

CHAPTER 2: TECHNOLOGY, PACKAGING AND MANUFACTURING 2/13/78

TECHNOLOGY PUSH

Whereas the rest of the book gives examples of the effects of technology push, *this*
in this section we *define* the individual components *within the context of use.*
As users, we must have a model of what the technology has been so that *that have made up Technology push.*
short-term extrapolations are possible.

Rather than of
We should, for completeness, discuss all the technologies shown in the tree of
the relevant memory and logic technology in Fig. Mementax. *Instead we only*
will concentrate on semiconductors, as they have been the dominant factor. *Semiconductors have been the dominant factor in technology push, but*
Magnetic recording density, on disks and tapes has evolved rapidly too, and must be
understood as a component of cost and a limit of performance. *Therefore, we*
present a cursory model of it. We have ignored communications links (because
they have not evolved as rapidly as they might). *The packaging (cabinets and*
boxes), interconnection and power aspects are given in the second section. *discussed*
These do not represent pushes but rather are large, high inertia objects that
have to be pushed against and eventually yield.

24
In the semiconductor section we *start* by presenting a tree of the possible
technologies by the function they carry out and show how these have evolved
over the last two or three generations to affect computer engineering. The
other relevant cost, semiconductor density, performance, and reliability
parameters are also briefly discussed.

G. Bell

created 1/4/78, latest edit 2/13/78

Following the semiconductor evolution model, ^{the} we discuss ^{it moves on to} how the semiconductors have been applied, using various logical design methods, and how they have pushed the methods. Of special interest is ^{IC} ~~how~~ and whether users design ICs, ^{or} as opposed to design with ICs. (The conclusion should be obvious...IC design is expensive and time consuming and should be avoided for all but a few applications.) The advice can also be carried over to those who believe it is necessary to program in assembly language or to have access to microprograms. For some reason, ^{people believe that} the belief is that there is a need to have access to lower level building blocks when building many large objects. ^{This is akin to finding a} We advise using ^{sand pit and} bricks and mortar, not finding a sand pit, cement and brick factors to ^{a clay formation} fabricate bricks and mix mortar.

There are two sections on memories. ^{The first, on} The core memory, provides a number of lessons on managing evolving technology, ^{Core} it was the dominant primary memory for about 15 years even though it was challenged several times. It eventually lost because of the larger amount of development being applied to semiconductors. ^{memory.} A section describes the notion of memory hierarchies and how they operate to utilize virtually every memory that's either cheaper or faster than any other memory. ^{the second memory section}

In the final section we present some general observations about technology evolution: how technology is measured, why it evolves (or doesn't), cases of it being overthrown, and a general model for how it operates and is managed.

Since the chapter is on the evolution and interaction of logic and memory

looking for
 finding a
 sand pit and
 a clay formation
 from which to
 make bricks and
 mortar when
 attacking the task
 of building
 construction.
 It is much
 more practical
 to buy the
 bricks and
 mortar.

G. Bell

created 1/4/78, latest edit 2/13/78

technology as they effect computer design, the best way to observe the change is via the time line diagram (see Fig. LMtimeline). For logic, small memories, read-only memories and primary memories four lines give the significant events (~~both in the industry and DEC~~) that have affected the technology push. The remainder of this section will refer back to the tradeoffs.

SEMICONDUCTOR TECHNOLOGY

A single transistor circuit performing a primitive logic function within an integrated circuit (IC) is among the smallest, ^{and} most complex of man-made objects. Alone, such a circuit is intrinsically trivial; but the fabrication process for a set of structures to form a complete integrated circuit is complex. To us, ^{as} ~~as~~ digital IC users, there are several relevant parameters:

1. The function an individual circuit performs within the IC, the aggregate function of the IC, and the functions a complete IC family perform.
2. The number of primitive digital switching circuits (or transistors) per IC. This density is a measure of the capability of the IC process.
3. Cost.
4. The performance of each circuit and the performance of the aggregate IC and/or set of ICs within a family as they affect system performance (as measured by the time it takes to perform its function). The semiconductor

G. Bell

created 1/4/78, latest edit 2/13/78

circuit technology family usually determines this performance.

(TTL, Schottky, ECL)

5. The number of interconnections (pins) to communicate outside the IC. This in turn interacts to permit more complex functions to be performed by a complete IC family.
6. The reliability. This parameter is a function of the circuit technology, density, number of pins, the operating temperature and use (or misuse).
7. Power consumption.

Function

Figure ICtree shows a family tree (taxonomy) of the most common digital IC's, roughly in order of increasing complexity. The secondary ordering is roughly by the regularity of the function being implemented, and whether there is memory associated with the function. The clustering of circuit types is also by technology generations (from the second to possibly the fifth). Note that we can build large totally regular functions with only memory (e.g., various registers and memories) and with no memory (e.g., adders, multipliers, multiplexors). With large scale integrated circuits it is desirable to implement regular structures to simplify understanding, and aid the testing build in generality.

In the early third generation only completely unconnected components were built. Collections of the basic logic primitives (AND, NAND, Exclusive OR,

G. Bell

created 1/4/78, latest edit 2/13/78

Adders) and sequential circuit components (i.e., separated flip flops or collections to simple form registers) permitted what had been logic functions that occupied a ^{had} single printed circuit board module of the first and second generations to occupy a single IC. ^{step} This forced modules to take on specialized functions. ^{to be collection of} The drastic reduction in size from the second to the third generation transition can be ^{seen} most vividly, ^{by comparing} by example in the 18-bit ^{PDP-9 and} implementations for the PDP-15 (page 00).

As the densities began to improve to a hundred gates, complete arithmetic units began to exist; and the earliest and most famous function, the ALU (Fig. ALU), provided up to 32 functions of two variables (each 4-bits). By the fourth generation, purely combinational circuits included a complete 16 x 16-bit multiplication circuit on a chip, requiring about xx gates.

Without well-defined functions such as adders and multipliers, ~~there is no way~~ semiconductor suppliers can ^{not} provide high density, high volume products because there are not large scale, general purpose universal functions. The alternative for the users is interconnecting simple logic circuits (AND gates, flip flops). This, of course, does not permit efficient use of the technology and the cost per function remains high (about that of the third generation) because the printed circuit board and IC packaging costs (pins) limit the attendant cost reduction.

The problem of effectively utilizing LSI by customization methods will be discussed after we have first traversed and discussed the tree. For now, the

TP The methods of *TP* are shown in Fig 1.3. There are the PLA and the gate array.

G. Bell

created 1/4/78, latest edit 2/13/78

PLA (for Programmable Logic Array) is an array of AND-OR gates for combinational design that can be interconnected to form the sum of products terms. Specialization ^{by blowing fusible links,} can be done either in the factory or in the field (FPLA) to give the general functions of input variables. ~~Shown in the next~~

~~branch is~~ ^{is} the gate array structure, an alternative for doing logical design, ^{but is more powerful than the PLA because both combinatorial and sequential circuits can be constructed.} Gate arrays are simply a large number of gates placed on the chip in fixed locations which can be interconnected together ^{and} during using metalization ~~done~~ in the final stages of semiconductor manufacture.

^{of the tree shown in fig 2.6} There is a special branch for the purely memory functions. Since memories have so many uses, this branch is discussed separately in the memory section.

Memory is ^{not only} used in the processor as conventional memory, but it is also an ^{alternative for performing combinational logical functions, (i.e., a value can be looked-up rather than computed as is custom for logical design) and sequential logic functions (i.e., the memory can hold states in a microprogram).} ^{can be used as}

^{For example, the inputs to a combinatorial function can be used as an address and the output obtained by reading the contents of that address.}

^{the memory can also be used to implement sequential logic functions. For example the memory can hold states in a microprogram.}

The remainder of the interesting logical functions include combinations of logic and memory. There are various special ^{such as} functions like encoding

algorithms (e.g., LPC, for Linear Predictive Coding) ^{for} to use in real time and communications systems. Similarly, data encryption can be carried out

utilizing a single, special LSI chip. One of the most useful transducers, and

the first one to use LSI, was the UART, ^{for} Universal Asynchronous Receiver

Transmitter). The reader is invited to read our description of our interaction

to define the UART.

data encryption algorithm for use in...

G. Bell

created 1/4/78, latest edit 2/13/78

~~Although we consider the circuit function to be an attribute that's separated from others,~~ ^{performed on a chip} it is clearly dependent on the number of ^{gates} circuits which can be placed on a chip. Thus, circuit density in ^{gates} circuits/chip is the single most important parameter. ^{in determining chips from technology} From this measure, we can predict the functions likely to be implemented by just following the tree. It should be noted that the whole tree is relatively alive, ^{i.e. has dense areas of new branches} i.e., increases in density. ~~Both improves a node and allows another tree branch to grow.~~ Only the very low density branches at the top (e.g., unconnected gate and register structures) are relatively static. ^{On the other areas,} Thus as density increases sufficiently, a new branch grows. For example, the processor-on-a-chip started out as a 4-bit processor (or rather as 2 chips for a single processor) and then progressed to have 8-bit and 16-bit processors on a single chip. Similar effects are observed with the arithmetic logic unit, memories, etc. ~~as density improves.~~

^{gates} The number of circuits per IC is the measure of density as seen by a user (see Fig. Semiden). This metric is ^{actually} the product of the circuit area times the number of circuits per unit area. Circuits are ~~also~~ ^{also} photographically reduced in size to yield higher speeds, higher densities and to have better yields. ~~A third factor,~~ ^{and} circuit and device innovation, also contributes to density increase. Both measures improve with time, reflecting improvements in fundamental process control.

Semiconductor device (individual circuit) performance is also correlated with the implementation technology and density because reduction in size also reflects reduction in power.

G. Bell

created 1/4/78, latest edit 2/13/78

An Operational Model for Memory Size versus Time

The model given in Fig. Semidens is exponential and correlates with ~~our previous~~ observations (Bell) in 1975 that the number of bits per chip doubled ^S every two years according to the relationship:

$$\text{number of bits per chip} = 2^{t-1962}$$

There are separate curves, each following this relationship, for ROMs in prototype quantities, ROMs in production quantities, RAMs in prototype quantities, and RAMs in production quantities. Thus, depending on the product and the quantities, one
 This does not give account for the technologies, whether read only or read write, and whether the time value is associated with production quantities or a laboratory curiosity. We must scale the state of the art line appropriately by one or two years according to the following rules:

[rules]

This gives the following availability of various semiconductor memories:

<u>Year when first widely available</u>	<u>Number of bits</u>
1969-70	16
1971-72	64
1973	256
1975	1024
1977	4096

The significance of these values is that they determine where certain architectures and implementations can occur. The chapter critiquing the PDP-11 uses this model to show how semiconductors accomplish this push.

Cost

G. Bell

created 1/4/78, latest edit 2/13/78

The most important characteristics of ICs after ^{density is} considered cost.

The cost of ICs is probably the hardest of all the parameters to identify and predict because it is set by a complex marketplace. For circuits that have been in production for some time and for memory arrays, the price is essentially ^{year of} a commodity ^{like eggs or bacon.} For more circuits that have not yet reached ^{this} commodity status, the prices are higher and depend on the strategy of the ⁹ supplier--whether he is willing to encourage competition. ^{As users, we} ^{generally} consider ICs ^{as be} as commodities, with the attendant benefits (cost) and problems (having a sufficient source of supply). In these cases, the prices are proportional to the die cost (i.e., the die area) and the manufacturing volume.

Two curves are presented to reflect ^{the} price of various components ^{implemented} ICs. Figure Comprice shows the price per component for an IC assuming LSI. There is a price band for the circuit size and circuit technology. Table GateComp gives some idea of how circuit density (in components) relates to actual functions.

The most useful data to understand past and future computer structures is the semiconductor memory cost curves (see Fig. Memprice). Here, the basic cost/bit of various sized memories is given. ^{was} We note that ^I in 1978, the cost per bit ^{is} roughly .08 and .07 cents per bit for the 4 Kbit and 16 Kbit IC chips respectively, giving ^{chip} costs of \$3.30 and \$11.50 ^{for the ICs} respectively. Whereas the chip density improves by a factor of two each (Fig. Semiden) year, the cost per bit (at the IC level) is declining at only a factor of 2 every two years. The line drawn in Fig. Memprice has the equation:

$$\text{cost/bit} = K \times 0.5^{t-1978}$$

G. Bell

created 1/4/78, latest edit 2/13/78

It is also interesting that the cost compares favorably with the price decline observed in core memory over the period since 1960-1970 for the 18-bit computers, (page 00) and for the cost declines in both PDP-11 and the PDP-8 (pages 00 and 00).

Performance

The performance for each semiconductor technology evolves at different rates depending on the cumulative learning associated with design and manufacturing process together with the marketplace pressure to have higher performance for the particular technology. ^{one} We may hypothesize that each technology can be looked at as being relatively appealing or relevant to the particular design(er) styles associated with the computer market levels (view 4, page 00). One would expect the evolution to continue along the lines shown in the following table for the next few years.

Table SemChar: Characteristics of Dominant (1978) Semiconductor Technologies

<u>Type</u>	<u>Meaning</u>	<u>Evolution</u>	<u>Use</u> (FIX THIS)
TTL	Transistor-Transistor Logic	- TTL/Schottky TTL/LS	logic, bus interfacing high performance low power, relatively same as TTL
ECL	Emitter Coupled Logic	MECL II, III MECL 10K, 100K	original, very high performance delays of ___ and ___ ns.
MOS	Metal-Oxide-Semiconductor	p-channel n-channel	for cost for greater densities, cost evolving to performance (memory)
CMOS	Complementary MOS		for low power, speed,

G. Bell

created 1/4/78, latest edit 2/13/78

have been omitted.

MOS

noise immunity

Note that we have omitted some of the lesser used technologies such ^{as} I²L¹ for Integrated-Injection Logic and SOS for Silicon on Sapphire, which both promise higher speed and at lower power than MOS, and hence might displace its use. ^{than the} Both of these technologies have been touted as replacements for ^{the} ~~the~~ ^{technology} ~~technology~~ shown in the table. Thus, if an entrenched technology has evolved for some time and continues to evolve, it is difficult for alternative technologies to displace it because of the cumulative investment in process and understanding. This is clearly seen in the case of the numerous technologies that attempted to displace the core memory technologies during the 15 years it dominated primary memory use. (It lost because more effort was applied to MOS technology).

shown in the table.

We have also omitted the early predecessor technologies, (RTL for Resistor-Transistor Logic, TRL for Transistor-Resistor Logic) and DTL for Diode-Transistor Logic because they were sufficiently deficient in some aspect of cost, performance, or reliability to avoid becoming standards of note. ^{too become} Here, we should point out that these were the first ICs, and as such DEC did not use them because they did not represent an advance (to us) over the discrete transistor circuits ^{already in use,} we used (e.g., the PDP-8). ^{In addition,} Similarly, these early circuits were packaged in a flat package, not the dual in-line package used today, hence machine insertion of components was not possible.

The table of parts, above, is weighted heavily to DEC's use: TTL for mid- and high-sized minicomputers; ECL for the larger scale DECsystem 10; MOS for memories, microprocessors, and specialized high density circuits; and CMOS for the CMOS-8 (because it exists...not for either lower power or high noise

¹Semiconductors appear to be characteristic of other technologies in that usually only a single technology is used for a given problem (use)--that is, each technology has a niche and there is only one winner!

G. Bell

created 1/4/78, latest edit 2/13/78

immunity reasons although they are nice).

Figure Speedpwr and Gatedelay give the two most useful measures of performance for the various technologies as they have evolved with time. The performance of circuit is also proportional to the power applied to operate at it. The speed-power product metric for a technology at a given time is a more refined measure and indicates that the user may tradeoff performance against power.

There are limits to that tradeoff because
However, this is a bit misleading because unless special cooling is used, only about one watt can be dissipated by an IC package, *unless special cooling is used.* The first curve presents the speed-power product and was generated by Jerry Luecke of Texas Instruments (TI) at a time when I²L technology had just been introduced (Oct. 1975) by TI; some recent data points have been added for recent advances in MOS technology. Table Speedpwr gives some of the operating points for common processes at various times. Figure Gatedelay shows the gate delay through a single gate (logic) circuit and is the best performance measure because the implication of speed and power are interchangeable is erroneous. That is, one can neither operate at very high or very low densities subject to the package power dissipation constraints.

Interconnections Outside the IC

The number of pins that communicate outside the circuit indicate the packaging technology level. Low and medium scale IC's often need more pins than LSI since the latter is capable of carrying out a complete function independently - and hence can operate longer without *less* interaction with the rest of the world. Memories are especially easily adapted to live with *in* pin limitations, since only

G. Bell

created 1/4/78, latest edit 2/13/78

1 pin is required for every factor of 2 increase in size, (roughly every year).

By time multiplexing the address signals, the requirements for address bits is

further decreased.

Reliability

Over the past 15 years, the failure rate for standard ICs has been reduced by two orders of magnitude to the neighborhood of .01% per 1000 hours. This corresponds to 10^7 hours mean time to failure (MTTF) per component. Figure Reliab, from a recent survey article by Hodges [1977] shows the trend. The lower curves show the higher reliability obtained when more extensive testing and screening is employed. The improved MTF of between 10^8 and 10^9 is obtained at a cost increase of 4 to 100 times per component.

Power Consumption

The tradeoff of performance and power for a given technology is implied by the speed power product described above. The package constrains power dissipation as previously described.

EVOLUTION OF SEMICONDUCTOR USE

The LSI Dilemma

The economics of the LSI circuit industry make it essential that circuits with a high degree of universality be produced. Because of the learning curve of a manufacturing process (said to be X% per m circuits), cost is inversely proportional to volume. For a design to be sold in high volume, it must be

G. Bell

created 1/4/78, latest edit 2/13/78

usable in a large number of applications. However, the trend in circuit complexity, which allows semiconductor manufacturers to put more transistors on a constant die area each year, lends to increasing specialization of function. The more specialized the function, the lower the potential volume and the higher the price. Figure generally shows the dilemma. The LSI product designer is therefore continually in search of universal primitives or building blocks. For a certain class of applications, ~~e.g.~~^{such as} controller applications, the microprocessor is a fine primitive and has been so exploited [Noyce, 1977]. For some applications, circuit complexity can embrace even higher functionality at the PMS level. The Intel 8XXX is an interesting example here: two processors, a 2.5(?) microsecond byte-processor and a 200 nanosecond bit-processor are combined in one LSI circuit.

Moore [1976] discusses this dilemma in a paper on the role of the microprocessor in the evolution of microelectronic technology. He points out that a similar situation existed when i.c.'s were first introduced. Users were reluctant to relinquish the design prerogative they had when they built circuits from discrete components. It was not until substantial price reductions were made that the impasse was broken. Then the cost advantages were sufficient to force users to adopt circuits that fit the technology.

The difference, however, between the i.c. switching circuit level of RT-level building blocks on the one hand and the PMS-level block on the other is that the latter uses the powerful concept of a shared-program computer. Its function is varied by programming.

G. Bell

created 1/4/78, latest edit 2/13/78

For many applications, including most computer systems, the microprocessor on a chip is not a cost-effective building block, and other solutions to the dilemma are found. For example, microprogramming is a highly general way of generating control signals for data path elements, table lookup is a highly general technique. Both methods are attractive because they use memory, an inherently low cost LSI circuit.*

Other techniques involve changing the pattern of interconnections via the top level(s) of metallization of an integrated circuit. This can be done by a factory made change or by the user, a much more flexible solution. Some of this customization is required in nearly all digital system design of the fourth and fifth generation. At the very minimum, the values for a read only memory have to be set. Table funvar contrasts the building blocks available in the fourth generation.

As a result of the increased basic circuit functionality available at each new generation, design methods have changed with the generations. This book provides an example of each, as summarized in the following table.

Table: Design Method versus Generation

<u>Design Method:\Generation:</u>	<u>First</u>	<u>Second</u>	<u>Third</u>	<u>Fourth</u>	<u>Fifth</u>	<u>Examples in this book</u>
Combinational and sequential; use of "standard" modules, IC's.	s	s	s			18-bit; PDP-8
Read only memory and PLA; microprogramming			s	m		PDP-9; PDP-11

*Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing rate to be 700 microwords per man year for horizontal micro machines. This tends to limit the maximum control store size to about 16 Kwords.

G. Bell

created 1/4/78, latest edit 2/13/78

Microprogramming with standard RT elements (high perf.) minor logical design			s	m	CMU-11
Programming using micros and logic for interfaces	p	p	s	x	LSI-11
PMS design using completely specified and pre-designed microcomputer components				s	Cm* (almost)
Customized chip design and standard logical design (high performance)		m	m	m	LSI-11

s - the standard method for most digital systems

m - done by manufacturers of basic equipment

x - also used

p - prelude to micros, also done using minis

Table funvar

Building block	Technique for varying function	Degree of generality	Permanence of change
Computer module	program	v. high	none
microprocessor	program	high	low to medium
bit slice	microprogram	medium	medium
ROM	factory mask change	v. high	irreversible
PROM	field change	v. high	irreversible
EAROM	field change	v. high	low
PLA	factory mask change	medium	irreversible
FPLA	field change	medium	irreversible
gate array	factory mask change	medium	irreversible
RAM	write	v. high	none

The design of most relatively high speed digital systems (including low- to

*Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing

G. Bell

created 1/4/78, latest edit 2/13/78

mid-range minicomputers is carried out using standard register transfer ICs complete with data path and memory. Part IV discusses the current standard RT modules set, ~~we would expect~~ ^{well} these bit-slice components ^{register transf} to evolve and take on specialized functions ^{in order} to achieve both higher performance (e.g., a multiplier) and to be specialized to particular tasks (e.g., the interpretation of a particular computer's ISP). As an example, the series may be modified for the coding and encoding required in signal processing.

For higher performance computers, there is no alternative to using either tightly packed standard ICs ^{or to build a particular set of ICs using some form of customization.} Although Cray continues to build the high speed computers (in the CDC 6600, 7600 and Cray 1) with no custom logic, he does so by using impressively dense modules with high density interconnection and freon cooling. The high performance IBM and Amdahl machines ^{use} are custom ECL circuits to improve packaging.

The current spectrum of IC's in use is summarized in Table cost speed spectrum.

With the advent of the processor on a chip, digital system design has ^{been} or will ^{soon} be ~~soon~~ converted completely to computer design. The ~~process of dissolving a particular problem~~ ^{such as} (e.g., control of a CRT, a lathe, or building a billing machine or word processing system) ^{is just computer design system design as it has been practiced over the first three generations.} The hardware part of the design ^{is straightforward} i.e. (the interface to the particular equipment) (e.g., keyboard, lathe, printer), ^{whereas} the major part of the design is

* Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelation is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing

G. Bell

created 1/4/78, latest edit 2/13/78

programming, in the same way that ~~it has evolved since the late 40's.~~ In fact, we have seen three complete generations learn about this form of computer design, especially programming. The first generation discovered and wrote about it. We rediscovered and applied it to minicomputer systems. Now this time, it is being learned by everyone who must use and program the microcomputer. Each time (for each individual or organization) the story is about the same: people start off by programming (using binary, octal or hexadecimal codes) small tasks, using no structure or method of synchronizing the various multiple processes; the interrupt mechanism is learned, and the symbolic assembler is employed; and finally some more structured system--possibly an operating system is employed. Occasionally users move to higher level languages or macro assemblers, but usually not because we (as engineers) are taught to minimize product cost (as opposed to development, life cycle, maintenance, etc. costs).

In view of the cyclical history, it seems likely that
 We believe the design of systems as currently practiced will be relatively short lived. *The new* design method here will be at the PMS-level component, *although* it is still too difficult to be done reliably and cheaply by large numbers of engineers. *Even though* by comparison with logical design and microprogramming, *current* digital systems design is straightforward and consists of building simple hardware interfaces to relatively poorly defined busses together with programming the application. *Therefore*, we believe that the components from which the microcomputer systems will be formed will be significantly more advanced using much better packaging, clearly defined busses, standard more general interface, and base level operating systems that are embedded in

* Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing

G. Bell

created 1/4/78, latest edit 2/13/78

hardware (i.e., placed in read only memory to give the feeling of permanency so that users are less likely to embark on the expensive, unreliable rediscovery path). Standard components will be built which can be interfaced to a wide range of external systems using parameters that are specified by a field programming method instead of ^{by} using logical design and building with interconnection on modules. In this way, the complexity of individual ICs can be increased and ~~thereby~~ having a standard method for interconnection, higher volume and lower costs will result.

Before ~~we~~ ^{my} discuss the alternatives associated with IC design, it is important to characterize the various costs. Figure Design.cost shows, at a crude level, what ^{one} we might expect the relative design costs to be for various inter- and intra-IC design methods. Even the design cost is highly variable depending on the project size, its goals, manufacturing volumes expected, and more importantly on the computer aided design programs.

The lowest cost designs stay completely away from modifying the ICs, except to ~~bind~~ ^{to} programs to read only memories. If these use masked ROMs, the design costs are proportionally higher in order to both make the masks and to make corrections in the event that the ROMs are incorrect. (The manufacturing costs are lower for permanent ROMs.) At the RT level, the standard microprogramming design method is (conservatively) only twice as expensive per instruction as conventional programming, but likely on the order of 5 to 10 times as expensive to solve the same problem that a program written for a microprocessor solves (the speeds are at least a factor of 10 more too). Given that one must design

* ~~Microprogramming~~, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing

G. Bell

created 1/4/78, latest edit 2/13/78

controllers with ROMs and PLAs, the cost can be still higher, but lower than with the standard sequential circuits we used in the first few generations--particularly if the module etch is used for the interconnection structure (see also comments about interface design for the 18-bit computers, page 00).

We make no attempt to quantify the manufacturing cost per function using the various design methods because the costs are too dependent on the quantities produced, the manufacturing technology (particular testing) and even how the functions are measured. We intuitively feel that the costs are relatively in the same order as the design costs, however, the cost per function is higher with SSI and declines with density. Doing intra-IC design should yield lower cost for all except standard programming and microprogramming because this method of design is identical to conventional logical design (of the first three generations) and we have come full circle! Therefore, if we postulate that design costs are higher and it is possible that the manufacturing costs are higher with custom design, then why would anyone design their own ICs?

Design of ICs (Intra-IC Design)

There are numerous reasons why a designer is forced to design ICs. In some engineering environments where there are extremely small space, low power, or extremely high reliability requirements, the engineer is forced into building ICs, without attendant cost savings. Within DEC, however, unless there is a significant manufacturing cost advantage, we do not build an IC. Although the cost differential might be marginal, the performance gain for specially

*Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing

G. Bell

created 1/4/78, latest edit 2/13/78

designed ICs is often high. As we pointed out in Cray's computers, not designing ICs (permitting greater design flexibility, lower design costs and faster design turnaround) is simply a tradeoff with special high density packaging and freon cooling. Neither IBM nor Amdahl make this tradeoff, nor do their computers run as fast (but they do build many more computers, more cheaply). Therefore, the added complexity of ICs may be the only way that a high volume product at a given performance may be possible. The maintenance costs are important with higher volume complex designs since the reliability of a system is mostly a function of the man-made explicit connections (including the bond to the semiconductor die) and we can ignore the reliability of the IC die interconnection. Thus reliability is simply measured by counting discrete circuit pins, IC pins module and connector pins to determine the reliability.

To summarize, IC design is used along the same lines as we observe design styles: we build them for performance (for reliability too, because it may be the only way that a complex design can be maintained); we build them to get some decreased cost and higher performance (this permits mid-range designs to be more cost effective); and finally we build very high volume, lower performance computer components--i.e., microprocessors and microcomputers because it is the cheapest way to do a task. The secondary reasons to get size (and cost) reduction for some design occasionally enter in too. With greater semiconductor densities for the non-microcomputer components used to build more general digital systems, the increase in density is double edged--and may force specialization. That is, with these more complex components, there is a greater risk that as they become too large they will become less useful for

* Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing

G. Bell

created 1/4/78, latest edit 2/13/78

building a particular system. This is akin to building with bricks that continue to grow in size. At some point, the brick size is well beyond the size of the object being built, and the only way to get the intended function is to sculpture the brick back to a useful form.

The various design methods we might use for various objects and densities are given in the following table.

Table: IC design method versus semiconductor density.

IC Design Method	Density			
	SSI	MSI	LSI	VLSI
Minor variations in standards ckts.(high perf.computers)	busses	RT components	Special interfaces - (e.g., UART)	
Gate arrays	-	Set useful for integrating a large system (e.g., a computer)		possible alt, to custom design
Standard cells	-	Small system relatively high volume system		"
Custom design (for high volume parts)	bus interface, signal convers.	high perf.	memories, microcomputers including peripherals	

Therefore, the most straightforward intra-IC design may well be the modification of an existing design. This has been used extensively by use to get the components for implementing computers. Failing, slight design

* Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A more subtle, but practical, limitation is the development cost of microcode. We have measured the writing

G. Bell

created 1/4/78, latest edit 2/13/78

modifications, we are left to the next most straightforward task, designing using arrays of gates and then interconnecting them to form the desired function. This is still desirable because there is only one circuit from which the gate is formed and the gate can be completely parameterized and defined so that interconnection is logic design that we understand. Also design occurs in a completely defined environment. Gates are interconnected to form more complex macro structures (e.g., various flip flop types, adders) and the design is carried out using the arrays of arrays.

It should be noted that this style of design, though never practiced at DEC was one of the extreme design philosophies advocated in the first few generations. Namely, there was a single module¹, containing a set of gates, and all subsequent logical design was done in its terms (e.g., flip flops are constructed by interconnecting the gates). Note that a design predicated on a single module type simplifies the spare stocking and servicing aspects immensely, and it is possible to troubleshoot a problem by simply replacing circuits according to a pattern. At the other extreme of this design style is to make all modules different from one another and specialized to each task.

Gate array design uses a single well-understood, well-defined component and the fabrication of all but the last few semiconductor processing steps is identical for all designs (the interconnection of the gates by metal is carried out last). However, there is lower density than more custom methods such as the standard cell (or more precisely, standard logical design element).

Standard cell design is identical to the logical design of the first few

¹In the most recent Cray design there are only three IC types used: two are memory configurations and the third is a set of gates that are interconnected using the printed circuitry of the module and backplane. Thus, while only 3 IC types have to be stocked there is still a large number of module types, and servicing would hardly be done in the field by changing (unsoldering) ICs in the module.

G. Bell

created 1/4/78, latest edit 2/13/78

generations since there is a well-defined set of components (e.g., AND gates, flip flops) in which the design is carried out. The advantage is density. While the disadvantage is that each cell occupies a different space and hence improvement in packing density may not materialize. Similarly, there are a large number of circuit types and the set may not be defined well enough to be reliable.

A representative gate array is a Raytheon RA-116. It has 300 Schottky TTL gates, of three configurations:

120 internal driver gates (3-input NAND)

60 external driver gates (4-input NAND)

120 internal expansion gates (7-input NAND

or

2-input OR expanders)

The gates have a typical propagation delay of 5-6 nanoseconds and dissipate 5.5-6 milliwatts per driver and 1 milliwatt per OR expander. Two metal layers are used for tailoring; there are 64 external pins.

Since the designer can arbitrarily interconnect, he constructs flip flops, adders, decoders, etc. Because the primitive is recent, data on package count reduction is scarce. One informal study we know of (for the Raytheon RP-16 aerospace computer) had a measured factor-of-three improvement.

¹In the most recent Cray design there are only three IC types used: two are memory configurations and the third is a set of gates that are interconnected using the printed circuitry of the module and backplane. Thus, while only 3 IC types have to be stocked there is still a large number of module types, and

G. Bell

created 1/4/78, latest edit 2/13/78

For higher speed applications an ECL gate array has been proposed. The ULG exploits the inherent properties of current mode logic to obtain a particularly flexible element (Gaskill et al., 1976].

The most common form of intra-IC design is custom design, although considering the designer, it is a variant of the standard cell since a particular designer has a set, even though he may not have enumerated all the design components a priori. With custom design the designer (theoretically) can specify a circuit for each use within a particular logical design. In this way, by observing that a particular gate or flip flop only drives a certain load, the circuit can be modified to carry out just the desired function in the context of its use. Therefore, with custom design, the whole circuit can theoretically be of optimum since each part is no larger than need by. The advantages are clearly size, cost and speed. The design costs are high because each part can, in principle, be customized. The goodness of the circuit is totally dependent on the single designer who must analyze each circuit geometry in terms of his expectation of performance, operating margins, etc. To the extent that this analysis is carried out, the circuit is clearly good. Design of this type rarely occurred in the first few generations; i.e., both circuit and logical design are varied for the context.

Also on the graph is a hypothetical line for universal logic arrays. For at least 15 years, academicians have studied the possibility of designing a single array or arrays of arrays of logical design elements that can be interconnected on a custom basis to carry out a given function. The gate array can be looked

¹In the most recent Cray design there are only three IC types used: two are memory configurations and the third is a set of gates that are interconnected using the printed circuitry of the module and backplane. Thus, while only 3 IC types have to be stocked there is still a large number of module types, and

G. Bell

created 1/4/78, latest edit 2/13/78

at as the simplest example of this type design. While we are skeptical that such a line exists, we put it here as a target for those who search for the one truly universal logic array to aim at.

Both Read Only Memory (ROM) and Programmable Read Only Memory (PROM) are trivial forms of the truly universal arrays because they can be used in a table look up fashion to several functions of a number of input variables. For example, a 1,024 word ROM arranged in a 256 x 4-bit fashion can generate 4 independent functions of 8 variables. This is a distinct alternative for using a conventional gate structure to carry out combinational functions. Therefore we consider the ROM to be a common form of customization that nearly all users engage in.

However, a disadvantage of table look up implemented by ROM is that the required ROM size doubles for each additional input variable. The PLA is a combinational circuit which remedies this by allowing the use of product terms rather than completely decoding the input variables. Fig. PLA shows a typical circuit. It consists of separate AND and OR arrays; inputs are connected to the AND array, and outputs are drawn from the OR array. Each row in the PLA can implement an AND function of selected inputs or their complements, thus forming a boolean product term, and the OR array can combine the product terms to implement any boolean function.

A simple application is operation-code decoding. For the PDP-11, the 16-bit Instruction Register could be directly connected to a PLA, and the output

¹In the most recent Cray design there are only three IC types used: two are memory configurations and the third is a set of gates that are interconnected using the printed circuitry of the module and backplane. Thus, while only 3 IC types have to be stocked there is still a large number of module types, and

G. Bell

created 1/4/78, latest edit 2/13/78

function would be a microprogram address. Three different types of interpretation are usually needed: source mode decoding, destination mode decoding, and instruction decoding. Three PLAs could be used. A ROM version would require 64Kx8 for instruction decoding and 128x8 for address mode decoding.

With 2Kx8 ROM's this would require 33 packages as opposed to the 3 PLA packages. The 11/34 CPU uses a PLA in this fashion.

Microcomputers, e.g., the LSI-11 (Chapter) commonly use the PLA technique for conserving the die area used by their control units.

The PLA becomes an even more feasible building block when it is made field programmable -- the FPLA. The programmable connectors shown in Figure PLA are fusible nichrome links.

When a register is added to the outputs of the PLA, ~~as with the Signetics 82SXXX,~~ and incorporated in the same integrated circuit, a simple sequential machine is obtained in one package. ~~This greatly increased the package count reduction potential of PLAs.~~ Since register circuits are pin intensive, a factor-of-two package reduction occurs as soon as i.c. complexity allows the registers to be incorporated. ~~The same reduction factor occurs when a ROM chip is augmented by a register.~~

count of PLA implementations still further.

into the PLA chip, thus reducing the package
a similar

The first PLAs had propagation times of the order of 150 nanosec [check with

¹In the most recent Cray design there are only three IC types used: two are memory configurations and the third is a set of gates that are interconnected using the printed circuitry of the module and backplane. Thus, while only 3 IC types have to be stocked there is still a large number of module types, and

Spencer] and were thus suitable building blocks for slow, low-cost computers. Propagation times of 45 nanoseconds are quite common today, and the PLA is more widely used. A candidate application with these higher speed components is the replacement of the SSI and MSI packages used to implement the control logic for Unibus arbitration.

A more complex application than IR decoding has been documented in (Logue et al., 1975]. An IBM 7441 Buffered Terminal Control Unit was implemented using PLAs and compared with an SSI/MSI version. The PLA design included two sets of registers fed by the OR array: one set fed back to the AND array, and the other set held the PLA outputs. A factor-of-two reduction in printed circuit board count was obtained with the PLA version. The seven PLA's used in the design replaced 85% of the circuits in the SSI/MSI version. Of these circuits 48% were combinational logic, and 52% were sequential logic.

MEMORY TECHNOLOGY (AND SEMICONDUCTORS USED AS MEMORIES)

In the preceding section we observed how core memory evolved and was challenged several times by alternative technologies and then finally overthrown in its use as the computer's primary memory. We also observed slightly unconventional semiconductor memory use for microprogramming and table look up in logical design. This section will not discuss these uses, but will present the various parameters and discuss the use of memory within a hierarchy to store information both on a short term basis while a program runs and on a longer term basis as permanent files. The slower speed electromechanical memories

G. Bell

created 1/4/78, latest edit 2/13/78

including drums, disks and tapes will be considered superficially, as their performance and price improvements have pushed the computer evolution.

The Parameters Determining Use

We try to have a small number of parameters for memory because it is the simplest of components. Since memory prices have declined at a combined rate of 30% per year compound, (which amounts to over 50% in two years), all parameters are time variants. While the sole single metric is price/bit, it is mandatory to know the price (or size) of the total memory system because there is an economy of scale. In order to get the lowest price per bit, a user may be forced to a large system. (Is there economy of scale? What is the smallest or largest memory that can be built with the technology?)

Performance for electromechanical memories is expressed in two parameters: the number of bits that can be accessed per second after a transfer begins; and time to access the start of a block of memory. Since most all memories we're interested in here can be both read and written, the concern will not be on using write once or read only memory such as video disks.

The operational environmental parameters of power consumption, temperature, space and weight effect the utility. Finally, reliability measures are needed in order to see how much redundancy must be placed in the memory to operate at a given level of availability and likelihood of losing information.

In summary, the relevant parameters for a given memory are:

G. Bell

created 1/4/78, latest edit 2/13/78

-
1. the time the observation is made together with a time history in order to deduce the location of the technology in its lifetime;
 2. price per bit; and
 3. total memory size or total memory price;
 4. performance has two components:
 - a. the access time to the first word of the block
 - b. the time to transfer each word (data-rate) in the block
 5. operational power, temperature, space, weight
 6. reliability and repairability in order to compute availability and the maximum

For this discussion we will consider only the first four parameters (and often omit no. 3.). In the solely core and semiconductor memories 4b is omitted, whereas for electromechanical memory (e.g., disk, tape) the performance is usually determined by 4b (and 4a can be omitted).

Before we observe how the use of memories has affected the evolution, it is important to posit the various technology models as a basis of the past and navigational aids to the future. Figure Memprice gives the expected price of semiconductor memory. Namely, these memories decline in price every two years

G. Bell

created 1/4/78, latest edit 2/13/78

by a factor of two and the decline is expected to continue well into the 80's based on continued increases of semiconductor densities. Figure Memsizeperf, a graph by Dean Toome, vice president of engineering for Texas Instruments, shows memory size and performance improvements with time and includes the slower perform cyclically accessed Charged Coupled Devices (CCDs) and magnetic bubbles.

While these two graphs allow the processor, primary memory, cache and small paging memories to be understood, we need a model for disk and tape memory in order to complete the memory hierarchy. (I'd like graphs here of:

- . Price/byte for 1 platter, 4 platters, 10 platters
- . Price per disk for 1 platter, 4 platters, 10 platters
- . Tape units at each speed 12/45/75/125/200
- . Tape density and transfer rate vs time for each speed
- . Price/byte of small systems: DECTape, floppy, cassettes

Computational Locality and Memory Hierarchies

All information processing systems (computations) appear to be associated with relatively local memory use; as time passes, the location of the computation changes, but still the amount and the location of memory involved in a

computation over a relatively long time, is comparatively small.

This phenomenon has been observed to apply to a number of activities within a computer system, and in each case the principle of operation appears to be the same. Thus, armed with the observation, a new memory--providing proportionally more memory for less, should be useful if it is:

cheaper and slower than an existing memory; or
it is faster and more expensive (implying that it is smaller).

Of course, any memory that is better in all dimensions of speed, size and cost automatically replaces an existing memory.

Transparency is another consideration about the utility of a given physical memory within a hierarchy that effects use. Namely, does the program(mer) know or have to consider the memory in his use. If we observe the machine at the language level, then nearly all memory technologies are transparent. For example, the original FORTRAN had explicit statements to read and write tapes where the modern counterpart has operations to access files in different access arrangements; hence, these operational improvements are not quite transparent. However, whether a processor has no, 1 or 16 accumulators is irrelevant to the FORTRAN programmer.

The following table gives the memory hierarchy as currently known. There is a continuum based on need together with memory technology size, cost and

G. Bell

created 1/4/78, latest edit 2/13/78

performance parameters.

Table: Computer System Memory Component and Technology

Part	Transparency (to machine language programs)	Based-on
Microprogram memory	yes ¹	very fast
Processor-state	no	very small, very fast register set (e.g., 16 words)
Alternative processor- state context	yes	same (so speed up processor context swaps)
Cache memory	yes	fast. used in larger machines for speed
Program mapping and segmentation	yes	small number of association, or large map
Primary (program) memory	no	relatively fast, and large depending on Pc speed
Paging memory	yes	can be electromechanical, e.g., drum, fixed head disk, or moving head disk. Can be CCD or bubbles.
Local file memory	no	usually moving head disk, relatively slow, low cost
Archival files memory	yes (preferably)	very slow, very cheap to permit information to be kept forever

Nearly every part of the hierarchy can be observed in the computers in this book. To begin, microprogrammed memories are normally transparent to the

G. Bell

created 1/4/78, latest edit 2/13/78

general registers in an explicit (transparent) fashion, or whether the stack architecture should be used (also requiring a number of registers), but the use is transparent to the programmer. We adopted the general register structure to give better program control of a small number of local variables (i.e., locality) and permit the performance advantages. The change in register use can be observed between the 12-bit and 18-bit designs and the later DECsystem 10 and PDP-11 designs.

As the number of registers increased, and technology improved the processor state was increased to have multiple sets of registers to improve process context switching time. In this way several multiple programs could be active at a time and selected on the basis of interrupt urgency to provide better real time and multiprogramming response.

In the late 60s the cache memory was introduced for large scale computers. This concept was applied to the KL10 and 11/70 in 1975 when the relatively large, (1 Kbit) relatively fast (factor of 5) memory chip IC was introduced. The cache is discussed extensively in chapters 00, 00 and 00. It derives much power by the fact that it is an automatic mechanism and hence transparent to the user. It is the best example of the use of the principle of memory locality.

A similar memory circuit is required to manage (map) multiprogrammed systems by providing relocation and protection among various user programs. The requirements are similar to the cache and may be incorporated in the caching

G. Bell

created 1/4/78, latest edit 2/13/78

structure. The KI 10 used an associative memory for this mapping function.

The use of semiconductors for the primary memory, replacing core memory is legendary. Section 00 discusses this evolution and tradeoff.

The Atlas computer (Kilburn et al 1962) was designed to have a single, one level large memory and paging drum. This structure ultimately evolved so that multiple users could each have a large virtual address and virtual machine. However, the concept of paging mechanism works because there is not equally random access to each page, but rather only local access to various parts of a program by the processor at a given time. Denning pointed out the clustering of pages for a given program at a given time and called this phenomenon the working set [19]. For most programs the number of pages accessed locally is small compared with the total program size. Initially a magnetic drum was used to implement the paging memory, but as disk technology began to dominate the drum both fixed head and moving head disks (backed up with larger primary memories) were used as the paging memories. In the next few years, the relatively faster and cheaper CCD semiconductor memories and bubble memories are clearly the candidates for paging memories.

For medium sized to large scale systems there is no alternative to disks, with archival files on magnetic tape. Thus files can be stored cheaply on an indefinite basis. For smaller systems there must be less numbers of technologies due to high overhead of a level. At most, two levels would probably exist as separate entities.

G. Bell

created 1/4/78, latest edit 2/13/78

Alternatively we would expect a combination of floppy disk, low cost tape and magnetic bubbles to be used to reduce the primary memory size and at the same time provide file and archival memory. Currently the floppy disk operates as a single level memory. Here we can see two alternatives for technology tradeoff using the hierarchy: a tape or floppy disk can be used to provide removability, and archivability, whereas bubbles provide the performance.

MEASURING (AND CREATING) TECHNOLOGY PROGRESS

The previous sections have presented technology in terms of exponentially decreasing prices and/or exponentially increasing performance. Here we present a basis for this constant change. The metric of technology, $T(t)$ at some time, t , has been classically observed to be just:

$$T(t) = K \times e^{ct}$$

This can be converted to a yearly improvement rate, r , by changing the base of the exponential to:

$$T(t) = T \times r^{t-t_0}$$

where T = the base technology at t_0

r = yearly increase (or decrease) in the technology metric

This is the same form we have used for declining (or increasing) cost from base

G. Bell

created 1/4/78, latest edit 2/13/78

c

$$C = c \times r^{t-to}$$

The questions that may arise as we observe the previous graphs (i.e., semiconductor prices) are:

1. Under what conditions does cost decrease exponentially?
2. Under what conditions does technology improve (i.e., in performance) exponentially?
3. When does a technology reach a limit of improvement?

Clearly there are manufactured goods that neither improve nor decrease in price exponentially, although many presumably could with the proper design and manufacturing tooling investments. The notion of price decline is completely tied to cumulative learning curves both by people building a product for a long time, to process improvement based on learning to build it better and learning by engineers based on history of design. Production learning per se is inadequate to drive cost and prices down because after an extremely long time in production, a few more units contribute little to learning. With inflation in labor costs, then costs actually when the learning is flat. In order to provide us with a base for predicting the inflationary effect, the consumer price index has been plotted Figs. CPI and CPI.log.

G. Bell

created 1/4/78, latest edit 2/13/78

Learning curves don't appear to be understood beyond intuition. They are (empirical) observations that the amount of human energy, E_n , required to produce the n^{th} item is:

$$E_n = K \times n^d$$

where K and d are "learning constants". Thus, by producing more items, the repetitive nature of a task causes learning and hence the time and perhaps cost to produce an item decreases with the number produced and not with the calendar time an object is produced. Fusfeld [] furthermore conjectures that the technology of the i^{th} unit produced also improves exponentially with the number produced just as in the case of the learning curves. Thus, using the technology measure:

$$T_i = a \times i^b$$

he found the following technology progress constants:

Item	Measure, T_i	Quantity produced (i)	Technology progress (b)	Change observed in study	Total change
light bulbs	lumens/bulb	10^{10}	.04;.19	33	80
automobiles	vehicle h.p.	$3 \times 10^7; 10^8$.11;.74	10	6; 13
titanium	p.s.i./\$/16	3×10^8	.3; 1; 1.04	10	350
aircraft	max.speed	2×10^5	.33-1.2	6	56
turbojet engines	fuel consumed, weight	1.6×10^4	1.06	2	2.9×10^4
computers	mem.size x rate	10^5	2.51	10^9	3.5×10^{12}

Here we note that computer technology (till 197?) evolved substantially more

G. Bell

created 1/4/78, latest edit 2/13/78

than any other technology. Recent advances suggest this has continued. This in part because there are so many materials to store, transmit and process information nearly all of which are electronic based. In the above technologies, most are mechanically oriented with the associated physical limits. In essence we are comparing systems constrained by Newton's Laws with those determined by Maxwell's Equations.

Fusfeld also showed that provided the number of items produced increases exponentially (with time):

$$i = e^{c/b \times t},$$

then calendar time and units produced can be used interchangeably.

Since there has been exponential growth of computer systems, calendar time can be used instead of units. Time is an easier and more accurate method to measure than learning curves based on units.

The question of why the cost declines exponentially can be conjectured by using Fusfeld's observation that it is because of learning curves and the exponential increase in quantity produced. Furthermore, this exponential increase raises a fourth question;

4. Why is the demand exponential?

G. Bell

created 1/4/78, latest edit 2/13/78

The demand or quantity, q , sold per unit time is completely elastic, (exponential), according to the expression:

$$q = k/\text{price}$$

This creates a positive feedback market system whereby decreasing prices increase demand exponentially causing decreased costs (through learning) which support the decreasing prices as shown in Fig. Mkt.cycle (a variant of Fig. 1.).

There has been no attempt to answer question 2 of why technology improves exponentially nor is the answer for why cost declines exponentially at all satisfactory. Simple production learning does not account for the rapid technology changes in integrated circuit for example where totally different production processes have been evolved to support the greater technology. It appears best to simply observe that the three situations have been true, and can be extrapolated to hold over the next few years because we can see ways by which each limit can be overcome.

Technology historians (von Hippel, 197?, and Fusfeld 197?) and futurists have made a number of studies and observations about technology progress:

1. Quantity produced is simply a good dummy variable to measure improvement.
(With exponential increasing sales, we can use time as the dummy variable.)

G. Bell

created 1/4/78, latest edit 2/13/78

-
2. Technical problem solving is correlated with business activity. Inventors tend to be stimulated by sales and slacken efforts when sales are low.
(This is a counter-initiative observation.)
 3. Production alone does not stimulate innovation. A lesser number of inventions are stimulated by production needs. Of these, the same user-supplier relationship is the best framework. The users of equipment (the producer for the end product) stimulate the production equipment suppliers.
 4. Major innovations come from use (sales). This is opposed to innovation arising from a technical possibility for which use is subsequently discovered. In the case of semiconductor technology, the computer designer (user) is responsible for many of the design innovation. Computers also evolve rapidly under this and the levels of integration model because the suppliers are also significant users. The problem of market coupling is diminished significantly. Alternatively with computers, the users write (develop) the applications programs; hence again the user and developer are one in the same--this stimulates use!

The gains in packaging (i.e., printed circuit boards, backplanes, and cables) have been driven by production, versus product functional needs. Testing technology of all types has been motivated solely for production needs. These change least.

G. Bell

created 1/4/78, latest edit 2/13/78

The cost of computing is the sum of costs which correspond to the various levels of integration we described in view 4, page 00 plus the operational costs. In actual practice, each layer, or additional level-of-integration is often looked at as overhead. Using standard accounting practice, the basic hardware cost, at the inner layer, is then multiplied by an overhead factor at each subsequent outer level. While this may in principle work operationally, for a stable set of technologies it is hard to condone and we will describe the constituent technologies and resulting structures at the next level together with operational cost models. An overhead-based model will not adequately allow for rapidly evolving technologies or the elimination of levels, for example. By examining each part, as we have tried to do in this section on technology and in the packaging section, we can then make observations about the use and substitution of technology. More importantly, we can draw conclusions about how structures are likely to evolve.

Semiconductor technology is singled out as the main determinant of a computer's cost, performance, reliability and memory size. Magnetic storage technology is of equal importance because disks and tapes memory densities evolve rapidly, even though the cost of a given physical unit does not. These components become an increasing major component of the system price.

Cost, Performance and Size (Economy of Scale) Relationships Design

We often simplify the understanding of technology to just cost as the only dependent performance parameter. This simplification omits the most

G. Bell

created 1/4/78, latest edit 2/13/78

significant parameter, calendar time. In a similar way, we often fail to understand whether there is any associated economy of scale when considering the performance (utility) of a given set of devices. That is, do larger units perform significantly better than a set of small units when taken either independently or as a collection? Therefore, a more correct assessment, taking into account economy of scale, and performance, would be:

$$\text{performance} = k \times \text{costs} \times r^t$$

where

k = base case performance

s = economy of scale coefficient

r = rate of improvement of technology

t = calendar time

For many of the technologies in which we're interested, a second dependent parameter, size, is important because it is a measure of utility. It can, with a tradeoff, be a measure of performance but often the two must be taken as independent resources. For example, a certain memory size would permit a certain number of accesses. For some applications, we would ask whether the size were adequate and for other applications we ask whether there is adequate time to access the data. Thus the systems performance in such an example is bottlenecked by either size or access and is the minimum of one of the resources. That is:

$$\text{system performance} = k \times \min(\text{memory-size}, \text{memory performance})$$

G. Bell

created 1/4/78, latest edit 2/13/78

Figure Perfsize shows the various options of the amount of economy of scale:

1. Economy of scale holds. A particular object can be implemented at any price, and the performance varies exponentially with price.

$$\text{performance} = k \times \text{price}^s; s > 1$$

2. Linear price performance relationship.

$$\text{performance} = k \times \text{price}$$

3. Constant performance, no matter how much is spent.

$$\text{performance} = k$$

4. & 5. Only a particular device has been implemented. The performance (or size) is a linear sum of such devices.

$$\text{performance} = n \times (k \times \text{price})$$

Economy of scale holds where it might not otherwise operate because of several effects in the design and product introduction strategy. Because a product is already high priced, adding slightly more cost may have a proportionally higher effect on performance than for lower cost products where the same constant cost addition may be needed. This is akin to the design styles, page 00. This condition is especially true in disks and computer systems. A technique (e.g., a particular recording method employing costly logic for encoding/decoding, the cache memory) is employed at the high priced system first. For example, the cache memory is nearly constant cost, independent of the size machine it is

G. Bell

created 1/4/78, latest edit 2/13/78

used on. With time, and learning, the technique can then be applied to lower cost (e.g., the 360/85 in 1968 and the 11/70 in 1975) systems.

In Fig. Costvstime, we see what typically happens as the cost of the lowest price unit is kept to a minimum and decreasing whereas the mid range product continues to increase, and the highest performance product increases the most, because it can afford the overhead costs. If we look at the basic technology metric, then there are really three curves too as shown in Fig. Techvstime. The first curve is applied to get the greatest improvement and be applied to the large price unit. With time the technology evolves and is reapplied to the first level copy in the middle range products (to most likely provide the best cost performance) and finally, several years later the technique becomes commonplace and is applied on low cost products. The resultant cost/performance ratios are shown in Fig. Cost/tech.

In most industries, the management of technology is generally nil because no industries evolve like computers (and semiconductors). The computer industry is fundamentally driven by the semiconductors technology push on the one hand, and by IBM on the other. IBM follows the strategy of applying technology on an economy of scale basis. This permits the technology to be first tested at the high performance, high price lower volume systems before being introduced in higher volume production. The following examples (from IBM) show this at work: printing: dot matrix printing, chain printing, and Ink Jet printing; computer printing flexible disks as precursor to use as systems products using xerography, magnetic storage: basic technology for large disks as precursor to

G. Bell

created 1/4/78, latest edit 2/13/78

use on smaller disks, CPUs: the cache memory; the wide tape helical scan tape unit microprogramming for language and operating system performance improvement; (and possible user microprogramming), programming languages: APL, semiconductor memories, communications: networking and SDLC protocol.

Unsuccessful Products Can be Stopped

magnetic storage: large diameter disk files, the data cell, optical storage, programming languages: PL/1.

Technology Substitution

Since each constituent technology evolves at its own rate, hence the cost and performance of a system is roughly the additive and multiplication functions, respectively, of the parts. Usually when one component begins to dominate (e.g., packaging), then pressure occurs to more rapidly change and improve the technology to avoid the cost or performance bottleneck. Sometimes a slowly evolving technology is just eliminated as a substitute is found.

Some of the substitutions that have occurred:

1. Semiconductor memories are used in place of core memories. Since the latter has evolved more slowly in terms of price decline, semiconductors are now used to the exclusion of cores. (This has not occurred where information must be retained in the memory during periods of time without power.)

G. Bell

created 1/4/78, latest edit 2/13/78

-
2. Read-only semiconductor memories are substituted for semiconductor logic elements. Mudge explains this tradeoff in chapter 00. Using microprogramming, table (logic function) look up, and sequenced (vs parallel) processing, this substitution is quite natural. For combinational logic, it has been estimated that a single logic function of 1-3 variables is equivalent to 5-7 bits of a random access memory.
 3. In a similar way PLAs can be potentially substituted for ROMs and true content addressable memories can replace various read write and ROM memories. Other examples of substitution occur as natural parts of the memory hierarchy as described in the earlier section.

By using the principle of locality smaller, faster, more expensive memory can be introduced to replace intermediate speed, and moderately priced memory as in the cache.

4. The judicious use of CCD memory can cause drastic reduction (and quite possibly the elimination) in MOS random access, for primary memory. Note the fixed head disk is eliminated at the same time.
5. We conjecture that for small systems, the main operational memories could be completely non-electromechanical; electromechanical memories (e.g., tape cassettes and floppies) are used for loading files into the system and for archives. For lower cost systems, semiconductors ROMs replace cassettes and floppies for program storage.

G. Bell

created 1/4/78, latest edit 2/13/78

At a given time the slowly improving (or increasing) costs, like the packaging and power, may become a significant fraction of the total cost. Costs are additive and hence exponential improvements have disproportionate effects causing pressure for structural change.

For instance, although the PDP-8 is normally considered to be the first minicomputer, it post dates the CDC 160 (1960) and DEC's PDP-5 (1963).

However, the PDP-8 was unique in its use of technology to:

1. It eliminated the full frame cabinets used by other systems. This also presented a new computer style such that users could embed the computer in their own cabinets. A separated small box held the processor, memory and many options.
2. Automatic wire wrap technology was used to reduce printed circuit board interconnection cost. This also eliminated errors and reduced check out time.
3. Printed circuit board costs were reduced by using machine insertion of components.
4. The Teletype ASR33 (also used on PDP-5) was connected as the peripheral. It had a combined printer, keyboard, and paper tape i/o device (for program loading). It eliminated the paper tape reader and punch.

G. Bell

created 1/4/78, latest edit 2/13/78

Tradeoff Between Size and Performance

A memory system might permit an encoding to be used such that memory accesses could be traded for memory size. This is about the only example where such a condition might exist. This condition is shown in Fig. Tradeoff for 3 points. Here, the effective memory size can be increased or reduced by either using memory system accesses for computation or for storing precomputed values. This tradeoff permits the operating region for the system to be greatly extended to either include the memory (by giving up 1/2 of the processor or 2.5 times the amount of processing/by giving up 1/2 the memory).

The Effect of the Research, Advanced Development Process of the State of the Art Line

The complete development process is pipelined as shown in Fig. Dev.process. Here we note that ideas, theoretically flow through the various organizations in a process-like fashion, culminating in a product. Each product type has a different set of delays associated with a part of the pipeline and at the end, the education of use (also a delay) culminates in market demand. For well defined, commodity-like products (e.g., disks and primary memory) the delay is zero as each user "knows" the product; for a new language (e.g., PL/1) the delay is slow to build, and a new product may even eliminate one that potentially would create a demand. The following table shows the times for various technologies: disks, computers, more software (e.g., language), and semiconductor process.

G. Bell

created 1/4/78, latest edit 2/13/78

In order to understand the process, let us describe the disk supply process. The technology (as measured by the number of bits per areal inch) doubles every two years (i.e., the density improves 41% per year). IBM is estimated to invest about 100 million dollars per year in the pipeline and associated pipeline for the manufacturing process, not counting any improvements in semiconductor technology (semiconductors also influence the manufacture of disks). In this way, the IBM disks essentially establish the state of the art line in a structure that is typified by Fig. Techvstime. Note there are two other lines delayed, each delayed about two to three years. Using the obvious process, to build products competitive disks would lie somewhere about four to six years behind the state of the art line. This can be seen in Fig. Disk.delay and by looking at the Fig. Dev.process and taking into account the delays through each stage. In order to be more competitive, the disk industry short circuits the various delays as it engages in reverse engineering; this results in only two year lags. In reverse engineering, the tools are micrometers and reverse molds. At first ship of a new product, the product is purchased and basically copied on a function per function basis. The more successful designs use pin for pin compatibility so as to take maximum advantage of design decision.

From the process, it is also easy to see how by merely copying competitive products, one has built a process that guarantees products will be two to four years behind competitor products and most likely sub state of the art. This structure is most likely the result of having a strong market function which operates to define products based on existing product use. If the design and

G. Bell

created 1/4/78, latest edit 2/13/78

manufacturing process is quite rapid, then such a strategy can be effective. This process can be very effective for software products because there are no delays associated with manufacture and the time to learn about the products are long providing a window in which any competitive pressure is likely to be slow to build. However, software engineers are the most creative and most likely to ignore any previous work in an area.

A high technology, exponentially increasing (volume) product is denoted by:

1. Exponential yearly cost improvement (price decline) rates through product technology improvements as measured by price decline of $> 20\%$ (e.g., disks = .8, cpu = .79, memory = .7).
2. Short product life < 4 years.
3. Various types of learning curves. Some products have negligible manufacturing learning and some denote forgetting through turnover, increased training to meet demand and inflation.

The Product Problem (Behind the State-of-the-Art)

Typical "product problems" can be seen in Fig. Product.cost.

G. Bell

created 1/4/78, latest edit 2/13/78

Fig. 1, Product.cost problems/timing problems.

We have these possible situations:

1. ideal (on the state-of-the-art line)
2. advanced (moves below the line)
3. late (slip in time to the right)
4. expensive (more than expected in cost, straight above the line)
5. late and expensive (to the right and above the line)

Situations 3, 4, and 5 are product problems because they are behind the state-of-the-art line and hence less competitive. This implies increased sales costs, lower margins, loss of sales, etc.

Note that a late product could be OK if somehow the cost were lower. Similarly an expensive product is OK if it appears earlier in time.

Time is Money (and vice versa)

Thus product problems can be solved by either:

1. movement in time (left) to get on the line; or
2. movement in cost (straight down) to get on the line

G. Bell

created 1/4/78, latest edit 2/13/78

since

c = cost at time, t (in years)

b = base cost

r = rate of price decline

therefore, with exponential price declines a family of products over a long time will follow a cost curve, c .

t

$$c = b \times r$$

now

dc = change in cost above (or below) to get back to the state-of-the-art

dt = delay (or advance) in time to get back to the state-of-the-art line

let

$f = dc/c$ = fraction (%) of cost away from line

dt

$f = 1 - r$ poor cost, expressed as

G. Bell

created 1/4/78, latest edit 2/13/78

project slip

and

$dt = \ln(1 - f) / \ln(r)$ poor timing,
expressed as poor
cost

These formulas let us interchange time and money (cost).

For example, in disks or cpu's where $r = .8$ and $\ln .8 = .22$

dt

1. $f = 1 - .8$

2. $dt = - 4.45 \times \ln(1 - f)$

(using 1.) A 1 year slip is = to a 20% cost problem

(using 2.) A 10% cost increase is = to a .47 year slip

Who does what, and their effect

By and large engineering, by establishing the product direction has the

greatest effect on the product. However, since most product problems may have multiple components it's worth looking at each.

1. Timing

- a. Engineering schedule slips simply translates into a competitive cost problem (or it can be expressed or left alone) as simply a sub state-of-the-art, late product.
- b. Manufacturing - ramping up the learning curve quickly by risk taking has a high payoff when considering the apparent cost or delay.

2. Cost (see Fig. Net)

- a. Engineering is perhaps the major determinant of cost by the way product design - number of parts, ease of assembly, etc. The most common cost problems occur by continued product enhancement during the design stage providing increased functionality.
- b. Manufacturing - direct labor and overhead really count. Making major product moves with decreased learning (increased forgetting) all serve to stretch the product time out and put pressure on getting newer products. One curve shows the direct costs associated with manufacturing assembly and then some learning should take place as long as product volumes increase exponentially. New technology materials

show the greatest cost improvement for computers--here we assume semiconductors and other electronic materials improve with time. Note that by capital equipment investment (tooling), there can be stepwise cost decreasing in materials.

- c. Inflation - while not a direct cost function, it combines with labor cost to negate learning effects.
- d. Note the costs are taken altogether. In terms of a sub state of the art product, the costs are compound. A one year late, 10% overcost product has the effect of being about 1-1/2 years late or about 30% too expensive.

3. Manufacturing learning

Learning curves and forgetting curves really matter. Left alone, a typical product may:

t

- 1. $c = b \times .95$ decrease as little as 5%/year.
- 2. $c = b$ = stay constant with little attention.

t

- 3. $c = b \times 1.06$ = increase with inflation.

The effect of each is shown in Fig. Product.price:

G. Bell

created 1/4/78, latest edit 2/13/78

Fig. 2, Product.price problem at time, T after introduction.

Mid Life Kicker for Product Rejuvenation

We can by enhancement (the so called mid-life kicker) improve the cost/performance metric of a given product. This is non-trivial, and for certain products must be inherent (i.e., designed in). Under these conditions, improvements in cost go immediately to get the product back on the curve.

For example a factor of 2 in performance halves cost/performance. The effect, in time, to double the density of a disk is to move the product back to the state-of-the-art line. I.e.,

$$dt = 4.45 \times \ln(0.5) = 3.1 \text{ years!}$$

This situation is shown in Fig. Product.improve and is compared with a 5%/year learning curve.

G. Bell

created 1/4/78, latest edit 2/13/78

Fig. 3, Product.improve product cost improvement by enhancement of
cost/function.

G. Bell

created 1/4/78, latest edit 2/13/78

PACKAGING

Figure 1 suggests that packaging is a completely recursive process where boxes (packages) are placed within boxes (packages) on an indefinite basis. Packaging is iterative, not recursive, because for each level (IC, module,...cabinet, room) of packaging, a completely different design problem exists. However, for a computer system consisting of a set of components, each component can be thought of as a system itself which in turn is decomposed into lower level components. The computer systems we describe herein can be nicely decomposed into three to five distinct levels. At the topmost level these physical components correspond to the PMS level components from which the final computer engineer (the user/buyer) conceptualizes and builds the computer.

Packaging the electronic and electromechanical components that constitute a system at a given level consists of: interconnecting the set of the components via signal carrying links and holding (packaging) them within some basically mechanical structure such that as a unit system they carry out the objective function (i.e., meet the design specification). Figure 2 introduces a few of the packaging problem subtleties. There is some visual effect in addition to the function for which it was designed. The visual effect may, in fact, be the dominant reason (or deciding factor) in whether the object is purchased. While we are only interested here in what is basically electronic signal processing there are two side-effects: power is consumed and there is an electromagnetic interface (EMI). Power, in

G. Bell

created 1/4/78, latest edit 2/13/78

turn, requires cooling or heat dissipation to some outside environment by convection, radiation or conduction so that the system will ultimately not burn up...or malfunction when stored or operated outside a limited temperature range. The EMI interface is bi-directional: signals emanate from the package and must be controlled in order to avoid interference with the radio frequency spectrum; and other equipment broadcasts radio frequency interfering (RFI) signals which must not interfere with the package's function. In addition to cooling (or heating) other environmental constraints include humidity, air quality (dust, corrosiveness, salinity, etc.).

Seymour Cray*, the designer of the CDC 6600 (12/64) and 7600 (12/69) computers and most recently the Cray 1 (12/75) computer, described packaging as the most difficult part of the computers designer's job. The two major problems are heat removal and the thickness of the mat of wires which cover the backpanel. His rule of thumb indicates that with every generation of large computer, the size decreases by roughly a factor of five--and each generation takes roughly five years.

While it is easy to understand why one of Cray's super computers is so dominated by packaging, we want to examine the effect of packaging on small computers. At one extreme, we observe that the first hand-held scientific calculator, the HP35, was simply a new package for a common object, the calculator--which has been with us about 100 years.

*Lawrence Livermore Laboratory lecture in December 1974.

G. Bell

created 1/4/78, latest edit 2/13/78

Although semiconductor density was high enough to permit only a few integrated circuits to be necessary to implement a calculator, it wasn't until they were re-packaged in a particular fashion that the hand-held calculator came into existence. Currently this embodiment is synonymous with the calculator name--in the future, the calculator might take on some other form (e.g., watch, pencil, voice actuated, hearing aid, notebook).

Packaging also seems to be the dominant reason for the PDP-8 and minicomputer phenomenon--although marketing, the coining of the name, and the ability to easily manufacture (also part of packaging) are alternative explanations. We believe the PDP-8 was the original minicomputer, yet its antecedent PDP-5 had roughly the same instructions set, was about the same size for a complete system with peripherals. Since the PDP-8 was packaged in a separate box which could fit into roughly 1/3 of a standard relay rack cabinet, it was small enough to be incorporated into other systems on a dedicated basis. Thus, it was perceived and used as a component to build other larger systems.

The Packaging Design Problem

Packaging is the complete design activity of interconnecting a set of components together via a mechanical structure in order to carry out a given function. In order to package a large structure such as a computer, the problem is further broken into a series of levels each with components that carry out a given function. Figure 3 shows the hierarchy of levels that have evolved these last twenty years for the DEC computers. There are

G. Bell

created 1/4/78, latest edit 2/13/78

eight levels which describe the component hierarchy resulting in a computer system. For example the transition from the second to the third generation can simply be described as adding an intermediate level to the hierarchy, which in turn minimized the need for additional circuitry at the discrete and integrated circuitry (IC) level. That is, there was a transition of taking hardware that existed at one level and agglomerating at a lower level.

For each packaging level there is a set of interrelated design activities as shown in Fig. 4. The activities (or disciplines) are almost independent of the level they are carried out, although a design activity (e.g., logical design) is often carried out (i.e., partitioned) across several levels. For some levels a design activity doesn't exist. For example, since integrated circuits operate at low voltages, there is not a safety design problem associated with IC chip design. However, since there are many circuits on a single IC and each can be operated at a special voltage, there are both power supply design and power dissipation (cooling) problems associated with IC design.

We first note (Fig. 4) that the initial design problem is to simply perform a certain function. Furthermore, performing the function causes a number of side effects that have to be solved. For example, the integrated circuits and other equipment that do information processing require power to operate. The power creates a safety hazard; and since each power supply operates at less than 100% efficiency, heat must be carried away from the

G. Bell

created 1/4/78, latest edit 2/13/78

power supply and the components being powered. In this way the cooling problem is created. Sometimes cooling is carried out using conduction to an outside surface so that it may be carried away by the air in a room. Mostly cooling problems are solved by convection with a fan in the cabinet by carrying air into the room so that the room is left to cope with carrying the heat away. In the process, the fans create acoustical noise pollution in the room, making it harder for humans to cohabit the room. If the computer is used in an unusually harsh environment (dust, corrosion, etc.) then a special heat exchanger is required in order to avoid contamination of the components.

Finally, a particular package exhibits mechanical characteristics such as weight and size. These parameters directly affect manufacturing and shipment costs. They determine whether a system can be built, and whether it can be shipped in a certain size (e.g., Boeing 707 airplane) or distribution channel (e.g., parcel post or United Parcel). The dynamic characteristics determine the type of vehicle (air ride van) in which equipment must be shipped.

It is also necessary to examine the particular design parameter in order to determine whether it is a constant (meets the German VDE standard x), a goal (cheap as possible), or part of a more complex objective function (measured in bits/sec/\$ or part of a system benchmark--jobs/sec/\$). Thus a system may be determined by a government standard (e.g., safety and EMI), a market consideration, a distribution channel (shipped via parcel post), a

G. Bell

created 1/4/78, latest edit 2/13/78

method of sales (catalog and customer installable), a competitor characteristic (e.g., no louder than x, faster than y, or cheaper than z), for a physical operating environment (dusty, hot and with high RFI).

The following table lists the various kinds of design activities and whether they are likely to goals, constraints, or part of more complex objective functions. The table also gives the dimensions of various metrics (e.g., cost, weight) available to measure the designs and many of these metrics are used in subsequent comparisons.

Table Design Activities, Metrics and Environment Setting Goals and Constraints

<u>Design Activity</u>	<u>Determining Environment and [Metrics]</u>
Primary function and performance (e.g., memory)	Market, next highest level (i.e. the consumer) of system [memory size (bits), operation-rate (bits/sec.)]
Human engineering	Human factors criteria, competitive market
Visual/Aesthetics	Market, other similar objects, the environment in which the object is to exist--usually only important at outer-most level
Acoustic noise	Government standards, operating environment, market [decibels]
Mechanical	Shippability (e.g., air cargo container size, truck vibration), handling, assembly/disassembly time [weight, floor area, volume, expandability, acceleration, frequency response]
<u>EMI</u> radiation input	Government standards, must operate within intended environment (e.g., high noise) [power vs. frequency] [standard no. 000]
Power	Operating environment, market [watts, voltage supply range]

G. Bell

created 1/4/78, latest edit 2/13/78

Cooling and environment	Market, intended storage and operating environment, government standards [heat dissipation, temperature range, air flow, humidity range, salinity, dust particle, hazardous gas]
Safety	Government standards [standard no. 000]
Cost	
cost/metric ratios	[cost/performance (its function)--cost/bit and cost/bit/sec., cost/weight, cost/area, cost/volume, cost/watt]
density metrics	[weight/volume, watts/volume, operation-rate/volume]
power metrics	[operation-rate/watt; efficiency = power out/power in]
reliability	[reliability--failure rate (Mean Time Between Failures, Mean Time To Repair (MTTR))]

Given the basic design activities, we must now examine their interaction with the hierarchy of levels (i.e., the systems) being designed (see Table 00). Here we look at each level, and discuss the interaction of the design activities and with other design activities (e.g., function requires power, power requires cooling, cooling requires fans, fans create noise and noise requires noise suppression).

G. Bell

created 1/4/78, latest edit 2/13/78

Table Interrelationship of Hierarchy of Levels and Design Activities

Activity \ Level	Chip	IC	Module	Backplane*	Box*	Cabinet	Computer Systems
Functional components	logic electrical			spatial\	what	configurations	selection of right
	circuit design physical layout		physical layout	physical layout	will fit and operate	boxes, what configs. will operate	(user is designer)
Human interface						location of console, size for use	placement for use
Visual					somewhat seen, bought for integration	determines system appearance	set of cabs, attractive place to be
Acoustic				Airflow vibration			quiet for operators and users
Mechanical loading,	buildable		shippable	serviceable			floor room size
	signal transmission						
EMI	noise coupling and rejection of ext.RFI		inter/intra module noise			RFI containment, external RFI shield	away from rfi input (outside operating range)
				coupling, RFI			

G. Bell

created 1/4/78, latest edit 2/13/78

				containment and shielding		
Power	special on-chip supplies		dist. and regulation	dist. and regulation	control, dist. and regulation	inter- connection with computer system by user. special power for high availability
Cooling to and other environment	chip to IC cooling special envir.	module cooling special envir.	IC to cooling	module cooling & covering		source of interbox couplin room air/ environment
Safety			power -----> for various systems		determines safety if used at this level	determines user level safety
Dominant design activities	circuit/ logic	logic ----->			mechanical, power, cooling, EMI, acoustic	configura- tion visual, shipping EMI, safety user configuration design


*Can be taken together as a single level.
(Alternatively, the box level may be eliminated in large systems)

G. Bell

created 1/4/78, latest edit 2/13/78

Computer Systems Level

At the topmost level is the computer system, which for the larger mini and DECsystem 10 computers in this book consists of a set of components within cabinets, which are housed in a room, and interconnected by cables which can be subjected to external environment (e.g., people walking on them). Observe that a large computer system such as the DECsystem 10 occupies a set of cabinets (Fig. 00, page 00) whereas a WPS78 word processing system is either one or two cabinets (see Fig. 00, page 00). The functional design activity is the selection and interconnection of the cabinets, with a basic computer cabinet holding processor, memory, and interfaces to peripheral units. Disks, magnetic tape units, printers, and terminals occupy free standing cabinets. The functional design is usually carried out by the user and consists of selecting the right components to meet cost, speed, number of users, data-base size, language (programming), reliability and interface constraints. Aside from the functional design problem, cooling and power design are significant for larger computers. For smaller computers accessibility, acoustic noise and visual considerations are significant because these machines become part of a local environment--and must "fit in".



Cabinet Level

The ultimate person responsible* for a computer system takes as given, the collection of computer system components (the cabinets) and connects them to form a system as described above. For the hardware computer system designer, the component associated with the cabinet is often his largest

G. Bell

created 1/4/78, latest edit 2/13/78

system. For example, the central processor designer is often responsible for primary memory and interfaces to peripheral equipment. This functional design insures that the various components (i.e., boxes) that make up a cabinet level system will operate correctly when interconnected. Since the cabinet is the lowest level component that most users interface to and observe, the physical design, visual appearance, and human factors engineering are also dominant design activities. Safety and EMI characteristics are also important because the cabinet also serves as last place for shielding. Cooling and power distribution can be a problem within cabinets since a number of different boxes may cohabit a cabinet. Thus there may be thermal and EMI noise pollution when box components interface poorly. The mechanical structure of a cabinet is responsible for maintaining its physical integrity when shipped.

Box Level

Box level functional design consists of taking one or more backplanes, the power supplies for the box and any user interface such as an operator's console and interconnecting them mechanically. The PDP-8 is the original integral box level design (see Fig. 00, page 00). It should be noted that when system are not distributed at the box level, then no separate box is required and backplanes are mounted directly in a cabinet; hence box and backplane design are merged as one. In a similar fashion, when only a single backplane is used, it is hard to separate the design associated with the box and backplane. In this way, power, cooling the backplane to hold the modules and the box to cover and support the other components becomes a

*A significant computer engineering task performed mostly by the user/buyer,

G. Bell

created 1/4/78, latest edit 2/13/78

single design.

If systems are sold at the box level, then the visual characteristics may be important, otherwise, the design is basically mechanical and consists of cooling, power distribution, and control of acoustic noise. The structure must be basically quite sound in order to provide for shipment.

Backplane Level

This level of design (when we separate it from the box level design) is just the final level of interconnection for the computer components that are designed to stand alone (e.g., a basic computer disk, terminal).

Backplane design is part of the computer's logical design. Figure 00, page 00 shows the Omnibus backplane for PDP-8 and how it is used to interconnect the basic components. Secondary design activities include holding, powering and cooling modules so they may operate correctly. Since the signals are transmitted on the backplane then there is an EMI design problem. For industrial control systems whose function is to switch power (see PDP-14 on page), then additional safety problems are created.

Module Level

In the third and fourth generations, module level design is one of the physical layout design problems associated with logical design. In the second generation module level design (see Fig. 00, page 00) was a circuit design activity taking discrete circuits and interconnecting them to provide a given logic function. Chapter 00 describes the electronic

G. Bell

created 1/4/78, latest edit 2/13/78

circuit designs of the second generation. Now, this interface between circuit and logical design is within a chip level design. This shift in roles (function) will be described below on the section discussing the interrelationship between the technology generations and packaging. The integrated circuits that perform the functions are assigned to different positions on the module. Note in a recent implementation of the LSI-11, a processor with 32 Kbytes of memory is packaged on a single double sized module (see Fig. 00, page 00). Since module level design is basically electronic, we note again that power, cooling and EMI considerations dominate.

Integrated Circuit Package and Chip Level

[need brief description of . ceramic, plastic DIP . TO-5, flatpack, alternatives to DIP]

Since the Integrated Circuit is often identified as a package with a configuration of two rows of pins, called the Dual In-line Package (DIP (see Fig. 5, page 00) we synonymously equate the IC package with DIP. The part of the semiconductor wafer, called a chip (see Fig. 00, page 00) is actually the Integrated Circuit. Thus, for every DIP there is currently only one chip; and since we equate DIP to IC, we can discuss the IC and chip as a single level. In the future, we might expect multiple chips to be packaged within a single DIP, creating a different design problem at the DIP (IC) and at chip levels. Figure 5 shows a single DIP with two chips as used by the LSI-11.

G. Bell

created 1/4/78, latest edit 2/13/78

Logical design is a part of the functional design problem at the chip level. The task is to supply a terminal behavior that can be used at the next (module level) and depending on the technology generation this function can vary from a simple gate (in the early third generation) to a full computer (late in the fourth generation). A completely correlated design activity is the circuit design with transistors; as such, the logician can interconnect the various transistors in accordance with a well established set of rules so that the circuit will carry out the required logical function. The physical assignment of circuits (functions) to geometry forms the remaining part of the functional design.

Other design activities at this level include generic to electrical signal processing: power, heat dissipation and EMI. Since some ICs are designed to operate in hostile environments, there is a considerable mechanical design activity associated with packaging, interconnection and manufacturing.

The Packaging Evolution

Figure 6 shows how correlation of packaging and the computer classes with the computer generations (and time). For each new generation, there is a short, evolutionary transition phase, but ultimately, the new technology is repackaged in such a way that a complete information storage or processing component (e.g., bit, register, processor) occupies a small fraction of the space and takes a fraction of the cost it did in the former generation. We can observe from the lowest level processing that quite discrete events

G. Bell

created 1/4/78, latest edit 2/13/78

mark the generation starting from 1 bit per vacuum tube chassis in the first generation and evolving to a complete computer on a single integrated circuit chip to mark the fifth generation. It is also worth noting that in the first generation, a chassis permanently mounted in an equipment bay was used whereas later on, as computers were built on a production basis, modules were created which were significantly smaller, more producible and removeable for servicing.

Although it is difficult to do precisely, the second packaging time line indicates the transition in package size for a basic processor and primary memory. For the minicomputer we use Whirlwind (perhaps overzealously) which is roughly the size and power of the LSI-11 (a four chip, 16-bit processor). Whirlwind occupied a complete building and the processor portion, a large room.

In the case of the computer classes several computers are given to mark the various technologies and sizes. Here, it is clear that the package plays a significant event in the determination of the new classes. Note the super, mini, micro, calculator, and terminal lines.

It is difficult to be too precise about the class distinctions as a given computer class can neither be precisely defined in price nor performance terms; we mostly argue that the price, packaging hierarchy and market distribution structure determines the computer class. In the following table we give the packaging hierarchy with time for what might roughly be

G. Bell

created 1/4/78, latest edit 2/13/78

considered to be a minicomputer.

**Table Physical Structure Packaging Hierarchy versus Technology Generation
for Roughly Constant Performance Computers**

<u>building</u>	<u>room</u>	<u>room cabinets</u>	<u>cabinet box(es) backplane</u>	<u>cabinet ? module(s)</u>
rooms	cabinets	backplanes	modules	integ.ckts.
bays	?	module	integ.ckts.	chip
chassis	chassis	disc.ckts.	chip	
disc.ckt.	disc.ckts.			
early first (e.g. Whirlwind)	late first ?	late second (PDP-1)	late third (PDP-8/E)	early fourth (CMOS-8)

Note: The line shows what is required for the processor part only. Only one each (e.g., 1 room...1 integrated circuit) of next lowest level component is required/processor.

In a similar fashion, we can observe how a particular computer component, now called the Universal Asynchronous Receiver-Transmitter (UART) has evolved with time in the following table.

**Table __ Packaging Hierarchy Evolution for Universal Asynchronous Receiver
Transmitter (UART) Telegraph Line Controller**

backplane			
modules	2 modules	module	
discrete	discrete	IC	IC

G. Bell

created 1/4/78, latest edit 2/13/78

circuit	circuit	chip	chip	chip area
early second	late second	early third	late thrid	late fourth

This logic carries out the function of interfacing to a communications line carrying serial data which is encoded by a DC level to denote a 1 or 0 and transforming the data to parallel on a character by character basis for entry into the rest of the computer system. The UART has three basic components: the serial/parallel conversion and buffering; the interfaces to both the computer and to the communications lines; and the sequential controller for the circuit. The preceding table shows the evolution of the UART packaging hierarchy with time.

The UART is probably the first fourth generation part computer component, since it is somewhat less complex than a processor, yet large enough, identifiable with a clean, standard interface.

Of historical note, we feel that DEC played a significant part in the development of the UART technology. With the PDP-1, the first UART was designed using 500 Khz systems modules (see Fig. 8). The PDP-1 was used in a message switching application as described in chapter 00. We called the interface a line unit, and it was subsequently repackaged in the late second generation to be on two extended systems modules (see Fig. 9). The UART function was built into the PDP-8 using programming (see page 00). Late in the third generation (or at the beginning of the fourth generation)

G. Bell

created 1/4/78, latest edit 2/13/78

we were able to develop a two chip UART, and ultimately one that occupies a single chip; this, subsequently evolved to become a standard IC. It was used in the PDP-11 DL-11 communications line interface module most recently (see Fig. 10).

Currently a single chip (Fig. 11) can interface to a computer bus, handle both the receiver and transmitter functions, and have options for various line protocols and speeds.

The DEC Computers Packaging Generations

With this general background on packaging, we can examine the DEC packaging evolution more specifically. Figure 3 gives the general archetype of the packaging structure and DEC, but specific schemes have been used at different generations. Figure 12 shows how the hierarchies have changed with the technology generations. The figure is segmented into the different product groupings. A product is identified as being at a unique level if it is sold at the particular packaging level. Note that with time, lower level components are available for use within other systems and different packaging hierarchies. The first computers (i.e., PDP-1 to PDP-6) were sold at the cabinet level as complete hardware systems. The PDP-8 was significantly smaller and sold at the mechanical box level so that it could be incorporated in other hardware systems and within other packaging hierarchies (e.g., as part of an industrial control system, instrumentation for a submarine). For smaller systems, a box level system often sufficed. Note that the PDP-8 was also available at the cabinet

G. Bell

created 1/4/78, latest edit 2/13/78

level for a complete system.

Subsequently the systems evolved to be available down to the backplane level with the LSI-11, and currently to the module level for the PDP-8, as embodied in the CMOS-8. The following tables expand on Fig. 12 and give the components, the interconnection scheme and the mechanical structure for various members of family.

The following table shows the original packaging hierarchy for most of DEC's second generation computers, which used a relatively common packaging scheme based on the PDP-1. The most significant change occurred in the late second generation when Flip Chip modules (see Fig. 12) were introduced so that backplanes could be wire wrapped automatically, enabling lower cost and greater scale production. Figure 12 shows:

Table Physical Structure Hierarchy for Second Generation Cabinet-Held Computer Systems (1, 4, 5, 6, 7, 9, KA10, LINC, 12)

cables	computer	floor/room
int.cables	cabinets*	cabinet frame
soldered wires**	backplane(s)	backplane, connect
PCB etch	module***	PCB
(interconnect0	discrete ckt.	(holding structure)
	(component)	

*single cabinet pair for PDP-5

**changed to wire wrap after PDP-6

***originally called systems modules, and changed to Flip Chip modules after PDP-6

With third generation IC technology, another packaging level was also

G. Bell

created 1/4/78, latest edit 2/13/78

added. In the case of DECsystem 10 and the 18-bit family, these computers were not available as boxed components. In chapter 00 we discuss how the unavailability of "boxed" 18-bit computers may have caused their demise. Also of significance was the packaging of later third generation computers (past the 8/I and 8/L) to the M-series Flip Chip modules to reflect the change in logical design with ICs. In all recent computers the module is treated as part of a single monolithic logical design, and partitioning (assignment of particular functions to physical components across multiple levels) is across the IC, module and backplane levels. This design activity is also shown in Table 00.

Table Physical Structure Hierarchy for Third Generation Box-Held Computers
(8/I, L, E, F, M, A, 11/04-11/70)

	box ----->	int.cables	cabinet boxes	cab.frame
wire wrap bp.	module	box, etc.		
PCB etch	ICs	PCB		
Wire board	chip	DIP		
(interconnect)	(component)	(holding structure)		

Computer-in-a-box

Computer-system-in-a-cabinet

This change to wire wrap technology also enabled the box level production of computers as shown in the following table for PDP-8, LINC-8 and the 8/S. Here, the changes to wire wrap and two level (box and cabinet level) products is clear.

Table Physical Structure Hierarchy for Second-Generation Box-Held Computers
(8, LINC-8, 8/S)

G. Bell

created 1/4/78, latest edit 2/13/78

	box ----->	int.cables	cabinet	
int.cables	backplanes	box	box	cab.frame
wire wrap	module	backplane + connect		
PCB etch	discrete ckt.	PCB		
(interconnect)	(component)	(holding structure)		

Computer-in-a-box

Computer-system-in-a-cabinet

The change to IC packaging in the third generation is when in the preceding table for computers that are sold at the box level. Note that this set of computers includes all the PDP-8 and PDP-11 computers. In this way, the minicomputer tradition continues to be available at the box level of the hierarchy.

The LSI-11 hierarchy is represented in Fig. 12 at three levels. Although components are sold as separate modules (e.g., communications line interfaces, additional primary memory), a complete system requires a backplane, thus the lowest level for the product is the backplane. For larger systems, a power supply is combined and placed in a metal box (the 11V03), and finally a complete system such as the 11V03 is available with terminal, and mass storage in a single cabinet.

The CMOS-8 hierarchy is presented in the following table.



Table Physical Structure Hierarchy for Fourth Generation Module-Held

Computer System (CMOS-8)

module	----->	cables	terminal-->cables	term.,mass store, desk
			modules	term.

G. Bell

created 1/4/78, latest edit 2/13/78

PCB etch	ICs	PCB
wire board	chip	DIP
(interconnect)	(component)	(holding structure)

Computer Module

Terminal

System

Note, that chapter 00 on the PDP-8 family describes the structure of the CMOS-8 module. The CMOS-8 module¹ is a complete computer with processor, 16 Kword memory, and all the optional controllers to directly interface up to five peripheral options. For a computer packaged in this fashion, a backplane is not used, and doesn't exist at a level. The computer system within a CRT terminal exists at somewhere between a box and cabinet configuration called the VT78. the third level of the hierarchy is within 1 or 2 cabinets (i.e., a desk) complete with mass storage (i.e., two floppy disks) and a separate stand alone printer (cabinet) for letter quality printing.

Specific Cabinet and Box-level Designs

Cabinet and box-level design is perhaps the most difficult part of computer design, yet it is perceived (incorrectly) to be trivial and left till last or because there is not a single discipline or complete set of well understood laws that govern the design. This deception arises from the fact that, on first glance, the only physical law is that not more than one thing can occupy the same space at a given time. Also, everyone has some (usually strong) feeling about what an attractive package is. However, as we have tried to state emphatically in the introduction, a number of disciplines must be understood. These range from acoustical engineering to heat transfer, aesthetics, human engineering to RF engineering, and finally

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

understanding just what function a package is to perform. In addition to this understanding, each box type requires a separate specialized manufacturing process. Obviously no one person fully understands all the disciplines necessary to correctly design a package. Herein lies the problem: Packaging is the integration of a number of separate disciplines.

To be specific we want to look at the basic packaging problem by first enumerating the possibilities for placing modules within a particular box package. By being quite restrictive, we can build a grammar with six production rules that describe the possible DEC boxes. Even with this simple grammar, about three million possibilities can be generated. The large (combinatorial) number of possibilities arise from the basic size and way a box is held, the console mounting, cooling, power supply location and the way modules are mounted within the box. We can explore these possibilities by simply writing statements that express the alternatives for the separate decision dimensions.

A Grammar and Syntax to Generate Three Million Basic Box/cabinet

Alternatives

Size and mounting

1. The box is	1/2	cabinet depth by	3-3/4"	high width	fixed	mounting (40)
	full		5-1/4"		slides	
			10-1/2"		right	
			15-3/4"		hinge	
			21"		left	
					hinge	

Console

2. The console is	non-existent			(14)
	simple, with power on-off			

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

| for maintenance
| for programming and
| maintenance

Cooling

3. Cooling is carried out by | box-level | fan(s) giving | (29)
| cabinet- |
| level |
| plenum | flow | normal | to the module mounting axis (8); or
| forced air | | parallel

cooling is carried out by natural convection with

air entry at | and exhaust | top | (21)
| bottom |
| right side | | right side
| left side | | left side
| front | | front
| rear | | rear

Power supply location

4. The external power supply is mounted | separately somewhere | (12)
| behind the box (attached) |
| on the back of cabinet |
| on the front of cabinet |

or; an internal supply is mounted using a mounting scheme identical to the one used by the modules; or the power supply uses a different mounting scheme and is located at the

| bottom | of the cabinet
| top
| right side |
| left side |
| front
| rear

Module mounting

5. There are | zero | backplanes for mounting modules | (3)
| one |
| multiple |

6. The module pin axis is | top | mounted, oriented in | horizontal | (24)
| bottom |
| front | | vertical
| rear |
| right side |
| left side |

plane with IC side facing | top |
| bottom |
| front |

¹ Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

|rear |
|right side|
|left side |

 (3.9×10^6)

We can apply now given the six sentences (1 point in the decision space)
that describes the PDP-8/A:

1. The box is 1/2 cabinet depth with 10-1/2" high with fixed mounting.
2. The console is for programming and maintenance.
3. Cooling is carried out by box-level fans giving forced air cooling flow parallel to the module mounting axis.
4. The box power supply uses a different mounting scheme and is located at the bottom of the box.
5. There is one backplane for mounting modules.
6. Modules are rear (wall) mounted, oriented in the horizontal plane with IC side facing up.

Alternatively, we can give the conventional three views of a PDP-8/A to show the location and configuration of the various box components (see Fig. 13). In this case a top view is adequate to show the configuration and air flow--although a side or front view is necessary to show the number of modules and length.

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

Obviously there are more than the 3.9 million possibilities that can be generated from the above grammar--although not all of these are feasible. It is important to have some way to relatively methodically examine the more common possibilities. In fact, we note that the above grammar doesn't even include the original PDP-8 package which consists of a box mounted on slides with two backplanes that can be hinged apart so as to provide access. Thus, it is not purely a box mounted on slides or a book-like structure suspended with hinges.

It is useful to look at how some of the design alternatives for a given dimension or several dimensions interact with other dimensions or effect evaluation criteria. Of all the dimensions to consider in the design, perhaps the most important is how the box (or module mounting structure) is placed in a cabinet. This placement effects air flow, shippability, configurability, cable placement, serviceability, etc. Here, we have a classical case of design tradeoffs. The scheme that provides the best metrics (i.e., packaging density (in modules/cu.in.) highest weight), has the poorest access for service, and cable connection characteristics, and only reasonable serviceability. These characteristics are given in the following table:

Table __ Fixed, Drawer and Hinged Box/Cabinet Mounting

	<u>SERVICE ACCESS</u>	<u>CABLING</u>	<u>DENSITY</u>	<u>COOLING</u>	<u>APPLICABILITY</u>
Fixed	Good for either backplane or	Best (i.e., shortest,	Good for thin or rear	Best (known)	Box not needed; box can be used

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

	module, but not both unless a thin cabinet is used	and right)	cabinet PS mounting		
Drawer	One side access	Long + moves	Very high	Can be	High density, self contained
Drawer (with tilt) for service	Good	Longer + much more moves	Very high	Cooled*	
Drawer vertical mounting modules	Very good	Longer + moves	High		
Hinged (module backplane)	Very good	Short	Medium	Good (if fans are fixed to cage)	Separate box is awkward

*Density restricts cabinet airflow

We can look qualitatively at the past packaging schemes. Figure 14 shows the various boxes (top view) as they fit within a cabinet profile. In some cases there is not a two level cabinet/box hierarchy as we have noted in the previous discussion on the packaging evolution because these structures have only been available at the cabinet level as we showed in Fig. 12. As one might guess from the table, the author's biases run to packaging schemes that are not the highest density, that are fixed so that the conditions can be well-understood and that they may be cabled rigidly. It is interesting to note how the evolution for the larger computers is toward these fixed structures. In some real sense, packaging is quite a matter of designer preference because the only clear goal is manufacturing cost. Marketing considerations, especially for OEM use, drives toward getting the highest density (to minimize volume, floor space and rack mounting height).

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

Although it can be shown that poor airflow increases operating temperatures and therefore decreases component life, we designers are too loath to make all the necessary measurements and carry out the calculations. In this way we can usually show that any increase in MTBF has a corresponding cost savings.

In the following table, we give the metrics for the various packages in terms of their pay load--expressed in Printed Circuit Board area. Thus we get pay load ratios of board area per cu. ft. of box, per sq. ft. of floor.

Power Supplies

This section will give cost and performance characteristics of various DEC supplies as shown in Table x. It will list a set of supplies, their (relative) cost, MTBF, volume/year, power, technology, size, regulation, voltage, efficiency, and the ratios cost/watt, cost/cu. in., and what it's used on, the year introduced.

Modules

It isn't clear what should be said here. Phister has quite a bit. It will depend on what happens in the Modules chapter. We should clearly reference board area. We must bring in the backplane and get at the pay load of board area, square inches of panel area and cu. inches, then talk of efficiencies. The pin should also be a metric. Line width is the standard metric...but I'd just as soon not give it.

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

PACKAGING FIGURES LEGEND

Note there is a reference to photos elsewhere in the book to illustrate the various packaging concepts.

- Fig. 1. Packaging consists of placing boxes within smaller boxes...within boxes on an indefinite (not recursive) basis.
- Fig. 2. A packaged system provides some function, but in addition gives a visual impression, usually requires cooling of some sort, has certain mechanical characteristics and an Electromagnetic Interface (EMI).
- Fig. 3. Eight level packaging hierarchy for DEC second to fourth generation computer systems.
- Fig. 4. Packaging is a set of closely interrelated design problems.
- Fig. 5. A DIP with two semiconductors per chip.

Packaging Photos (need somewhere)

- Fig. 6. Time line evolution of computer generations, packaging and classes (with examples).
- Fig. 7. None
- Fig. 8. PDP-1 control (called Line Units) for interfacing eight teletype lines using what is now called the Universal Asynchronous Receiver-Transmitter, (UART).
- Fig. 9. 4706 and 4707 receiver and transmitter line units of the late second generation.
- Fig. 10. DL11 line unit based on early fourth generation IC.
- Fig. 11. UART chip and block diagram.
- Fig. 12. DEC physical structure (packaging) hierarchies vs technology generations.

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

Fig. 13. Three views of PDP-8/A box.

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

AN OVERVIEW OF MANUFACTURING

The result of a design project is an entity which is manufactured. However, very little is written about manufacturing in computer engineering literature as we generally discuss algorithms, logic design, and circuit technology. And yet for a computer to be commercially successful, it must be manufacturable, economically operable and serviceable. Moreover, for most of the computer engineering discussed in this book, because the designs are intended for production, engineering costs are small (1-10%) compared with other product and lifecycle costs. The product cost is determined by the price of the components and the manufacturing process. The lifecycle cost includes the purchase price, the operational and service costs. For production, machines must be easy to assemble and test, repair must be rapid, engineering changes must be introduced smoothly, and the production line cannot be held up because of shortages of components -- all parts of traditional manufacturing understanding.

A detailed discussion of manufacturing is clearly beyond the scope of this text. For process engineering we suggest [Grant and Leavenworth, 19xx]. Information on test equipment is to be found mainly in manufacturers manuals, but a survey is given in []. A reference on quality control is [Juran, 1962] and a discussion of learning curves is in [].

The Life Cycle of a Product

Figure Lifecycle shows a simplistic process flow for the major phases and milestones in the life of a product. In reality, planning and designs for

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

many of the phases go on concurrently. The early research, advanced development and definitional phases are not shown. Often times, products may proceed from the idea to engineering breadboard and are terminated because they may not meet original goals or because better alternative ideas arise.

The engineering breadboard usually is built with wirewrapped boards rather than printed circuit boards in order to accommodate changes. At this stage schedules are made for manufacturing start up. Other organizations formulate and execute plans: systems' engineering - for product test/verification; software engineering - special software and verification; marketing - for promotion and product distribution; sales - for training; field service - training and parts logistics; and software support.

The engineering prototype proves the design using the printed circuit modules that will be used in the manufacturing. All peripherals in the planned systems configurations are tested on the prototype. Usually a number (10-100-depending on the complexity, cost and volume) of prototypes are constructed. The complete system must meet the product specification.

Limited release (LR) to manufacturing is a major milestone. The product is placed under formal engineering change control; specifications and documentation are available for the product and manufacturing process. For the integrated circuits, second sources of supply and testing procedures

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

are in place. Process control tapes for the numerically controlled machine tools, e.g., component insertion, backplane wiring, and printed-circuit board drilling, are ready. Any special tooling for the mechanical packaging has been obtained. Finally, the engineering change order (ECO) procedure for the hardware and microcode is set. Testing at all levels has been specified; test programs for computer-controlled testers have been written, special test equipment has been built, and diagnostic programs are ready. There must also exist a clear process by which engineering change orders (ECOs), including microprograms, can be made quickly in response to any as yet undetected design errors or changes that are necessitated by manufacturing process or parts availability.

Design maturity testing (DMT) with a number of engineering prototypes verifies the design and justifies the risk of a pilot run. Tests for reliability and functionality are conducted. Environmental testing (shock, temperature, humidity, static discharge, radiation, power interrupt, safety) is conducted at this stage.

The pilot run shakes down and verifies the actual manufacturing process by building a small number of units at the manufacturing plant using the product and process documentation.

Product announcement usually occurs during the DMT period but can be as early as limited release or as late as first customer shipment -- depending on the marketing strategy. This strategy is clearly a function of the

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

volume, novelty, and competitive needs.

Process maturity testing (PMT) verifies that the product is being manufactured with the desired cost, quality, and product rate. The steady-state phase of manufacturing continues, with possible perturbations due to the introduction of product enhancements or process changes to lower product costs, until the product is phased out.

Manufacturing Process Flows

An overview of a manufacturing process is given in Figure Manufover which shows how a product moves through the various factories. There are different DEC and independent manufacturer plants for boards, peripherals, memories, and central processors that form the box-level and independent hardware (e.g., memory modules) components. Integration from the other stages and stack storage occurs at the stage called final assembly and test (FA&T). Here, the software system operations manuals and other documentation that is to be run is also integrated and tested.

In Figure process60 we give the complete flow for a typical volume manufacturing line -- the 11/60 central processor facility in Aguadilla, Puerto Rico. Integrated circuits occur outside this factory and are sent to be inserted, soldered in, and tested. The photographs in Figures x through y show several test points in the process.

Integrated circuit failure rate versus time follows a bathtub-shaped curve,

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

high at either end of the life cycle, as shown in Figure Bath. The manufacturing process includes extensive thermal cycling to ensure that the "infant mortality" cases are discovered early during manufacturing, because it is more expensive to find defective components at the larger, more integrated systems level. The temperature/humidity environmental chambers which house n CPU's each are shown in Fig. TChamber. Figures 2224 and lvstation show small heated enclosures used to induce failure during the test and repair of modules.

Since testing occurs at each stage in the manufacturing process, dedicated logic must be added to the design, to provide physical access "probes" for the test equipment. To test a particular function, it must be: specifiable, invocable, and observable. For example, the function of an adder can be clearly specified but it cannot be easily invoked or observed if its inputs and outputs are etch runs on a printed circuit board. Several testing strategies are used: add signal lines from the adder to the backplane where there are adequate probe access points; probe directly onto the module etch or IC pins; and subsume the adder in a function whose inputs and outputs can be more easily controlled and observed. The general problem can be modelled by the simplecircuits shown in Fig. obs. Each gate could be part of a computer on a chip where there is no possibility for repair. The problems of observation and control exist at all levels of integration. Examples of observation points at each level are given in Table Obslevels for the 11/60 computer.

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

In Fig. QVstation, the function being tested is a complete CPU which is packaged on several printed circuit modules. Behavior is being induced by PDP-11 programs and observed by inspecting of the results in memory using a diagnostics program or manually. However, a lower level of behavior can be observed (on the special display panel at the right) and controlled (by varying the clock rate of the CPU). The lights on the display panel are driven from backplane signals and show the contents of certain registers, e.g., micro-instruction register, program status register, and the ALU output register.

Table: Obslevels

Level in computer hierarchy	Observation point	Stage in manufacture of computer	Example
electrical circuit	transistor contacts on metallization layer	semiconductor fabrication	wafer test with micro-probe
switching circuit	leads on I.C. package	incoming inspection of ICs	
register transfer	etch run	module	probe on PC board (module-specific test) GR
register transfer	backplane	module	2224 memory exerciser for cache
PMS (Pc)	Unibus	CPU	
PMS (Pc)	contents of memory	CPU	diagnostic programs at subsystem level, e.g., memory-management unit, or processor instruction

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

			set tests
PMS (C)	contents of memory	System integration	peripheral diagnostic programs
PMS (C)	Unibus	System integration	DECX-11 bus exerciser

Figure APT shows the final acceptance of a unit (CPU, memory, power supply, and console) prior to final assembly and test of the system using a system called Automatic P? Testor. In both Figs. OVstation and APT, the entire instruction set is tested; a central computer system loads the diagnostic programs and monitors their execution. Several hundred lines emanate from this system to all of the computers under test throughout the production line. The Unibus of the processor under test serves as the umbilical cord [Chapter 00, p. 00] to accept the diagnostic programs and to be monitored by making memory observations.

Modules are tested using several methods. One method (see step j of Fig. Process60), uses the GR tester. Here, every signal input and test point on the module is probed using a fixed "bed of nails" test probe. The testor then selects the desired input and test point. In another method, (see step k, Fig. Process60) the module under test is placed in a processor in which all of the other modules are known to be good to be verified. This later test is necessary because the first test usually doesn't run at the operational speed of the computer, nor is such a test guaranteed to cover 100% of the logic.

Following the inter-plant transfer, systems integration begins. Figure

¹ Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

FA&T shows the process flow for the 11/60 at the Westminster Plant. For some computer systems, FA&T is carried out by having an individual technician follow a system through each of the three phases: incoming test of the CPU cabinet assembly, integration of peripherals, and final acceptance. Thus one or more have individual responsibility for fabricating a single system in what is a traditional, hand-crafted, job-shop fashion.

Alternatively, FA&T is carried out in what is fundamentally a continuous, production line environment called the Common Systems Integration (CSI) line. Here, workstations exist at each stage of the production line, and are interconnected by conveyors.

The floor plan of the Westminster Common Systems Integration (CSI) area is shown in Fig. CSI. Figure spurs shows four of the workstations in the integration phase. A turntable in the center of the spurs routes the systems to the workstations.

"Statistical Quality Control", Grant and Leavenworth, Fourth Edition, McGraw-Hill, 19xx.

"Quality Control Handbook", Juran, J. M., McGraw-Hill 1951; Second Edition, 1962.

¹Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

G. Bell

created 1/4/78, latest edit 2/13/78

An Overview of Manufacturing

List of Figures

Figure	Lifecycle
	Breadboard
	Manufover
	process60
	Bath
	Tchamber
	2224
	QVstation
	GRtester
	Obs
	APT
	CSI
	spurs
	[Waferprobe]

¹ Here we are taking the liberty of describing a component that is not currently available as a product, except as a spare part.

CHAPTER 2: TECHNOLOGY, PACKAGING AND MANUFACTURING

It is customary when reviewing the history of various industries to ascribe the events therein to either market pull or technology push. The history of the auto industry contains many good examples of market pull, such as the trends toward large cars, small cars, tail fins, and hood ornaments. The history of the computer industry, on the other hand, is almost solely one of technology push. Whereas the rest of this book gives examples of the effects of technology push, this chapter explores the individual elements that have made up the technology push.

Rather than discuss all of the technologies shown in Fig. Memtax, this discussion will concentrate on semiconductors and magnetic recording. Semiconductors have been the dominant factor in technology push within the computer industry, but magnetic recording density on disks and tapes has evolved rapidly too, and must be understood as a component of cost and a limit of performance. Communications links have been left out of the discussion because during the past twenty years they have not evolved as rapidly as other technologies have. Rapid evolution is expected in the immediate future as satellites and fiber optics become more economically attractive. The packaging (cabinets and boxes), interconnection, and power aspects are discussed, principally because these do not represent pushes but rather are large, high inertia objects that have to be pushed against in the hope that they will eventually yield.

The semiconductor portion of the discussion begins by presenting a tree of the

possible technologies arranged according to the function they carry out and showing how these have evolved over the last two or three generations to affect computer engineering. The cost, density, performance, and reliability parameters are briefly reviewed, and then the application of semiconductors, using various logical design methods, is discussed with particular emphasis on how the semiconductor technology has pushed the design methods. For example, one question of interest is whether users should design ICs or design with ICs. The conclusion in this particular case should be obvious...IC design is expensive and time consuming and should be avoided for all but a few applications. The same advice could also be carried over to those who believe it is necessary to program in assembly language or to have access to microprograms. For some reason, many people believe that there is a need to have access to lower level building blocks when building many large objects. This is akin to starting a building construction project by looking for a clay formation and a sand pit from which to make bricks and mortar. Successful contractors buy their bricks and mortar.

The discussion of semiconductors is followed by two sections on memories. The first, on core memory, provides a number of lessons on managing evolving technology. Core was the dominant primary memory for about 15 years even though it was challenged several times. It eventually lost because of the larger amount of development being applied to semiconductor memories. The second memory section describes the notion of memory hierarchies and how they operate to utilize virtually every type of memory that's either cheaper or faster than other types of memory.

The final section of this chapter presents some general observations about technology evolution: how technology is measured, why it evolves (or doesn't), cases of it being overthrown, and a general model for how it operates and is managed.

The best way to observe the evolution and interaction of logic and memory technology as they effect computer design is via a time line diagram (see Fig. LMtimeline). For logic, small memories, read-only memories, and primary memories, four lines give the significant events that have affected the technology push.

A single transistor circuit performing a primitive logic function within an integrated circuit (IC) is among the smallest, most complex of man-made objects. Alone, such a circuit is intrinsically trivial, but the fabrication process for a set of structures to form a complete integrated circuit is complex. To the digital IC users there are several relevant parameters:

1. The function an individual circuit performs within the IC, the aggregate function of the IC, and the functions a complete IC family performs.
2. The number of primitive digital switching circuits (or transistors) per IC. This density is a measure of the capability of the IC process.
3. Cost.
4. The performance of each circuit and the performance of the aggregate IC

and/or set of ICs within a family as they affect system performance (as measured by the time it takes to perform its function). The semiconductor circuit technology family (TTL, Schottky TTL, ECL) usually determines this performance.

5. The number of interconnections (pins) to communicate outside the IC. This determines the complexity of the functions to be performed by a complete IC family.
6. The reliability. This parameter is a function of the circuit technology, density, number of pins, the operating temperature and use (or misuse).
7. Power consumption.

Figure ICtree shows a family tree (taxonomy) of the most common digital IC's, roughly in order of increasing complexity. The secondary ordering is roughly by the regularity of the function being implemented, and whether there is memory associated with the function. The clustering of circuit types is also by technology generations (from the second to possibly the fifth). Note that large totally regular functions can be built with only memory (various registers and memories) and with no memory (adders, multipliers, multiplexors). With large scale integrated circuits it is desirable to implement regular structures to simplify understanding and to aid testing.

In the early third generation only completely unconnected components were

built. Collections of the basic logic primitives (AND, NAND, Exclusive OR, Adders) and sequential circuit components (individual flip flops or registers formed from groups of flip-flops) permitted logic functions that had occupied a printed circuit module of the first and second generations to occupy a single IC. This had the benefit of providing a drastic reduction in size between second and third generation designs, as shown most vividly by comparing the PDP-9 and PDP-15 (page 00), but had the drawback that modules now contained a wide variety of functions and were thus specialized.

As the densities began to improve to a hundred gates, the construction of complete arithmetic units on a single chip became possible. The earliest and most famous function, the ALU (Fig. ALU), provided up to 32 functions of two 4-bit variables. By the fourth generation, it became possible to construct very large combinational circuits, such as a complete 16 x 16-bit multiplication circuit requiring about xx gates, on a single chip.

Without well-defined functions such as adders and multipliers, semiconductor suppliers cannot provide high density, high volume products because there are few large scale, general purpose universal functions. The alternative for the users is to interconnect simple logic circuits (AND gates, flip flops), but this does not permit efficient use of the technology and the cost per function remains high (about that of the third generation) because the printed circuit board and IC packaging costs (pins) limit the attendant cost reduction.

Two methods of effectively customizing LSI are shown in Fig ICTree and discussed in greater detail later in the chapter. These are the PLA and the gate array. The PLA (Programmable Logic Array) is an array of AND-OR gates

that can be interconnected to form the sum of products terms in a combinatorial design. Specialization, by blowing fusible links, can be done either in the factory or in the field (FPLA) to give the general functions of input variables. The gate array structure is an alternative for doing logical design, but is more powerful than the PLA technique because both combinatorial and sequential circuits can be constructed. Gate arrays are simply a large number of gates placed on the chip in fixed locations where they can be interconnected during the final metalization stages of semiconductor manufacture.

There is a special branch of the tree shown in Figure ICTree for the purely memory functions. Memory is used in the processor as conventional memory, but can also be used as an alternative to conventional logic for performing combinatorial logical functions. For example, the inputs to a combinatorial function can be used as an address, and the output obtained by reading the contents of that address. Memory can also be used to implement sequential logic functions. For example, the memory can be used to hold state information for a microprogram. Since memories have so many uses, this branch is discussed separately in the memory section.

The remainder of the interesting logical functions include combinations of logic and memory. There are various special functions such as LPC (Linear Predictive Coding) encoding algorithms for use in real time applications and data encryption algorithms for use in communication systems. One of the most useful transducers, and the first one to use LSI, was the UART (Universal Asynchronous Receiver Transmitter).

There is a special section on bit-slice components (actually sets of 1, 2, 4, 8 and 16-bits). These are being used to construct most of today's high-speed digital systems, mid-range computers, and computer peripherals. Although there have been several bit slice families, the AMD 2900 series, whose register transfer diagrams are shown in Figure AMD, has become the most widely used. Note that all the primitives of this series were present in the RTM family, including the microprogrammed control unit referred to in Chapter xx as the K(PCS) (Programmed Control Sequencer). RTMs were also the archetype antecedent in the case of Fairchild's macrologic, according to the person responsible for the family, Kris Rallapalli.

The final branch of the tree is the most complex, and is used to mark the fourth (microprocessor-on-a-chip) generation of technology and the beginning of the fifth (microcomputer-on-a-chip) generation. The fourth generation is marked by a complete processor being packaged on a single silicon die, and the fifth generation, by this measure, has already begun since a complete computer (processor with memory) now occupies a single die. The evolution in complexity during each generation simply permits larger word length processors or computers to be placed on one chip. At the beginning of the fourth generation, a 4-bit processor was the benchmark, whereas toward the end of the fourth generation, a complete 16-bit processor, such as the PDP-11, could be placed on a chip. The LSI-11 (Chapter 00) was designed at about the middle of the fourth generation and hence, four chips were required for its implementation.

The function performed on a chip is clearly dependent on the number of

gate that can be placed on a chip. Thus, density in gates/chip is the single most important parameter determining chip functionality. From this measure, one can predict the functions likely to be implemented by just following the tree. It should be noted that the whole tree is relatively alive and has dense areas of new branches everywhere except at the top, where unconnected gate and register structures have been relatively static. In the growing areas, as density increases sufficiently, a new branch grows. For example, the processor-on-a-chip started out as a 4-bit processor (or rather as 2 chips for a single processor) and then progressed to have 8-bit and 16-bit processors on a single chip. Similar effects can be observed with the arithmetic logic unit and with memories.

The number of gate circuits per IC is the measure of density as seen by a user (see Fig. Semiden). This metric is actually the product of the circuit area and the number of circuits per unit area. Circuits are photographically reduced in size to yield higher speeds and higher densities. The processing techniques to create the semiconductor materials have also been improved to have better yields. Circuit and device innovation have also contributed to density and yield increases.

Semiconductor device (individual circuit) performance is also correlated with the implementation technology and density because a reduction in size also reflects a reduction in power, allowing greater speed of operation without exceeding IC package heat dissipation limits.

An Operational Model for Memory Size versus Time

The model given in Fig. Semidens is exponential and correlates with previous observations that the number of bits per chip doubles every two years according to the relationship:

$$\text{number of bits per chip} = 2^{t-1962}$$

There are separate curves, each following this relationship, for ROMs in prototype quantities, ROMs in production quantities, RAMs in prototype quantities, and RAMs in production quantities. Thus depending upon the product and the maturity of its production process, one must scale the state of the art line appropriately by one or two years according to the following rules:

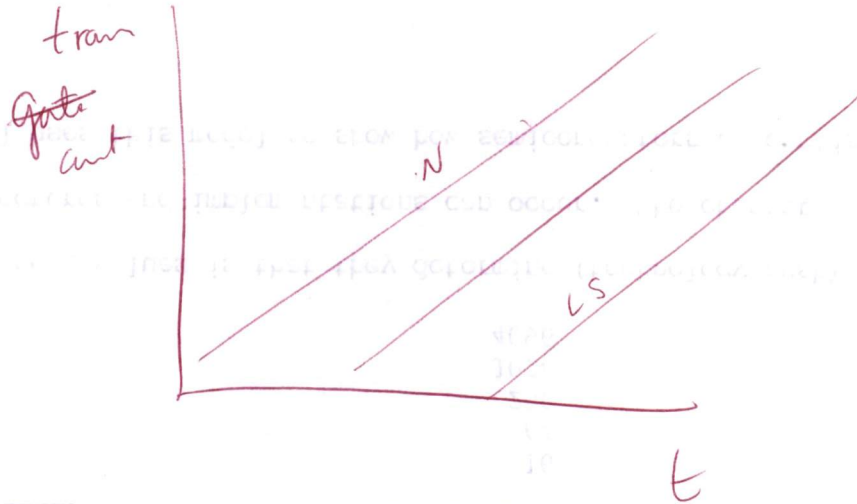
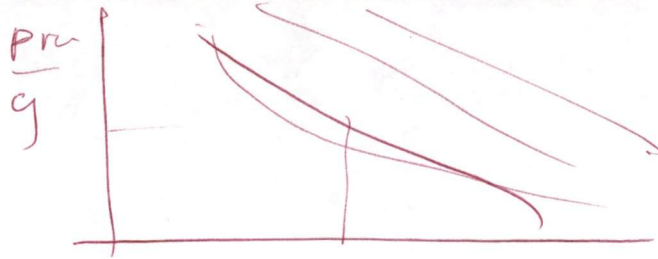
[rules]

This gives the following availability of various semiconductor memories:

<u>Year when first widely available</u>	<u>Number of bits</u>
1969-70	16
1971-72	64
1973	256
1975	1024
1977	4096

The significance of these values is that they determine (technology push) where certain architectures and implementations can occur. The chapter critiquing the PDP-11 uses this model to show how semiconductors accomplish this push.

The most important characteristic of ICs after density is cost. The cost of



	CMOS	N	B _n
2 Imp.	6	3	2

4-1083

ICs is probably the hardest of all the parameters to identify and predict because it is set by a complex marketplace. For circuits that have been in production for some time and for memory arrays, the price is essentially that of a commodity like eggs or bacon, and users generally consider these ICs as very similar to commodities, with the attendant benefits (cost) and problems (having a sufficient source of supply). In these cases, the prices are proportional to the die cost (which is proportional to the die area) and the manufacturing volume. For those circuits that have not yet reached commodity status, the prices are higher and depend on the strategy of the supplier - whether he is willing to encourage competition.

{Two curves are presented to reflect the price of various components/IC.

{Figure Comprice shows the price per ^{fundamental switching unit} ~~component~~ ^{gate} for an IC assuming LSI. There

{is a price band for the circuit size and circuit technology. Table GateComp

{gives some idea of how circuit density (~~in components~~) relates to actual

{functions. ^{logic function implemented}

The most useful data to understand past and future computer structures is the semiconductor memory cost curves (see Fig. Memprice). Here, the basic cost/bit of various sized memories is given. In 1978 the cost per bit was roughly .08 and .07 cents per bit for the 4 Kbit and 16 Kbit IC chips respectively, giving IC costs of \$3.30 and \$11.50 respectively. Whereas the chip density improves by a factor of two each (Fig. Semiden) year, the cost per bit (at the IC level) is declining at only a factor of 2 every two years. The line drawn in Fig. Memprice has the equation:

$$\text{cost/bit} = K \times 0.Y^{t-197y}$$

It is also interesting that the cost compares favorably with the price decline observed in core memory over the period since 1960-1970 for the 18-bit computers, (page 00) and for the cost declines in both PDP-11 and the PDP-8 (pages 00 and 00).

Performance

The performance for each semiconductor technology evolves at different rates depending on the cumulative learning associated with design and manufacturing process together with the marketplace pressure to have higher performance for the particular technology. One may hypothesize that each technology can be looked at as being relatively appealing or relevant to the particular design(er) styles associated with the computer market levels (view 4, page 00). One would expect the evolution to continue along the lines shown in the following table for the next few years.

Table SemChar: Characteristics of Dominant (1978) Semiconductor Technologies

<u>Type</u>	<u>Evolution</u>	<u>Use</u>
TTL (Transistor-Transistor Logic)	TTL TTL/Schottky TTL/LS	logic, bus interfacing higher speed than plain TTL same speed as TTL, but low power
ECL (Emitter Coupled Logic)	MECL II, III MECL 10K, 100K	very high performance easier to work with
MOS (Metal-Oxide-Semiconductor)	p-channel n-channel	low cost greater densities, cost evolving to performance (memory)

CMOS (Complementary
MOS)

low power, higher speed,
better noise immunity

Some of the lesser used technologies such as I²L (Integrated-Injection Logic) and SOS (Silicon on Sapphire), which both promise higher speed than TTL and lower power than MOS have been omitted. Both of these technologies have been touted as replacements for various technologies shown in the table. But, if an entrenched technology has evolved for some time and continues to evolve, it is difficult for alternative technologies to displace it because of the cumulative investment in process technology and understanding.

Semiconductors appear to be characteristic of other technologies in that usually only a single technology is used for a given problem. Each technology has a niche and there is only one winner.

The early predecessor technologies, RTL (Resistor-Transistor Logic), TRL (Transistor-Resistor Logic), and DTL (Diode-Transistor Logic) have also been omitted. They were too deficient in some aspect of cost, performance, or reliability to become standards of note. These technologies were important to the history of ICs because they were used in the first ICs, but many manufacturers, including DEC, did not use them because they did not represent a sufficient advance over the discrete transistor circuits already being used. In addition, these early circuits were packaged in flat packages rather than in the dual in-line package used today, and machine insertion of components was thus not possible.

DEC's use of the various IC technologies shown in the table is probably typical of most of the computer industry: TTL for mid- and high-sized

minicomputers; ECL for the larger scale machines (DECsystem 10); MOS for memories, microprocessors, and specialized high density circuits; and CMOS for special microcomputers (CMOS-8).

Figure Speedpwr and Gatedelay give the two most useful measures of performance for the various technologies as they have evolved with time. The speed-power product metric for a technology at a given time indicates that the user may tradeoff performance against power. There are limits to this tradeoff, as only about one watt can be dissipated by an IC package unless special cooling is used. The first curve presents the speed-power product and was generated by Jerry Luecke of Texas Instruments (TI) at a time when I^2L technology had just been introduced (Oct. 1975) by TI; some recent data points have been added for recent advances in MOS technology. Table Speedpwr gives some of the operating points for common processes at various times. Figure Gatedelay shows the gate delay through a single gate and is the best performance measure because the implication of speed and power are interchangeable is erroneous. That is, one can neither operate at very high or very low densities subject to the package power dissipation constraints.

The number of pins that communicate outside the circuit indicate the packaging technology level. Low and medium scale IC's often need more pins than LSI since the latter is capable of carrying out a complete function independently--and hence can operate with less interaction with the rest of the world. Memories are especially easily adapted to living within pin limitations, since only one pin is required for every factor of two increase in size. In addition, even this modest pinning requirement can be reduced by time

multiplexing the address signals.

Reliability

Over the past 15 years, the failure rate for standard ICs has been reduced by two orders of magnitude to the neighborhood of .01% per 1000 hours. This corresponds to 10^7 hours mean time to failure (MTTF) per component. Figure Reliab, from a recent survey article by Hodges [1977] shows the trend. The lower curves show the higher reliability obtained when more extensive testing and screening are employed. The improved MTTF of between 10^8 and 10^9 are obtained at a cost increase of 4 to 100 times per component.

EVOLUTION OF SEMICONDUCTOR USE

The LSI Dilemma

The economics of the LSI circuit industry make it essential that circuits with a high degree of universality be produced. Because of the learning curve of a manufacturing process (said to be X% per m circuits), cost is inversely proportional to volume. For a design to be sold in high volume, it must be usable in a large number of applications. However, the trend in circuit complexity, which allows semiconductor manufacturers to put more transistors on a constant die area each year, lends to increasing specialization of function. The more specialized the function, the lower the potential volume and the higher the price. Figure generality shows the dilemma. The LSI product designer is therefore continually in search of universal primitives or building blocks. For a certain class of applications, such as controller applications, the microprocessor is a fine primitive and has been so exploited

[Noyce, 1977]. For some applications, circuit complexity can embrace even higher functionality at the PMS level. The Intel 8XXX is an interesting example here: two processors, a 2.5(?) microsecond byte-processor and a 200 nanosecond bit-processor are combined in one LSI circuit.

Moore [1976] discusses this dilemma in a paper on the role of the microprocessor in the evolution of microelectronic technology. He points out that a similar situation existed when i.c.'s were first introduced. Users were reluctant to relinquish the design prerogative they had when they built circuits from discrete components. It was not until substantial price reductions were made that the impasse was broken. Then the cost advantages were sufficient to force users to adopt circuits that fit the technology.

The difference, however, between the i.c. switching circuit level of RT-level building blocks on the one hand and the PMS-level block on the other is that the latter uses the powerful concept of a shared-program computer. Its function is varied by programming.

For many applications, including most computer systems, the microprocessor on a chip is not a cost-effective building block, and other solutions to the dilemma are found. For example, microprogramming is a highly general way of generating control signals for data path elements, table lookup is a highly general technique. Both methods are attractive because they use memory, an inherently low cost LSI circuit. Microprogramming, however, does have limitations. The extra level of interpretation extracts a performance penalty and some potential datapath parallelism is often given up to reduce cost. A

more subtle, but practical, limitation is the development cost of microcode.

Assuming the writing rate to be 700 microwords per man year for horizontal micro machines, a desire to limit the effort to 20-24 manyears would limit the maximum control store size to about 16 Kwords.

Other techniques involve changing the pattern of interconnections via the top level(s) of metallization of an integrated circuit. This can be done by a factory made change or by the user, a much more flexible solution. Some of this customization is required in nearly all digital system design of the fourth and fifth generation. At the very minimum, the values for a read only memory have to be set. Table funvar contrasts the building blocks available in the fourth generation.

As a result of the increased basic circuit functionality available at each new generation, design methods have changed with the generations. This book provides an example of each, as summarized in the following table.

Table: Design Method versus Generation

<u>Design Method:\Generation:</u>	<u>First</u>	<u>Second</u>	<u>Third</u>	<u>Fourth</u>	<u>Fifth</u>	<u>Examples in this book</u>
Combinational and sequential; use of "standard" modules, IC's. PDP-8	s		s	s		18-bit;
Read only memory and PLA; microprogramming			s	m		PDP-9; PDP-11
Microprogramming with standard RT elements (high perf.) minor logical design					s	m CMU-11

Programming using micros and logic for interfaces	p	p	s	x	LSI-11
PMS design using completely specified and pre-designed microcomputer components				s	Cm* (almost)
Customized chip design and standard logical design (high performance)		m	m	m	LSI-11

s - the standard method for most digital systems

m - done by manufacturers of basic equipment

x - also used

p - prelude to micros, also done using minis

Table funvar

Building block	Technique for varying function	Degree of generality	Permanence of change
Computer module	program	v. high	none
microprocessor	program	high	low to medium
bit slice	microprogram	medium	medium
ROM	factory mask change	v. high	irreversible
PROM	field change	v. high	irreversible
EAROM	field change	v. high	low
PLA	factory mask change	medium	irreversible
FPLA	field change	medium	irreversible
gate array	factory mask change	medium	irreversible
RAM	write	v. high	none

The design of most relatively high speed digital systems (including low- to mid-range minicomputers) is carried out using standard register transfer ICs complete with data path and memory. Part IV discusses the current standard RT

modules set. These bit-slice components will evolve and take on specialized functions to achieve both higher performance (e.g., a multiplier) and to be specialized to particular tasks (e.g., the interpretation of a particular computer's ISP). As an example, the series may be modified for the coding and encoding required in signal processing.

For higher performance computers, there is no alternative to using either tightly packed standard ICs or building a unique set of ICs using some form of customization. Although Cray continues to build the high speed computers (in the CDC 6600, 7600 and Cray 1) with no custom logic, he does so by using impressively dense modules with high density interconnection and freon cooling. The high performance IBM and Amdahl machines use custom ECL circuits to improve packaging.

The current spectrum of IC's in use is summarized in Table cost speed spectrum.

With the advent of the processor on a chip, digital system design has been or soon will be converted completely to computer design. Problems such as controlling a CRT, controlling a lathe, building a billing machine, or implementing a word processing system become just computer system design problems similar to those attacked over the first three generations. The hardware part of the design, the interface to the particular equipment, is straightforward. The major part of the design is the programming. Since the late 40's three complete generations have learned about computer design, especially programming. The first generation discovered and wrote about it. Then it was rediscovered and applied to minicomputer systems. This time, it

is being learned by everyone who must use and program the microcomputer. Each time, for each individual or organization, the story is about the same: people start off by programming (using binary, octal or hexadecimal codes) small tasks, using no structure or method of synchronizing the various multiple processes; the interrupt mechanism is learned, and the symbolic assembler is employed; and finally some more structured system--possibly an operating system is employed. Occasionally users move to higher level languages or macro assemblers, but usually not because engineers are taught to minimize product cost (as opposed to development, life cycle, maintenance, etc. costs).

In view of this cyclical history, it seems likely that current digital systems design practice, which consists of building simple hardware interfaces to relatively poorly defined busses together with programming the application, will be relatively short lived. The design method of the future will be at the PMS-level component, although at the moment it is still too difficult to be done reliably and cheaply by large numbers of engineers. The components from which the microcomputer systems will be formed will be significantly more advanced using much better packaging, clearly defined busses, standard more general interface, and base level operating systems that are embedded in hardware (i.e., placed in read only memory to give the feeling of permanency so that users are less likely to embark on the expensive, unreliable rediscovery path). Standard components will be built which can be interfaced to a wide range of external systems using parameters that are specified by a field programming method instead of using logical design and building with interconnection on modules. In this way, the complexity of individual ICs can

be increased and having a standard method for interconnection, higher volume and lower costs will result.

Before discussing the alternatives associated with IC design, it is important to characterize the various costs. Figure Design.cost shows, at a crude level, what one might expect the relative design costs to be for various inter- and intra-IC design methods. Even the design cost is highly variable depending on the project size, its goals, manufacturing volumes expected, and more importantly on the computer aided design programs.

The lowest cost designs stay completely away from modifying the ICs, except for programming read only memories. There are two elements to the cost of read only memories. One is the cost of the chips and the other is the cost of programming them. There are two kinds of read only memories, the field programmable read only memory (PROM) and the factory programmable (masked) read only memory (ROM). PROM chips have a higher initial cost than ROMs but provide some inventory advantages in a manufacturing environment because a common stock of unprogrammed parts can be divided up into various programmed parts rather than stocking a full supply of each required part. In many high volume applications, however, there is no substitute for the price advantage {offered by masked ROMs. At the RT level, the standard microprogramming design {method is (conservatively) only twice as expensive per instruction as {conventional programming, but likely on the order of 5 to 10 times as {expensive to solve the same problem that a program written for a {microprocessor solves *Some times the* ~~(the speeds are at least a factor of 10 more too).~~ *obtains in the program*
{Given that one must design controllers with ROMs and PLAs, the cost can be

{still higher, but lower than with the standard sequential circuits we used in
{the first few generations--particularly if the module etch is used for the
{interconnection structure (see also comments about interface design for the
{18-bit computers, page 00).

Design of ICs (Intra-IC Design)

Despite the prospects of higher design cost with custom ICs than standard ICs, and, in some cases, higher manufacturing cost, there are numerous reasons why a designer is often forced to design ICs. In some engineering environments where there are extremely small space, low power, or extremely high reliability requirements, the engineer is forced into building ICs, without attendant cost savings. Another motivating factor for custom IC design is to obtain high performance in high volume products. As was pointed out in mentioning Seymour Cray's computers, he chose not to design custom ICs, and obtained greater design flexibility, lower design costs, and faster design turnaround at the expense of special high density packaging and freon cooling to obtain the desired performance. Both IBM and Amdahl chose to make custom ICs instead. Their computers do not run as fast, but their performance is very impressive and they build many more computers more cheaply. Therefore, the added complexity of ICs may be the only way that a high volume product at a given performance can be obtained. The use of custom ICs to reduce the number of discrete components or to reduce the total number of ICs in a machine also has a salutary effect on the reliability, since the reliability of a system is mostly a function of the man-made explicit connections, including the bond to the semiconductor die. The failure rate for the IC die interconnections is so

low relative to the man-made connections that it can be ignored. Thus reliability is simply measured by counting discrete circuit pins, IC pins module and connector pins to determine the reliability.

To summarize, IC design is used along the same lines as other design styles. Custom ICs are built for performance and for reliability (it may be the only way that a complex design can be maintained). They are built to get some decreased cost and higher performance, permitting mid-range designs to be more cost effective. Finally, they are used to build very high volume, lower performance computer components (microprocessors and microcomputers) because it is the cheapest way to do a task. The secondary reasons, to get size and cost reduction for some design, occasionally enter in too. With greater semiconductor densities for the non-microcomputer components used to build more general digital systems, the increase in density is double edged--and may force specialization. That is, with these more complex components, there is a greater risk that as they become too large they will become less useful for building a particular system. This is akin to building with bricks that continue to grow in size. At some point, the brick size is well beyond the size of the object being built, and the only way to get the intended function is to sculpture the brick back to a useful form.

The various design methods that might be used for various objects and densities are given in the following table.

Table: IC design method versus semiconductor density.

IC Design Method	Density			
	SSI	MSI	LSI	VLSI
Minor variations in standard ckts. (high performance computers)	busses	RT components	Special interfaces - (e.g., UART)	
Gate arrays	-	Set useful for integrating a large system (e.g., a computer)		possible alt, to custom design
Standard cells	-	Small system relatively high volume system		"
Custom design (for high volume parts)	bus interface, signal convers.	high perf.	memories, microcomputers including peripherals	

The most straightforward intra-IC design method is to modify an existing design. This has been used extensively to get the components for implementing computers. If this approach cannot be used, the next most straightforward method is to use arrays of gates and interconnect them to form the desired function. Design with gate arrays occurs in a completely defined environment because there is only one circuit from which the gate is formed and the gate can be completely parameterized and defined. Furthermore, interconnection is a well understood aspect of logic design and is used to form the more complex macro structures (various flip flop types, adders) and then to form the higher levels of design by using arrays of gate arrays.

It should be noted that gate array design is not a new idea brought about by the need for a simple method of customizing LSI. Rather, it was one of the design philosophies advocated in the first few generations. The concept then was to have a single module containing a set of gates, and all subsequent logical design would be done in terms of that module. For example, flip flops would be constructed by interconnecting the gates. Note that a design predicated on a single module type simplifies the spare stocking and servicing aspects immensely, and it is possible to troubleshoot a problem by simply replacing circuits according to a pattern.

Gate array design uses a single well-understood, well-defined component and the fabrication of all but the last few semiconductor processing steps is identical for all designs. The interconnection of the gates by metal is carried out last. Gate array design methods do not permit the high density possible with the more custom methods such as the standard cell design methods. Standard cell design is identical to the logical design of the first few generations since there is a well-defined set of components (AND gates, flip flops) in which the design is carried out. The advantage of the standard cell design methods is that higher densities than those possible with gate arrays can be achieved. However, each cell occupies a different space and hence the improvement in packing density may not be as substantial as direct comparisons between gate array technology and standard cell technology might at first indicate. In addition, there are a large number of circuit types and the set may not be characterized well enough to be reliable.

A representative gate array is a Raytheon RA-116. It has 300 Schottky TTL

gates, of three configurations:

120 internal driver gates	(3-input NAND)
60 external driver gates	(4-input NAND)
120 internal expansion gates	(7-input NAND
	or
	2-input OR expanders)

The gates have a typical propagation delay of 5-6 nanoseconds and dissipate 5.5-6 milliwatts per driver and 1 milliwatt per OR expander. Two metal layers are used for tailoring, and the resulting circuitry can be connected to the outside world by means of 64 external pins.

Since the designer can arbitrarily interconnect, he constructs flip flops, adders, decoders, etc. Because the use of IC gate arrays is recent, data on package count reduction is scarce, but one informal study for the Raytheon RP-16 aerospace computer measured a factor-of-three improvement.

For higher speed applications, an ECL gate array has been proposed. This device would exploit the inherent properties of current mode logic to obtain a particularly flexible element [Gaskill et al., 1976].

The most common form of intra-IC design is custom design. It is in some ways a variant of the standard cell, since a designers typically have a set of favorite circuits which they interconnect to create designs for specified applications. With custom design the designer can (theoretically) specify a

circuit for each use within a particular logical design. For example, upon observing that a particular gate or flip flop only drives a certain load, the designer can modify that gate or flip-flop to provide only the appropriate driving capability. Therefore, with custom design, the whole IC can theoretically be an optimum, since each part is no larger than it need be. The advantages are clearly size, cost, and speed. The design costs are high because each part can, in principle, be customized. The quality of the circuit design is totally dependent on the single designer who must analyze each circuit geometry in terms of his expectation of performance, operating margins, etc. To the extent that this analysis is carried out, the circuit is clearly optimal. Design of this type, where both circuit and logical design are varied for the context, rarely occurred in the first few generations.

Also on the graph is a hypothetical line for universal logic arrays. For at least 15 years, academicians have studied the possibility of designing a single array of logical design elements, or a collection of such arrays, that could be interconnected on a custom basis to carry out a given function. The gate array can be looked at as the simplest example of this type of design. While many are skeptical that such a device exists, a line representing it is placed on the graph as a target for those who search for the one truly universal logic array.

Both Read Only Memory (ROM) and Programmable Read Only Memory (PROM) are commonly used, but trivial, forms of the truly universal arrays, because they can be used in a table look up fashion to create several functions of a number of input variables. For example, a 1,024 word ROM arranged in a 256 x 4-bit

fashion can generate 4 independent functions of 8 variables. This is a distinct alternative for using a conventional gate structure to carry out combinational functions. A disadvantage of this method is that the required ROM size doubles for each additional input variable.

The PLA is a combinational circuit which remedies the disadvantages of the ROM implementation of combinatorial functions by allowing the use of product terms rather than completely decoding the input variables. Fig. PLA shows a typical circuit, which consists of separate AND and OR arrays. Inputs are connected to the AND array, and outputs are drawn from the OR array. Each row in the PLA can implement an AND function of selected inputs or their complements, thus forming a boolean product term, and the OR array can combine the product terms to implement any boolean function.

A simple application is operation-code decoding. For the PDP-11, the 16-bit Instruction Register could be directly connected to a PLA and the output thereof used to specify the address of the microprogram that executed that instruction. Three different types of operation-code decoding are customarily applied to PDP-11 instructions: source mode decoding, destination mode decoding, and instruction decoding. With a PLA implementation, a PLA could be used for each of these decoding operations, and only three chips would be required. A ROM implementation, on the other hand, would require 128x8 for address mode decoding and 64Kx8 for instruction decoding. Using 2Kx8 ROM's, 33 chips would be required. For this reason modern minicomputers, such as the PDP-11/34 use PLAs rather than ROMs or combinatorial logic for instruction decoding. The technique is also extended downward into microcomputers such as

the LSI-11, where PLAs are used to conserve the die area used by their control units.

The PLA becomes an even more useful building block when it is made field programmable -- the FPLA. The programmable connectors shown in Figure PLA are fusible nichrome links.

When a register is added to the outputs of the PLA and incorporated in the same integrated circuit, a simple sequential machine is obtained in one package. Since register circuits are pin intensive, adding registers to PLAs or ROMs permits about a factor-of-two package count reduction in typical applications and has been undertaken by IC manufacturers (Signetics 82Sxxx) as soon as i.c. density improvements permit.

The first PLAs had propagation times of the order of 150 nanosec [check with Spencer] and were thus suitable building blocks for slow, low-cost computers. Propagation times of 45 nanoseconds are quite common today, and the PLA is more widely used. A candidate application with these higher speed components is the replacement of the SSI and MSI packages used to implement the control logic for Unibus arbitration.

A more complex application than instruction decoding has been documented in (Logue et al., 1975). An IBM 7441 Buffered Terminal Control Unit was implemented using PLAs and compared with an SSI/MSI version. The PLA design included two sets of registers fed by the OR array (PLA outputs): one set fed back to the AND array (PLA inputs), and the other set held the PLA outputs. A

factor-of-two reduction in printed circuit board count was obtained with the PLA version. The seven PLA's used in the design replaced 85% of the circuits in the SSI/MSI version. Of these circuits 48% were combinational logic, and 52% were sequential logic.

MEMORY TECHNOLOGY (AND SEMICONDUCTORS USED AS MEMORIES)

The previous section discussed the use of memory for microprogramming and table look up in logical design, but that is not the principal use of memory in the computer industry. Rather, the more typical use is within a hierarchy to store information both on a short term basis while a program runs and on a longer term basis as permanent files. This section will present the various parameters and discussion relevant to this use. While the principal focus will be on core and semiconductor memories, slower speed electromechanical memories (drums, disks, and tapes) will be considered superficially, as their performance and price improvements have pushed the computer evolution. Since the typical uses for memory usually require read and write capabilities, write once or read only memory such as video disks will be excluded from the discussion.

It should be possible to discuss memory using a minimal number of measurement parameters because memory is the simplest of components. Many of the parameters are time variants, particularly price, which has declined at a rate of 30% per year compound, (which amounts to over 50% in two years). The price is expressed only as price/bit, but it is important to know the price (or size) of the total memory system for which that price applies because of the

economy of scale. In order to get the lowest price per bit, a user may be forced to a large system. (Is there economy of scale? What is the smallest or largest memory that can be built with the technology?)

Performance for electromechanical memories is expressed in two parameters: the number of bits that can be accessed per second after a transfer begins; and time to access the start of a block of memory.

The operational environmental parameters of power consumption, temperature, space and weight effect the utility of memories in various applications, and the reliability measures are needed in order to see how much redundancy must be placed in the memory to operate at a given level of availability and data integrity.

In summary, the relevant parameters for a given memory are:

1. state of development of the technology at the time the measurements are taken relative to the likely life span of the technology
2. price per bit
3. total memory size or total memory price
4. performance
 - a. the access time to the first word of the block
 - b. the time to transfer each word (data-rate) in the block

5. operational power, temperature, space, weight

6. reliability and repairability

A good example of a technology that is young relative to its total lifetime is semiconductor memory. Figure Memprice gives past prices and expected future prices of semiconductor memory. These memories have declined in price every two years by a factor of two, and that rate of decline is expected to continue well into the 80's because of continued increases in semiconductor densities. Figure Memsizeperf, a graph by Dean Toome, Vice President of Engineering for Texas Instruments, shows memory size and performance improvements with time and includes Charge Coupled Devices (CCDs) and magnetic bubbles. These latter devices show slower performance figures than the other semiconductor memories because they are cyclically accessed in a fashion similar to disks.

While these graphs of semiconductor prices and performance permit an understanding of trends in the principal use areas for these devices, processors, primary memory, cache, and small paging memories, additional information is needed for disk and tape memory in order to complete the memory hierarchy. (I'd like graphs here of:

- . Price/byte for 1 platter, 4 platters, 10 platters

- . Price per disk for 1 platter, 4 platters, 10 platters

- . Tape units at each speed 12/45/75/125/200

. Tape density and transfer rate vs time for each speed

. Price/byte of small systems: DECTape, floppy, cassettes

{All information processing systems appear to be associated with relatively
{local memory use. As time passes, the location of the computation changes,
{but still the amount and the location of memory involved in a computation
{over a relatively long time, is comparatively small.

{This phenomenon has been observed to apply to a number of activities within a
{computer system, and in each case the principle of operation appears to be
{the same. Thus, armed with the observation, a new memory--providing
{proportionally more memory for less, should be useful if it is:

{cheaper and slower than an existing memory; or
{it is faster and more expensive (implying that it is smaller).

{Of course, any memory that is better in all dimensions of speed, size and
{cost automatically replaces an existing memory.

Transparency is another consideration about the utility of a given physical
memory within a hierarchy that effects use. At the language level, nearly all
memory technologies are transparent, and the programmer does not have to
consider the characteristics of the memory when writing programs. There are
some exceptions, however. For example, the original FORTRAN had explicit
statements to read and write tapes, and the modern counterpart has operations

to access files in different access arrangements; hence, these operational improvements are not quite transparent. However, whether a processor has zero, one, or sixteen is irrelevant to the FORTRAN programmer.

The following table gives the memory hierarchy as currently known. There is a continuum based on need together with memory technology size, cost, and performance parameters.

Table: Computer System Memory Component and Technology

Part	Transparency (to machine language programs)	Based-on
Microprogram memory	yes ¹	very fast
Processor-state	no	very small, very fast register set (e.g., 16 words)
Alternative processor- state context	yes	same (so speed up processor context swaps)
Cache memory	yes	fast. used in larger machines for speed
Program mapping and segmentation	yes	small number of association, or large map
Primary (program) memory	no	relatively fast, and large ¹ depending on Pc speed
Paging memory	yes	can be electromechanical, e.g., drum, fixed head disk, or moving head disk. Can be CCD or bubbles.

Local file memory	no	usually moving head disk, relatively slow, low cost
Archival files memory	yes (preferably)	very slow, very cheap to permit information to be kept forever

Nearly every part of the hierarchy can be observed in the computers in this book. Chapter 0, 0, and 0 describe PDP-11 implementations than use microprogrammed memories. These memories are transparent to the user, except in machines such as the PDP-11/60 which provide user microprogramming via a writeable control store. Mudge (chapter 0) describes the writeable control storage user aspects associated with the 11/60 and the user microprogramming. Chapter 0 describes similar possibilities in the LSI-11, although the writeable control store option was not at the time the article was written.

{In principle, it is possible to have a cache to hold user micrograms, {hence, there could be another level to the hierarchy. Without memory of this {size and speed it is not possible to build microprogrammed machines that are {cost effective with hardwired systems. Mudge also discusses this aspect and {the technology tradeoff. In retrospect, the small, read only memory for {PDP-9 in 196? was adequate and could have been enhanced to be used in later {machines. This would have permitted earlier entry of more complex ISPs at {lower costs. (Similarly this ROM could have been used to implement lower {cost PDP-10s earlier.)

To the machine language program, the number of registers in the processor state is the most non-transparent part of the architecture. This number is solely dictated by the availability of fast access, low cost registers. It is also occasionally the means of segmenting architectures (e.g., single

accumulator based, general register based and stack based).

In 1964, even though registers were not available in single IC packages, the PDP-6 adopted the general register structure because the register was only a small part of the system cost (see chapter 00). In the chapter on the DECsystem 10 there is a discussion of whether the architecture should use general registers in an explicit (transparent) fashion, or whether the stack architecture should be used (also requiring a number of registers), but the use is transparent to the programmer. We adopted the general register structure to give better program control of a small number of local variables (i.e., locality) and permit the performance advantages. The change in register use can be observed between the 12-bit and 18-bit designs and the later DECsystem 10 and PDP-11 designs.

As the number of registers increased, and technology improved, the processor state storage was increased to provide multiple sets of registers to improve process context switching time. In this way several multiple programs could be active at a time and selected on the basis of interrupt urgency, thus providing better real time and multiprogramming response.

In the late 60s, the cache memory was introduced for large scale computers. This concept was applied to the KL10 and 11/70 in 1975 when the relatively large (1 Kbit), relatively fast (factor of 5), memory chip was introduced. The cache is discussed extensively in chapters 00, 00 and 00. It derives much power by the fact that it is an automatic mechanism and hence transparent to the user. It is the best example of the use of the principle of memory

{locality.

A similar memory circuit is required to manage (map) multiprogrammed systems by providing relocation and protection among various user programs. The requirements are similar to the cache and may be incorporated in the caching structure. The KI 10 used an associative memory for this mapping function.

The Atlas computer (Kilburn et al 1962) was designed to have a single, one level large memory and paging drum. This structure ultimately evolved so that multiple users could each have a large virtual address and virtual machine. However, the concept of paging mechanism works because there is not equally random access to each page, but rather only local access to various parts of a program by the processor at a given time. Denning pointed out the clustering of pages for a given program at a given time and called this phenomenon the working set [19]. For most programs the number of pages accessed locally is small compared with the total program size. Initially a magnetic drum was used to implement the paging memory, but as disk technology began to dominate the drum, both fixed head and moving head disks (backed up with larger primary memories) were used as the paging memories. In the next few years, the relatively faster and cheaper CCD semiconductor memories and bubble memories are clearly the candidates for paging memories.

For medium sized to large scale systems there is no alternative to disks, with archival files on magnetic tapes. These permit files to be stored cheaply on an indefinite basis. For smaller systems there are usually fewer memory technologies used than in larger systems, because the smaller systems cannot

afford the overhead costs (disk drives, tape drives, etc.) associated with the various technologies. At most, two levels of storage would probably exist as separate entities.

Alternatively, one might expect a combination of floppy disk, low cost tape, and magnetic bubbles to be used to reduce the primary memory size and at the same time provide file and archival memory. Currently the floppy disk operates as a single level memory. Here one can see two alternatives for technology tradeoff using the hierarchy: a tape or floppy disk can be used to provide removability, and archivability, whereas bubbles provide performance.

MEASURING (AND CREATING) TECHNOLOGY PROGRESS

The previous sections have presented technology in terms of exponentially decreasing prices and/or exponentially increasing performance. This section presents a basis for this constant change. The metric of technology, $T(t)$ at some time, t , has been classically observed to be just:

$$T(t) = K \times e^{ct}$$

This can be converted to a yearly improvement rate, r , by changing the base of the exponential to:

$$T(t) = T \times r^{t-t_0}$$

where T = the base technology at t_0

and r = yearly increase (or decrease) in the technology metric

This is the same form used for declining (or increasing) cost from base c

$$C = c \times r^{t-t_0}$$

The questions that may arise upon studying the previous graphs (i.e., semiconductor prices) are:

1. Under what conditions does cost decrease exponentially?
2. Under what conditions does technology improve (i.e., in performance) exponentially?
3. When does a technology reach a limit of improvement?

Clearly there are manufactured goods that neither improve nor decrease in price exponentially, although many presumably could with the proper design and manufacturing tooling investments. The notion of price decline is completely tied to the cumulative learning curves of a) people building a product for a long time, b) process improvement based on learning to build it better, and c) design improvement by engineers based on learning from the history of design. Production learning per se is inadequate to drive cost and prices down because after an extremely long time in production, a few more units contribute little to learning. With inflation in labor costs, the costs actually rise when the learning is flat. In order to provide a base for predicting the inflationary

effect, the consumer price index has been plotted Figs. CPI and CPI.log.

Learning curves don't appear to be understood beyond intuition. They are (empirical) observations that the amount of human energy, E_n , required to produce the n^{th} item is:

$$E_n = K \times n^d$$

where K and d are "learning constants". Thus, by producing more items, the repetitive nature of a task causes learning, and hence the time (and perhaps cost) to produce an item decreases with the number produced and not with the calendar time an object is produced. Fusfeld [] conjectures that the technology of the i^{th} unit produced also improves exponentially with the number produced just as in the case of the learning curves. Thus, using the technology measure:

$$T_i = a \times i^b$$

he found the following technology progress constants:

Item	Measure, T_i	Quantity produced (i)	Technology progress (b)	Change observed in study	Total change
light bulbs	lumens/bulb	10^{10}	.04;.19	33	80
automobiles	vehicle h.p.	$3 \times 10^7; 10^8$.11;.74	10	6;13
titanium	p.s.i./\$/16	3×10^8	.3;1;1.04	10	350
aircraft	max.speed	2×10^5	.33-1.2	6	56
turbojet engines	fuel consumed, weight	1.6×10^4	1.06	2	2.9×10^4
computers	mem.size x	10^5	2.51	10^9	3.5×10^{12}

rate

Computer technology (until 197?) evolved substantially more than any other technology, and recent advances suggest this has continued to a slightly lesser degree. This has occurred in part because in computer technology there are so many materials to store, transmit, and process information about, nearly all of which are electronic based. In the above technologies, most are mechanically oriented with associated physical limits. In essence the table is comparing systems constrained by Newton's Laws with those determined by Maxwell's Equations.

Fusfeld also showed that if the number of items produced increases exponentially (with time):

$$i = e^{c/b \times t},$$

then calendar time and units produced can be used interchangeably.

There has been exponential growth of computer systems, so calendar time can be used instead of units. Time is an easier and more accurate method to measure than learning curves based on units.

The question of why the cost declines exponentially can be conjectured by using Fusfeld's observation that it is because of learning curves and the exponential increase in quantity produced. Furthermore, this exponential increase raises a fourth question;

4. Why is the demand exponential?

The demand or quantity, q , sold per unit time is completely elastic, (exponential), according to the expression:

$$q = k/\text{price}$$

This creates a positive feedback market system whereby decreasing prices increase demand exponentially causing decreased costs (through learning) which support the decreasing prices as shown in Fig. Mkt.cycle (a variant of Fig. 1.).

There has been no attempt to answer question 2 of why technology improves exponentially nor is the answer for why cost declines exponentially at all satisfactory. Simple production learning does not account for the rapid technology changes in the integrated circuit, for example, where totally different production processes have been evolved to support the greater technology. It appears best to simply observe that the three situations have been true, and can be extrapolated to hold over the next few years because one can see ways by which each limit can be overcome.

Technology historians (von Hippel, 197?, Fusfeld 197?) and futurists have made a number of studies and observations about technology progress:

1. Quantity produced is simply a good dummy variable to measure improvement.
(With exponential increasing sales, time can be used as the dummy

variable.)

2. Technical problem solving is correlated with business activity. Inventors tend to be stimulated by sales and slacken efforts when sales are low. (This is a counter-initiative observation.)

3. Production alone does not stimulate innovation. A lesser number of inventions are stimulated by production needs. Of these, the same user-supplier relationship is the best framework. The users of equipment (the producer for the end product) stimulate the production equipment suppliers.

4. Major innovations come from use (sales). This is opposed to innovation arising from a technical possibility for which use is subsequently discovered. In the case of semiconductor technology, the computer designer (user) is responsible for many of the design innovations. Computers also evolve rapidly under this and the levels of integration model because the suppliers are also significant users. The problem of market coupling is diminished significantly. Alternatively with computers, the users write (develop) the applications programs; hence again the user and developer are one in the same--this stimulates use!

The gains in packaging (i.e., printed circuit boards, backplanes, and cables) have been driven by production, versus product functional needs. Testing technology of all types has been motivated solely for production needs. These change least.

The cost of computing is the sum of costs which correspond to the various levels of integration we described in view 4, page 00 plus the operational costs. In actual practice, each layer, or additional level-of-integration is often looked at as overhead. Using standard accounting practice, the basic hardware cost, at the inner layer, is then multiplied by an overhead factor at each subsequent outer level. While this may in principle work operationally, for a stable set of technologies it is hard to condone and we will describe the constituent technologies and resulting structures at the next level together with operational cost models. An overhead-based model will not adequately allow for rapidly evolving technologies or the elimination of levels, for example. By examining each part, as we have tried to do in this section on technology and in the packaging section, we can then make observations about the use and substitution of technology. More importantly, we can draw conclusions about how structures are likely to evolve.