Chapter 2 Technology, Packaging and Manufacturing

TECHNOLOGY PUSH 1/4/78

In this section we discuss how the important technological advances and how they have pushed the evolution of computer systems. The examples of the push are given in this book, but here we try to define the individual components so the reader can observe each push within the context of use. We must have a model of what the technology has been (and its position in the immediate future). whereas the rest of the book gives examples of the effects of technology push, in this section we define the individual components within the context of use. As users, we so that short-term extrapolations are possible.

We should, for completeness, discuss all technologies (for example a tree of the relevant memory and logic technology is given in Fig. Memtax). Instead we only concentrate on semiconductors as they have been the dominant mover for the computer to change. 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 elsewhere. These do not represent pushes but rather are large, high inertia objects that have to be pushed against. In fact, the main gains in these areas have come about by eliminating or reducing the need, not by evolving the technology rapidly so that it can help evolve computers.

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

Ray Jane,
Louise

Here are the changes for Chapter 2
– the original was under Chapter 1 (or zero?).

Please separate the chapter from Ch 1.

When you have finished, please give the text plus the attached figures to John McNamara.

Thanks,
Craig

parameters are also briefly discussed.

Following the semiconductor evolution model, we discuss how the semiconductors have been applied using various logical design methods and how they have pushed the methods. Of special interest is how and whether users design ICs, 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, the belief is that there is a need to have access to lower level building blocks when building many large objects. We advise using bricks and mortar, not finding a sand pit, cement and brick factors to fabricate bricks and mix mortar.

There are two sections on memories. The core memory provides a number of lessons on managing evolving technology; 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. 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.

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 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, 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 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. ~~For 1980 technology, this set of functions constitute a two level hierarchy as described in views 1 and 3.~~

2. The number of primitive digital switching circuits (or transistors) per IC. This density is a measure of the process capability ~~of~~ of the IC.

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 usually determines this performance.

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, 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 single printed circuit board module of the first and second generations to occupy a single IC. This forced modules to take on specialized functions. The drastic reduction in size from the second to the third generation transition can be most vividly, by example in the 18-bit 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 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

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 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 the gate array structure, an alternative for doing logical design including sequential circuits. Gate arrays are simply a large number of gates placed on the chip in fixed locations which can be interconnected together using metalization done in the final stages of semiconductor manufacture.

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 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).

The remainder of the interesting logical functions include combinations of logic and memory. There are various special, functions like encoding algorithms (e.g., LPC, for Linear Predictive Coding) to use in real time and communications system. 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.

There is a special section on bit-slice components (actually sets of 1, 2, 4, 8 and 16-bits). Again, we claim to be a party to this evolution, as the Register Transfer Modules are about the first example of forming complex, register level building blocks. Now, mostly all high-speed digital systems, mid-range computers, and computer peripherals are constructed using the bit-slice approach. Although there have been several bit slice families, the AMD 2900 series has become the de facto standard. Several register transfer diagrams show the AMD system (Fig. AMD). Note that all the primitives of this series were present in the RTM family including the microprogrammed control unit, we called the K(PCS) for Programmed Control Sequencer). In the case of Fairchild's macrologic, the person responsible for the family, Kris Rapapielli claims that RTMs were the archetype antecedent.

The final branch of the tree is the most complex, and is used to mark the fourth and predictably fifth generations of technology. These branches show the microprocessor and microcomputer. The fourth generation is marked by a complete processor being packaged on a single silicon die, and the fifth generation, by this measure, has occurred since a complete computer (processor with memory) occupy a single die. The evolution in complexity simply permits larger word length computers to be placed on 1 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 can be placed on a chip. The LSI-11 (Chapter 00) was designed at about the middle of the fourth generation and hence, 4 chips were required for its implementation.

IC Density

Although we consider the circuit function to be an attribute that's separated from others, it is clearly dependent on the number of circuits which can be placed on a chip. Thus, circuit density in circuits/chip is the single most important parameter. 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., 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. 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.

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

An Operational Model for Memory Size versus Time

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

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

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:

~~[specific availabilities]~~

Year when first widely available	Number of bits
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

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 a commodity. For more 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. As users, we consider ICs 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 are) and the manufacturing volume.

Two curves are presented to reflect price of various components/IC. Figure Compprice 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. We note that 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 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:

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

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. 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

Type	Meaning	Evolution	Use
TTL	Transistor-Transistor Logic	-	logic, bus interfacing
		TTL/Schottky	high performance
		TTL/LS	low power, relatively same as TTL
ECL	Emitter Coupled Logic	MECL II, III	original, very high performance
		MECL 10K, 100K	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, noise immunity

Note that we have omitted some of the lesser used technologies such I^2L--for Integrated-Injection Logic and SOS--for Silicon on Saphire, which both promise higher speed and at lower power than MOS, and hence might displace[1] its use. 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 cummulative 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).

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. 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 we used (e.g., the PDP-8). Similarly, these early circuits were packaged in a flat package, not the dual in-line package used

[1]Semiconductors 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!

today, hence machine insertion of components was not possible.

The table of parts, above, is weighted heavily to DECs 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 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. However this is a bit misleading because unless special cooling is used, only about one watt can be dissipated by an IC package. 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 (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 interaction with the rest of the world. Memories are especially easily adapted to live with pin limitations since only 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

Insert 8

[needs some words]

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

~~Figure~~

8

Over the past 15 years, the failure rate for standard ICs has been reduced by two orders of magnitude to the neighbourhood 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 MTTF of between 10^8 and 10^9 is obtained at a cost increase of 4 to 100 times per component.

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, leads 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, e.g., 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 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 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.

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. ~~This~~ Some of this customization is required in nearly

With more circuits per IC, providing LSI and VLSI, the user is faced with the dilemma of using standard circuits that may not be appropriate to the task versus designing a new IC.

Bottom of page 14

~~Fortunately only a few designers have to work on specialized chips, but nearly~~ all digital system design of the fourth and fifth generation ~~requires some customization.~~ At the very minimum, the values for a read only memory have to

building block	technique for varying function	degree of gen-erality	permanence of change
~~CM~~ Computer module	program	v. high	none
microprocessor	program	high	low to med
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	med	irreversible
FPLA	field change	med	irreversible
gate array	factory mask change	med	irreversible
RAM	write	v. high	none

Table funvar

be set. Table funvar contrasts the building blocks available in the fourth generation.

We believe that logic design as it was practiced in the first through early fourth generations is essentially passe! The PDP-8 design, though a second generation design is given in a complete fashion, page 00 characterizes this style.

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.

~~The following table shows the evolution of design methods with generation.~~

Table: Design Method versus Generation

Design Method:\Generation:	First	Second	Third	Fourth	Fifth	Examples in this book
Combinational and sequential; use of "standard" modules, IC's[5].	s	s	s			18-bit; PDP-8
Read only memory and PLA; microprogramming[2,3,4]			s	m		PDP-9; PDP-11
Microprogramming with standard RT elements (high perf.) minor logical design[3]				s	m	CMU-11
Programming using micros[4] and logic for interfaces		p	p	s	x	LSI-11

single space please

PMS design using completely[5] specified and pre-designed microcomputer components			s	Cm* (almost)
Customized chip design[4] 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

1 see 18-bit and PDP-8 chapters.

2 see PDP-9 desing and PDP-11 implementations.

3 see Part 00 on Register Transfer Modules; Fig. AMD, page 00.

4 see LSI-11 design, chapter 00.

5 no present designs. Design is akin to configuring a mini to a task except that a different packaging scheme is used.

Toward the late third generation the change began to manifest itself as the designer began to switch to some form of microprogramming--i.e., ROM based design. Other uses of ROM was made including table look-up and functions of a few variables. The Programmable Logic Array was also introduced for similar use. At every minimum, ROMs were used to do character generation and as constants in arithmetic units. The various 11 implementations characterize

ROM-based design. The design is also predicated on standard arithmetic units ICs and small arrays of registers (see pages 00 and 00).

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 these bit-slice components Figure AMD shows the current standard RT modules set; we would expect this series 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 are custom ECL circuits to improve packaging. The customization aspect will be described below. ¶ 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 or will be soon converted completely to computer design. The process of disolving a particular problem (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

programming in the same way that it has evolved since the late 40's. In fact, we have seen three comlete 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).

We believe the design of systems as currently practiced will be relatively short lived. The design method here will be at the PMS-level component. 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 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 thereby having a standard method for interconnection, higher volume and lower costs will result.

Before we discuss the alternatives associated with IC design, it is important to characterize the various costs. Figure Design.cost shows, at a crude level, what 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 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 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. The purely product specifications, and cost, mostly in the province of the engineer, are to be traded off against higher engineering cost. In some engineering environments where there are extremely small space, low power, or extremely high reliability requirements, the engineer is perhaps 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 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 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 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 module[1], 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

Gate array

A primitive which is more powerful than the PLA, because it permits essentially arbitrary interconnect patterns, is the gate array. However, the method for changing the pattern is less flexible; at this point field changes are not possible - the changes are made by changing the masks at the factory.

Section α

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 manoseconds 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 ~~packet~~ package count reduction is scarce. One informal study we know of (Raytheon ~~16-bit~~ for the RP-16 aerospace computer) had a measured factor-of-three improvement ~~[CM: check with Nat Parke]~~.

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].

PMS-level design with multi-microcomputers

Whereas RTM's provide building blocks for system designers to work at the RT level, CM's provide blocks at the PMS level. The argument for using Cms is based on the trends to higher & higher circuit complexity in the LSI industry. Given the products which have appeared since the original Cm proposal (1973), the argument is even stronger. Before long the cheapest integrated circuit available may well be a computer on a chip! The challenge to designers & researchers is therefore to understand what communication structures are needed to interconnect these building blocks.

to make all modules different from one another and specialized to each task.

Gate array design uses a single well-understoood, 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 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.

Insert section d

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

[1] In 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.

RECENT TECHNIQUES FOR VARYING FUNCTION

PLAs

Section B

Read-only memories can be used to implement any logic function by table look-up. However, a disadvantage of this technique 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 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 fleasible 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.

The first PLAs had propagation times of the order of 150 nanosec [CM: 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 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.

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 occured 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 at as the ~~most trivial~~ 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.

Insert Section B

The PLA--for Programmable Logic Array is another basically combinational circuit that can be customized to permit logical design to be carried out within the IC so that a user may take advantage of the increased semiconductor

[1]In 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.

density. The PLA is an array of AND gates that can be connected to a common set of inputs (and their complements) to form product terms that are OR'd to a second level of OR gates on the same semiconductor chip, as shown in Fig. PLA. The complexity and ability of the circuit are severely pin limited, but it does permit sparse functions (i.e., only a few terms) of several variables to be formed easily. It is ideal for use in microprogrammed applications which require generating few output variables by looking at a number of terms. This function is identical to serving as a content-addressable memory. The number of input and output variables is determined by the pins. The array is fixed at design time by the number of AND and OR gates that can be interconnected. Since the functions can be programmed either as a special pattern (mask) in the factory, or in the field by the user (FPLA version) there is a high degree of customization without the attendant specialized costs.

Examples of Processor Design with Generations

The best way to observe changes in the design methods is to observe the resulting designs, relying on the diagrams in the rest of the book. The two design methods of interest are the ad hoc register transfer structures have a great deal of inherent parallelism. A bit of the PDP-8 accumulator is given together with the control, state diagram and the RT structure, hence it's possible to reconstruct the machine family easily and understand the degree of parallelism in the control.

The 18-bit chapter also provides an example of data path evolution. Although the PDP-1 data path is not presented explicitly, it can be built up easily, by using the modules and adder example of chapter 3. The PDP-4 used the same

design methods, though different circuits and integrated a single bit per module and the data path is given on page 00. The PDP-9 was DEC's first microprogrammed controlled computer and the reader can see the resulting structural change and simpler data path. By having a single shared data path, the designer is forced into more sequential control and lower perforrmance, but gains in lower cost and simplicity.

All the PDP-11s use a shared data path as described in chapter 00, and to obtain higher performance requires a method of getting parallel access to the registers (also described in the chapter). The chapter on the 11/60 gives another view of data paths, showing the necessary increases in complexity when both generality and performance are goals. Even within the CMOS-8 there is a shared data path as shown on page 00, even though the implementation is non-microprogrammed.

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 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 econcomy 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:

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 omited, 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 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 languge 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 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

data memory

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 machine language level user unless user microprogramming is provided. The use of memory for this application can be seen in the 11 implementations (chapters 0, 0, and 0). Mudge (chapter 0) describes the writeable control storage user aspects associated with the 11/60 and the user microprogramming. This can also be observed in the LSI-11 (chapter 0) although this option wasn't offered when the article was written. In principle, it is possible to have a cache to hold user microprograms, 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., 1 accumulator, general registers 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 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 ~~not~~ 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 use of semiconductors for the primary memory, replacing core memory is legendary. Section 00 discusses this evolution and tradeoff.

~~As an alternative to using less primary memory,~~ 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. 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 memmories.

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.

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:

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

where T_o = the base technology at t_o

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

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

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

$$En = K \times n^{d}$$

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:

$$Ti = a \times i^{b}$$

he found the following technology progress constants:

Item	Measure, Ti	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.	$3x10^{7}$;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 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?

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

$$q = k/price$$

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.)

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.

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 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:

system performance = k x min(memory-size, memory performance)

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.

   performance = k x price^{s}; s>1

2. Linear price performance relationship.

   performance = k x price

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

   performance = k

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

performance = n x (k x price)

Economy of scale holds where it might not otherwise operate because of several effects in the design and product introducion 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 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 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.)

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 sequented (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.

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.

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 vaious technologies: disks, computers, more software (e.g., language), and semiconductor process.

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 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).

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

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

$f = 1 - r^{dt}$ poor cost, expressed as 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