI merge unit and the format of MIDI bytes on the wire. The actual MIDI cables operate with two wires carying the send and return currents: this is known as a current loop circuit. The current is on for a zero and off for a one. The current is sufficient to drive a small led which is mounted in a dark space next to a photo-transistor. This optical link between devices avoids electrical connections which can cause earthing problems in a studio environment. When light falls on the photo-transistor, it conducts and pulls down the input wire to close to ground, creating a valid logic zero level. Otherwise the inputs are one. One is the idle line state. A MIDI bytes starts with a start bit of zero for 32 microseconds, followed by the data bits, l.s.b. first.

The function of the MIDI merge unit is to merge two streams of MIDI data: e.g. two different channels of MIDI bytes (different values of n) may need to be merged. The unit must be prepared to accept arbitrary timing and status relationships between its two inputs. If they are simultaneously active, then the data from one of the inputs will have to be slightly delayed and sent after the data from the other. If both are busy to more than 50 percent utilisation, the output cannot cope and some data must be discarded by the unit. Since this is a real-time system, the output needs to have as low a delay as possible from the input.

5.5 Top down design.

The next stage of design is to sketch an internal block diagram, as shown in figure 63. The system will assemble bits into bytes, then assemble bytes into three-byte command words. The command words will be stored in FIFO (first-in first-out) queues and then read out and merged. The reverse process is done on the transmit side.

Verilog is a textual language for designing hardware. The blocks in the block diagram are each suitable for encoding as a Verilog module. Except for the final parallel to serial block, each module will need an output guard signal which indicates when the data on the output wires of the module is valid. For most of the blocks we will define this to go high once, each time a new word is valid. For the FIFO blocks, we will implement a *handshake* on the output, where the valid signal indicates that the FIFO has some data in it (and that that data is currently sitting on the output wires) and a read signal which indicates that we have accepted the data, thereby causing the next data to move to the output.

We can define the signatures of the verilog modules as follows. First, the serial to parallel convertor

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

The running status remover should be

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

For the FIFOs we should have

```
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 we should have

```
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;
```

And for the running status inserter and parallel to serial unit the signatures will be the reverse of the reciprocal units.

A full set of Verilog for the design is on the ECAD WWW page:
`http://www.cl.cam.ac.uk/Teaching/1998/InroECAD`

5.6 Software Alternative

The function could be implemented with a single-chip processor with dual serial ports. A suitable device is a PIC from Microchip and would cost just a pound or so. See http://www.microchip.com.

> *Exercise:* Consider the pros and cons of the software implementation compared with the hardware implementation. Consider that the MIDI standard is mature and has not significantly changed in ten years. Consider the real-time response possible. *This exercise might be better undertaken after the Computer Design course.*

5.7 Design Compiler Challenge

Because the MIDI protocol works in real time, for the best performance, the MIDI merge unit should have a very low delay through it. The minimal possible delay might be just one clock tick, but the design presented in lectures and above always completely receives each command before sending it onwards. A good part II project would be to write a program which read in the netlist of the MIDI merge unit and wrote out a netlist of an improved design with lower delay. The program would work by applying general purpose transformations on the netlist which can manipulate it to achieve some goal, such as low delay.

Example Exam Questions

Q. An FPGA is to be used to generate a sequencer for disco lights. The FPGA will receive some pulses that are roughly in time with the music (generated by a crude analog circuit or else by a maniac with some switches) and generate 10 or so output signals to control the lights. There are also some control inputs to select the basic mode (blackout, strobes, dark, light, pattern select etc.). Sketch a block diagram of the whole system and sketch the circuitry of the FPGA. Is using an FPGA a good idea?

Q. A design is required to provide timing and information displays for the international final of the table-top Grand Prix (Scalelectrix) competition, which is to be televised by Eurovision. The budget for the system is fifty thousand pounds, including the sums paid to consultants (like yourself) and the system is expected to last for several years. Cars will be fitted with unique active tags, as used for counting livestock though gates on farms, and a sensor for the tags, together with light beams for accurate car detection, will be placed at the start-finish line and two intermediate points on the track. A number of display panels are needed based on LCD VGAs and a master operator's console is needed to control the system. Design the system. Pay attention to the modules needed and the wires that interconnect them. Give a 100 word instruction manual for the operator. If microprocessors are used, explain why each one is needed.

Q. What is semi-custom design ? When would you use semi-custom? Why might you use an FPGA to prototype a design that is later to be built in semi-custom? What cost difference would you expect between an FPGA and a semi-custom versions and why? How might your partition be affected by the amount of RAM needed?

Q. A design is required for a wand computer toy. The wand will have a barcode reader which will scan barcodes found on groceries and other goods and accumulate credit inside the wand in an unforegable, non-volatile way. A group of young friends, each with their own wand, will be able to compete against each other, gathering points from each item scanned, with more points for rare/valuable goods. Design the wand, including its display and controls and a method for wands to connect to each other to exchange/swap barcode sightings.

Q. A circuit is needed to divide a very high frequency clock by an average of 21.6. This is to be done with a counter which accepts a division ratio input to make it divide either by 21 or 22. A further counter, made from slower and cheaper logic, is clocked from the output which generates the control signal to the divider, such that the ratio of 21 and 22 counts overall gives the desired target division ratio. Design it. Prove there are no hazards in your design. Estimate the maximum clock frequency given that the setup time and gate delays of the higher speed logic are 1 nanosecond. (This question is no longer core to this course).