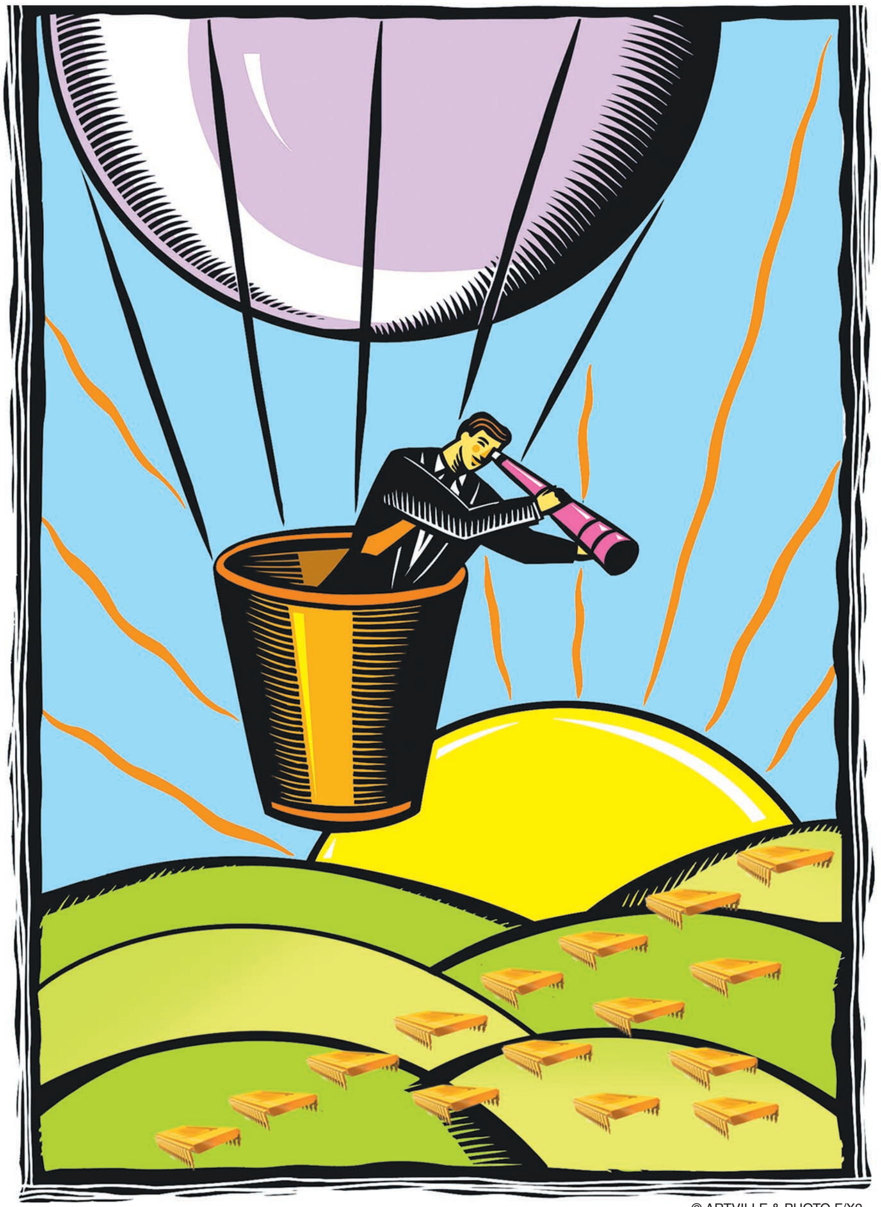
Moore's Law, Microcomputers, and Me

A confluence of skills made the microcomputer revolution possible: device design, process design, applications, and marketing.


© ARTVILLE & PHOTO F/X2

by Stanley Mazor

In 1960—ten years before Intel developed the first single-chip CPU (microcomputer central processing unit)—the revolution that would ensue was inconceivable: the cost of computing dropped by a factor of a million, modes of personal communication changed forever, and intelligent machines took over processes in manufacturing, transportation, medicine—virtually every aspect of our lives.

Certainly Moore's law—that the number of transistors on a chip doubles every year, later amended to every two years—is a dominant factor in this revolution. But at Intel, there were three other enabling conditions:

- a customer with a problem
- an applications engineering department that listened to the customer
- a talented engineering group to implement a solution.

Here I give my views on Moore's law and focus on the role of applications engineering in developing Intel's first microcomputer. (For an overview of basic chip technology, see "IC Backgrounder: Process and Design.")

Digital Object Identifier 10.1109/MSSC.2008.930943

I want to tell you about my experiences with Gordon Moore's law in Silicon Valley as an applications engineer using and designing digital computers, digital IC components, and microcomputers.

Although electrical engineering students study the *design* of logic and circuits, when they graduate, they find a large number of *nondesign* positions posted: quality engineer, process engineer, marketing engineer, and applications engineer. But what does an application engineer do? I want to tell you about my experiences with Gordon Moore's law in Silicon Valley (from 1964 to 1984) as an applications engineer using and designing digital computers, digital IC components, and microcomputers [1]. But first, let me contrast the change in the value of a computer over a 15-year period with the following personal anecdote.

Computer Values (1962–1977)

In 1962, Prof. Robert J. Levitt of the Math Department at San Francisco State University didn't let students play games on the campus's *only* computer, an IBM 1620 (Figure 1). "After all," he said, "it cost almost \$100k and could execute instructions in milliseconds. This valuable resource should neither be wasted nor made to do frivolous tasks." Accordingly, I played my own tic-tac-toe assembler language program only at night, when no one was watching; playing games on computers was not allowed.

Fifteen years later, I visited Midway Games, which had been acquired by Bally Technologies, a manufacturer of pinball machines and slot machines, and coordinated development of a custom Intel game chip for the Magnavox Odyssey2

IC BACKGROUNDER: PROCESS AND DESIGN

Chip Size and Cost

Although larger chips can accommodate more functions, a chip's manufacturing cost increases with the square of its die size. So the chip cost constrains the practical number of transistors that can be placed on a chip. But, as Gordon Moore observed in the mid-1960s, the practical limit doubles every year.

The early Intel microcomputers (circa 1970) contained about 2,000 transistors, organized into a few logic blocks—ALU, register array, instruction decoder, I/O pads, and so forth. As processing improved over the years, the individual transistor sizes shrank, and at the same time the chips grew bigger. As transistor features got smaller, the lower capacitance helped the circuits run faster.

MOS Process Technology

Modern complementary metal-oxide semiconductor (CMOS) ICs provide optimum power dissipation and speed, but the manufacturing process is complicated because two different transistor types are needed within the same chip: NMOS and PMOS, with n-type and p-type substrates, respectively. Standby power is reduced by pairing transistors, so that at any instant one of the transistors is turned off. NMOS devices are faster and smaller than PMOS devices because the carrier mobility of NMOS is about twice that of PMOS, but both transistor types are needed within a CMOS chip.

For lower manufacturing costs and simpler processing, either all NMOS or all PMOS ICs are preferred. Early semiconductor companies such as Electronic Arrays, General Instrument, and Texas Instruments produced ICs containing only PMOS transistors. NMOS required exceptionally clean fabrication facilities, as any impurities tend to permanently turn on NMOS transistors. Only IBM experienced early success with NMOS chip production. Early microprocessor chips were made with only PMOS transistors, but within ten years they were being made with all-NMOS transistors. After mastering both of these technologies, most semiconductor companies moved into CMOS production (about 1985), which required more photomasks and costlier production methods, but gave a better-performing product.

Power Dissipation

CMOS transistor pairs minimize the amount of power consumed in their steady or quiescent state, and lower operating voltages reduced the active power consumption. The combined effects reduced the power that has to be dissipated, thereby permitting more transistors on a chip. Process improvements such as ion implantation and polysilicon (rather than metal) gates lowered transistor threshold voltages and subsequent operating voltages. These processes were pioneered in the United States by: Fairchild, Intel, and Mostek. Operating voltages dropped from 14 V in 1970 to 5 V in 1975, and then to 1.5 V in 2000. Since power varies with the square of the voltage, reducing the operating voltage by a factor of 10 reduced power dissipation by a factor of 100. Early ceramic packages could handle about 1 W of power; plastic packages had a much lower rating.

Circuit Background

A CPU chip is characterized by its maximum clock frequency. A 100-MHz chip corresponds to a clock cycle period of 10 ns. The clock period in digital integrated circuits allows for signal propagation along the longest logic switching path within the chip's circuitry. This path typically goes from a flip/flop output, passes through a number of logic gates, and finally enters another flip/flop.

In typical MOS circuits, each logic gate's output transistor drives one or more transistor gate input loads. These loads are equivalent to an open circuit (high resistance with some stray capacitance to ground); there is no DC load. The switching speed of such a digital logic circuit depends on the voltage swing to be traversed, for example from .2 V (logic 0) to 4 V (logic 1), and home TV game system. This special video chip was used with an off-the-shelf single chip microcomputer to display game objects on a TV screen. In just 15 years, playing games had gone from being a frivolous use of computers to a big business [2]. Thanks to Moore's doubling, computers became cheap and new markets (for digital logic) appeared. (Today, Midway Games offers such popular video games as Mortal Kombat, *Ms. Pac-Man*, *Spy Hunter*, *Tron*, and NBA Jam.)

Moore's Law (1965)

I joined Fairchild Semiconductor in Mountain View, California, before Gordon Moore published his definitive paper [3], [4]. (Moore was in charge of Fairchild R&D.) I've lived with Moore's law from its beginning and have found it both difficult to ignore and difficult to grasp. The following story is enlightening: Legend has it that the man who saved a king's son's life was asked what he wanted as a reward. The hero asked that rice be placed on his checker board as follows: one grain on the first square, two grains on the second square, doubling on each successive square. The king agreed to this seemingly modest request, not realizing that the total would be more rice than there are grains of sand on the beach.

Similarly the significance of Moore's doubling of transistors each year is outside our normal expectations. Now after 40 years we have chips with *billions* (2^{30}) of transistors, a daunting result of this kind of exponential behavior. We live mostly in a linear world—miles/gallon, dollars/pound—and we just don't experience exponential relations (except perhaps for acceleration and compound interest) [5], [6].

Did anyone predict that "someday" the whole computer would fit on a single chip? Considering that a minicomputer (circa 1962) CPU needed about 16k transistors, and 16k = 2^{14}, one could have predicted a single-chip CPU after 14 years of Moore's doubling, or roughly in 1976. However there *weren't* any such predictions! Apparently Moore's law is easier to apply in hindsight than in

the speed of the voltage transition. This switching speed (*dv/dt*) is directly proportional to the current output (*I*) of the driving transistor and inversely proportional to the capacitance (*C*)of the driven transistor gates and the interconnection wiring, as given by the formula $dv/dt = I/C$.

The output current of an MOS driver transistor sets the switching speed, and depends on circuit layout and process features. The voltage on the gate and the transistor's size are the most important circuit design features, determining an output transistor's drive strength and the ultimate circuit's speed. With silicon (rather than metal) gate the transistor's size was reduced because the source and drain features were formed by the self-aligned silicon gate [31].

Although today's circuits utilize two transistor types for optimum drive for both rising and falling signals, that is, transition from logical 0 to 1 and from 1 to 0, earlier circuits weren't good at both "pushing and pulling." Accordingly, the circuits were operated in dynamic mode with a precharge and conditional discharge circuit. First a circuit was precharged by an on-chip amplifier, and then, according to the logic state, was conditionally discharged.

The PMOS transistors of 1970 required 14 V to operate; the circuits were operated dynamically in either a two-phase or a four-phase mode. The historic improvements in MOS process technology have resulted in

- lowering the voltage swing needed to switch a logic signal
- reducing the transistor size to improve a driver's output current
- reducing the transistor size to lower the gate's capacitance.

RAM Circuits

In 1970 an IC could contain about 256 bits of static RAM, the limit being imposed by both power dissipation and chip size. Dynamic RAM (DRAM) chips became practical in the early 1970s and had much lower power requirements, but needed to be refreshed periodically. The three-transistor dynamic memory cell was considerably smaller than a six-transistor static memory cell.

In addition, the cell connection signals were a major area constraint. Memory circuit design was often described in terms of the number of connecting bus lines, for example, a three-line or six-line organization. Larger DRAM memory chips were made possible by using a single transistor per cell and just two lines—gate selection and bidirectional data bus.

Early microcomputer chips used small, integrated dynamic RAM arrays for the CPU's registers and a program counter stack. For example, Intel's 4004 had a 64-b DRAM for its 16 four-bit registers, and the 8008 had a push-down stack of 14 × 8 within a DRAM array. The decisions to be made in designing ICs—then and now—are many. Table 1 highlights some of them.

TABLE 1. LSI CHIP ISSUES.

LSI Chip Issues:

■ How many pins on the IC package?	■ What are the operating voltages?	■ What chip speed and power goals?
■ What is IC package's power dissipation constraint?	■ What are the I/O interface voltages and signal timings?	■ What is the on-chip routing/bus strategy?
■ What is the projected die size and aspect ratio?	■ What are the technology constraints?	■ How will the chip be tested?

You will see that our early experience with small computers was a factor in creating the first microcomputer.

foresight. This despite the fact that we announced the MCS-4 chip at Intel in 1971, roughly five years "ahead of schedule"; the CPU had about 2k transistors [7], [8].

Generally, if you can double a chip's density next year, you have two choices:

- halve the cost of a chip you're currently making
- make a new chip with twice as much "stuff" on it.

By way of analogy, suppose you were a bicycle manufacturer that could reduce bicycle prices every year by 50%: $80, $40, $20, . . . $1.25. After a while everyone would own several and the market would become saturated; this wouldn't be a good business. Sometimes lowering prices increases market consumption and sometimes not. Accordingly, although many believe Moore's law is about semiconductor technology, thoughtful analysis reveals two fundamental business questions: Will decreasing chip prices dramatically increase chip sales? Will the expected profits justify improving semiconductor processes? In other words, Moore's doubling occurs only if it makes good business sense, that is, can you use more transistors sensibly?

If Moore's law is taken into account, there are more choices to consider, as Table 2 shows. First, for an existing product, one needs to know the results of lowering the price. Often new applications and new markets are needed to propel the sales volume of an existing IC chip.

For new chip designs, it is a challenge to determine what kind of chip to make. Table 1 illustrates the options for new chips for both existing and new markets. Deciding on how to use more transistors requires a good understanding of how new chips would be used—their *application*. The role of both the applications engineer and the product marketing engineer become prominent in new chip specification—how big, how fast, what features, how many will sell, what price?

Let me continue now with my personal story after I joined Fairchild.

Transistor Data (1964)

One of my application engineering projects at Fairchild was to write a program to calculate the *Y*-parameters of individual, or *discrete*, transistors. At that time each three-legged transistor had a serial number, and we recorded and calculated parameters for each part. When users paid US$150 per transistor, arguably they paid more for the data than for the transistor itself. Fairchild applications engineers were selling their service bundled with the devices.

When transistor prices dropped by a thousand times (to US15¢ each), the price of a five-transistor radio dropped to about US$2. At some point the solid-state devices weren't a factor in a radio's price; the costs of the case, power supply, battery, coils, capacitors, and resistors outweighed transistor costs. Soon everyone owned a couple of these radios, and the market for radios saturated. Chips containing more transistors *and their wiring*—that is, integrated circuits—would be

FIGURE 1: Stan Mazor and an IBM 1620 in 1963.

TABLE 2. LSI CHIP MARKETING ISSUES.

Lower the price of existing chips.
Find new uses for existing chips.
Build a similar chip with improved features.
Build a noncompatible chip with improved features.
Develop a new chip design for an existing chip market.
Develop a new chip to replace another technology—mechanical, magnetic, analog, other.
Develop a new chip for a totally new market.

the key to fueling Moore's law and lead to lower digital system costs.

Novel Computer Architecture (1966)

At Fairchild I programmed six different computers, and in 1966 I transferred to Gordon Moore's R&D labs in Palo Alto, joining Rex Rice's high-level language computer design project. Following Moore's predictions for "cheap" logic, we built a radical computer called Symbol (Figure 2) that had 100 times more logic within the CPU. It used more than 20,000 of Fairchild's complementary transistor logic (CTL) chips. However the idea of "maximizing" CPU logic, while consistent with Moore's law, was flawed and the project wasn't a success [9], [10].

However, my experience in designing the serial decimal floating point arithmetic logic unit (ALU) and the string-processing unit would later help me in understanding the Busicom calculator's arithmetic, with Intel's first microcomputer [1].

Intel Is Founded (1968)

Robert Noyce and Gordon Moore quit Fairchild, where they were general manager and director of R&D, respectively, and started Intel to capitalize on the emerging semiconductor memory market and fulfill Moore's promise of growing chip density [11]. My Fairchild officemate, Jim Angel, suggested they hire a brilliant Stanford research associate, M.E. (Ted) Hoff, as director of applications research. I joined Hoff at Intel in 1969 as an applications engineer. (I recall first meeting Hoff in 1963 while he was demonstrating his experiments in speech recognition on an IBM 1620 at Stanford University. At that time I was also programming an IBM 1620 on more mundane applications.)

You will see later that our early experience with small computers was a factor in creating the first microcomputer. But let's consider the first kinds of memory chips that were enabled by Moore's law.

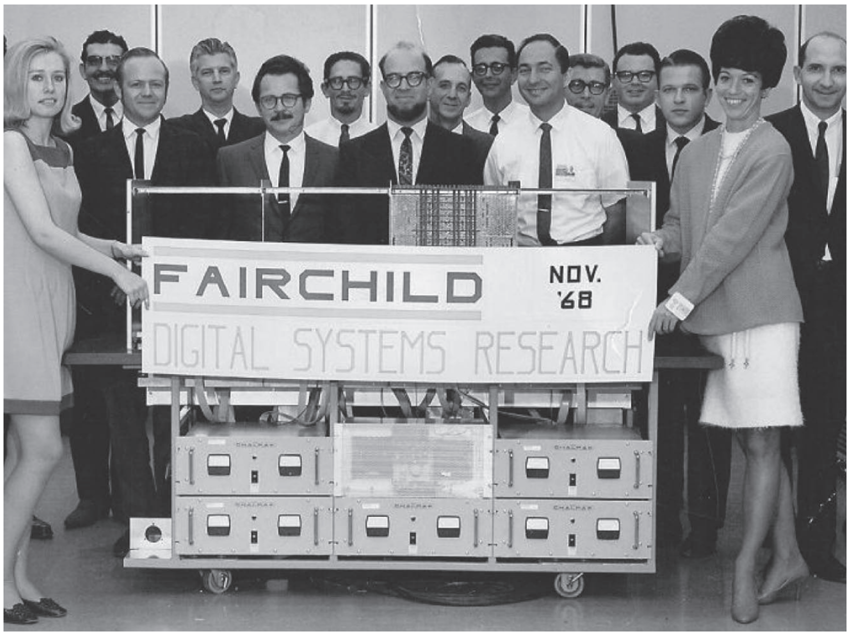


FIGURE 2: Experimental Fairchild symbol computer, circa 1968, shown with its developers in the rear and, in the foreground, two attendants at Fairchild's introduction.

Shift Register ICs (1969)

If you look at history you'll find that shift registers were one of the first large-scale integration (LSI) chips available, and from several companies—including: General Instrument, Electronic Arrays, MOS Technology, AMI, and Intel. Although dynamic random-access memories (DRAMs) are now common, the shift register was a precursor memory chip and had several advantages. A shift register chip has few leads and can be encased in a small eight-pin package (the TO-5 can).

Keep in mind that, although chip density had been doubling, the number of input and output pins on a package was growing slower, so I/O pin count was a real limitation to a chip designer. Normally chip wiring is a major problem for designers—one that eats up valuable chip real estate. But not in a shift register, for three reasons:

- Serial memories have no address pins, just the data-in, data-out, and clock and power pins—regardless of the number of bits inside the chip, as shown in Figure 3.
- Shift register chips are simpler to design and debug because they have no address decoder in the chip, and most of the circuit layout just repeats the memory cells, thousands (or millions) of times.
- On-chip wiring is minimal, since each cell communicates with just its left and right neighbors.

Finding new uses for shift register chips was a challenge for the Applications Engineering department. We built a few interesting systems such as a moving signboard using shift register chips. Realizing that some of the early computers used serial disk memories for the main program memory, we proposed using shift registers for main memory. However, a principal use of shift registers turned out to be video screen refresh circuits, since video is a bit-serial application [12], [23].

Minicomputer Market (1965–1969)

DEC's 12-b PDP-8 and Data General's 16-b Nova popularized the general-purpose minicomputer. A few

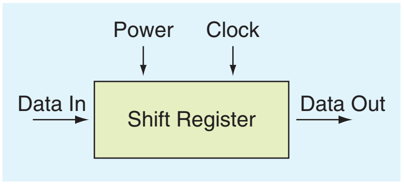


FIGURE 3: Shift register block diagram.

In just 15 years, playing games had gone from being a frivolous use of computers to a big business.

years later, many other vendors offered comparable minicomputers for under US$10,000. There were a variety of minicomputers featuring 12–18-b words for both data and instructions, and they came with 4K words of core memory. These minicomputer CPUs were built with bipolar transistor-transistor logic (TTL) gates and a few medium-scale integration (MSI) ICs. These MSI parts were typically 4-b wide—multiplexers, adders, shifters, and so forth.

While the new metal-oxide semiconductor (MOS) circuits offered ten times the density of TTL circuits, their slow speed made them unacceptable for CPU logic circuits. And even though MOS chips cost less than TTL, when you added the cost of main memory and peripherals, the cost savings would not be significant.

Moving from diode-transistor logic (DTL) and resistor-transistor logic (RTL) logic that was popular in the early 1960s, many manufacturers produced TTL logic in 14-pin dual in-line packages—Texas Instruments, National Semiconductor, Sylvania, and others. These sold for about US25¢ each on average, and the cost of a three-input NAND gate was less than a dime. The plentiful availability and low cost of these ICs made possible digital systems in general and minicomputers in particular. Moore's law was at work; the lower costs of ICs opened up a vast minicomputer market [13], [14].

Universal Arithmetic Element (1970)

Given the popularity of 4-b-wide MSI parts, one of Intel's first products was a 16 × 4 high-speed bipolar memory chip. It could be used to provide data registers within a CPU. Intel's bipolar read-only memory (ROM) could hold the microcode for the CPU's logic.

Hoff also began developing a 4-b universal arithmetic element (UAE) at Intel. My job was to design and demonstrate a CPU using this UAE with Intel ROMs and RAMs. We decided to emulate the 12-b DEC PDP-8 minicomputer, which was a popular standard. To make it interesting, my coworkers and I fit the entire CPU on a small 36-chip board (Figure 4), and we used ROM microcode to define an instruction set similar to the PDP-8's. We presented the results of this experiment at the Northeast Regional Electronics Meeting (NEREM) in 1970 [15]. Intel considered the UAE experiment as a demonstration of "miniaturization" and did not pursue the UAE as a product.

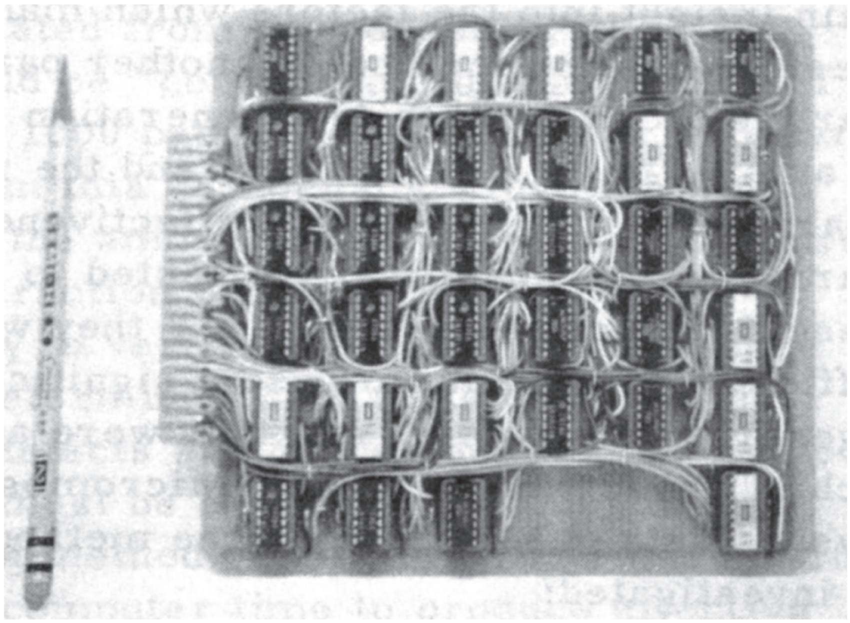


FIGURE 4: Intel's experimental 12-b CPU using universal arithmetic element chips.

DRAM Versus Core Memory (1972)

A general-purpose computer can't do much until a program is loaded into its memory. The magnetic cores used in memories in 1972 could hold a program with power absent. When Intel promoted semiconductor DRAM as a replacement for core memory, an oft-heard complaint was that DRAM would lose data if the power was off, and this was true. However, in reality not many computers relied on this feature, and it was common practice to load/reload a program just before executing it. Arguments from Intel applications engineers familiar with actual customer use overcame this criticism of DRAMs; we did succeed in getting computer designers to switch from magnetic core to DRAM chips. It's been said that Intel created the RAM business in 1972, and Intel was indeed a major DRAM chip supplier, but that success was a result of a combination of chip engineering and applications engineering support [11].

DRAM Improvements (1972)

Historically, doubling RAM chip bits according to Moore's doubling of the transistor count was a good fit—larger RAMs were both natural and needed. Furthermore, only one additional address input lead was needed on the chip's package—and, as I noted earlier, I/O pins are a severe limitation in chip design. With minor redesigns, customer memory boards were upgraded to use newer and larger DRAM chips, and customers readily accepted these improved chips [16].

When Intel successfully produced the first DRAM chips, commercial viability was slow to come.

Content-Addressable Memory—A Failure (1974)

After an address is input into a RAM, the contents are returned. In a content-addressable memory (CAM) it's just the opposite. Data for matching are entered, and if a match is found within the CAM, the location is output. CAMs are much faster than searching RAMs, but they require additional circuitry that increases the physical size of the CAM chip, which in turn increases manufacturing cost.

We believed CAMs would be useful in CPU memory page tables (virtual memory). Applications engineering promoted this product, but regrettably no large-scale market appeared after the chip materialized—an applications engineering failure.

Today, CAMs are only used in specialized applications where adequate searching speed cannot be achieved with a less costly method.

A Memory Market Pitfall—And a Solution (1969)

When Intel successfully produced the first DRAM chips, commercial viability was slow to come. Although customers would buy samples, their lead time from engineering to manufacturing meant that volume production orders wouldn't be realized for several years. Meanwhile Intel's own production line would be idle. Intel needed a way to utilize its factory with a shorter lead-time product [17].

The Busicom desktop calculator provided a way to keep idle production lines busy. While minicomputer unit sales were only in the low thousands at the time, desktop calculators were selling by the hundreds of thousands. Using a handful of MOS LSI chips, they sold for less than US$1,000. Again, low prices led to large sales volumes.

Because a desk calculator responds to keystrokes, not stored programs as in minicomputers, it was a fine match for the speed, density, and cost of MOS LSI. Japan's Busicom promised the substantial sales volume that Intel needed if we could design and build custom MOS LSI chips for their new desktop calculator [18].

Hoff was evaluating Busicom's design when I joined him at Intel in 1969. Busicom's Masatoshi Shima had designed the overall logic for his calculator's custom chip set. His design called for a processor that operated on multidigit decimal numbers, a ROM for coding floating-point operations, and separate control chips for the keyboard, display, and printer. Hoff proposed a simpler approach substituting programming for hardware, and I assisted in this design. Table 3 lists some of the key design decisions on this project [19].

Shima and I shared an office; I was the principal liaison on the project. Because he had done quite a bit of work on his design, he was skeptical of Intel's alternative proposal. I needed to demonstrate how we could achieve various calculator features by programming rather than in hardware. Moreover, all of his flowcharts for floating-point arithmetic assumed multidigit fixed-point numbers, but our CPU operated on only a single digit. I needed to make the CPU look more like Shima's original and show him

TABLE 3. KEY DESIGN DECISIONS FOR THE BUSICOM CHIP SET.

Busicom/Masatoshi Shima:	
	Family of systems using the same custom components
	Serial decimal floating point arithmetic via a ROM program
Intel/Ted Hoff:	
	4-b architecture
	Separate program ROM and data RAM chips
	Time multiplex 4-b bus; 16-pin IC packages
	Dynamic RAM for CPU registers and PC stack
	4-b I/O ports (RAM and ROM chips) for interfacing
	ROM program: keyboard, printer, lights
Intel/Stan Mazor:	
	FIN/JIN instructions to Fetch/Indirect jump within ROM
	Pseudo-code interpreter to reduce ROM code size
	4004 assembler and ROM code bit mapper
	Code snippets for calculator functions
Intel/Federico Faggin:	
	Custom chip methodology, circuits, layout
	Bootstrap amplifier circuit for silicon gate process
	Checking and debugging custom IC chips

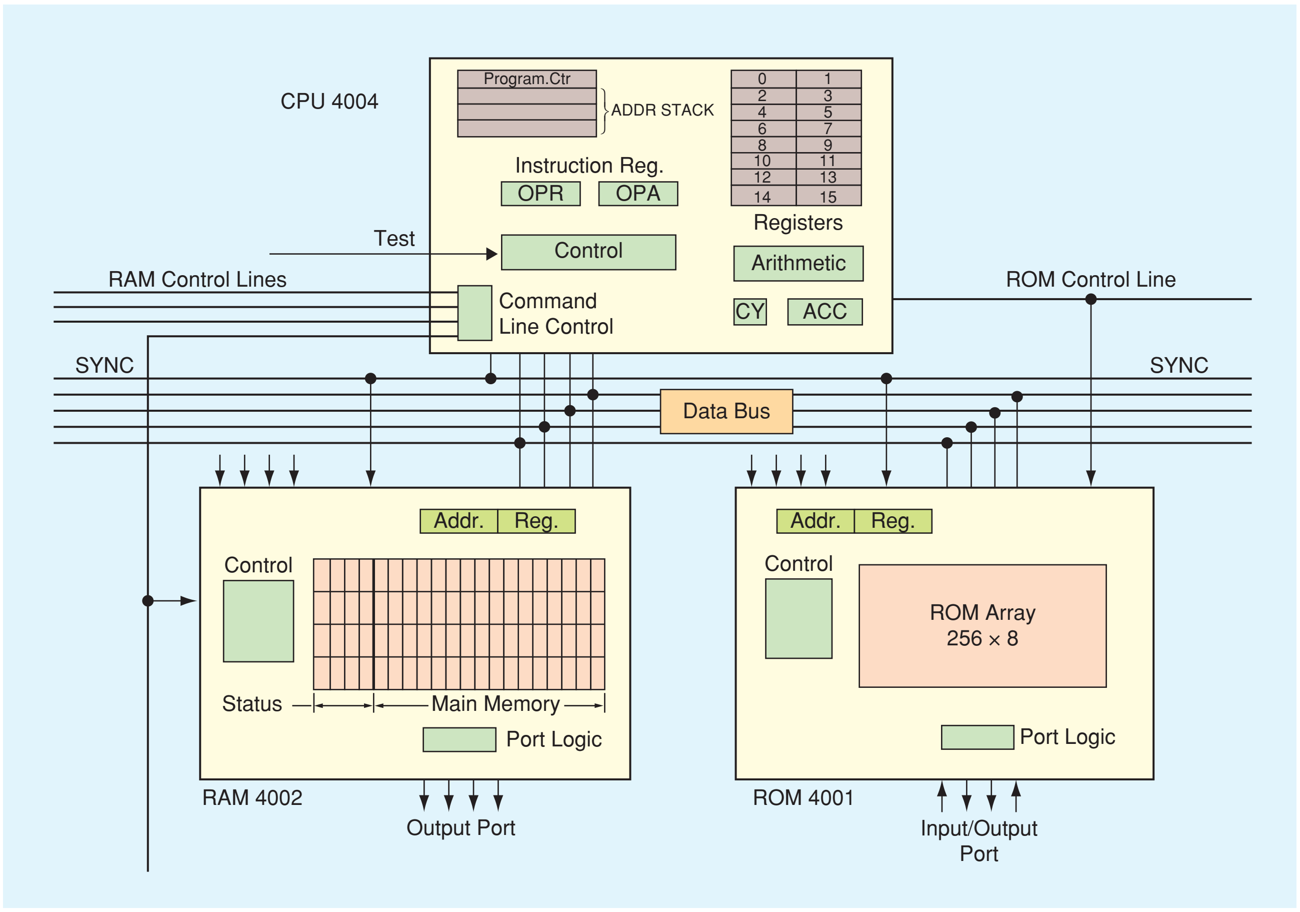


FIGURE 5: Block Diagram of the Busicom chip set (MCS-4).

how the features he needed could be provided.

Using my college programming experience with virtual machines, I made Intel's system look more like Shima's original. An interpreter program, occupying less than 20 bytes, was the solution.

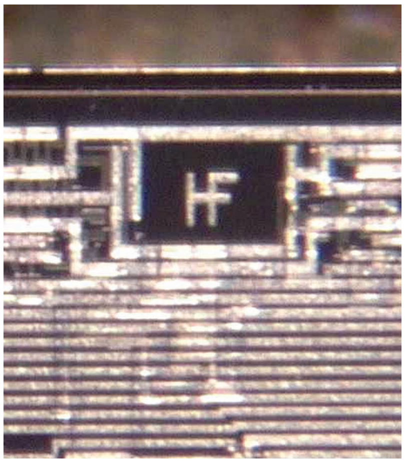


FIGURE 6: An 8008 die with designer Hal Feeney's initials.

This also reduced the amount of ROM needed by replacing 2-byte instructions with 1-byte pseudo operations. To interpret Shima's "pseudo-instructions," we added two CPU instructions—the ability to fetch data from ROM (fetch indirect) and to jump to a subroutine (jump indirect). I wrote programming snippets, to operate on a field of digits, and also wrote program pieces for scanning the keyboard, displaying data in lights, and running the printer.

In the end, Shima did all the calculator design and coding of four ROM chips for the Busicom calculator. The interpreter directed the program to the correct subroutines. Hoff's architecture was proven and provided a general-purpose solution (Figure 5). Federico Faggin did all the chip design, circuit design, and layout, resulting in a new microcomputer chip set that later became a standard product, MCS-4 [20]–[22].

Busicom produced several different calculators using this family of parts. However, in just a few years the growing density of LSI made the products obsolete. Busicom was ultimately beaten by competition that used more dense and less general chips.

MCS-8 (1972)

In 1969, Intel built custom shift register chips for Control Terminal Corporation's (CTC's) Datapoint display terminals. (The company later changed its name to Datapoint Corporation.) CTC asked me for a "stack chip" for use in their new 8-b CPU, unaware of our Busicom microcomputer project. Although a single-chip CPU like the MCS-4 was "conceivable," few knew *how* to do it practically. There were limits on the size of chip that could be built and the amount of circuitry we could put on a chip. Hoff and I discussed the possibility of building the Datapoint CPU as a single chip. At first the amount of logic circuitry required for an 8-b CPU seemed prohibitive, with twice as many transistors as a 4-b CPU. But in a CPU, the logic for instruction decoding and execution is *not* dependent on the data word size, so instruction decoding logic in an 8-b CPU doesn't require much more circuitry than that for a 4-b CPU—and we knew a 4-b CPU was feasible. See the photo of the 8008 chip in Figure 6 and designer Hal Feeney's signage. Subsequently, we proposed the first 8-b CPU chip, announced in 1972 as the 8008. This led to the 8080 and the 8086—the CPUs that launched the PC business [23]–[30].

In Retrospect

Moore's law of doubling density unquestionably affects everyone's life, with ubiquitous cell phones, personal computers, ATMs, and

FROM PROGRAMMING TO PHOTOLITHOGRAPHY: A PROFESSIONAL ODYSSEY

As Ted Hoff and I were computer users as well as computer designers, we organized and managed digital computers at Intel. One of our major accomplishments was providing computer-aided design (CAD) tools to assist chip designers—including logic simulation and circuit simulation tools (transient analysis). I wrote and maintained the production version of our home-grown (pre-Spice) circuit analysis program, using Hoff's circuit simulation strategy and Dov Frohman's transistor model. I constantly revised my program to handle larger circuits and the ever-changing technology on our DEC PDP-10 large-scale, time-shared computers. This was a nice way for me to learn some semiconductor physics and observe circuit phenomena. At Fairchild, I had done extensive logic simulation on my floating-point arithmetic unit, and there is a fine difference between functional logic simulation and lower-level circuit simulation.

In 1974 I transferred to Belgium to become Intel's first field applications engineer in Europe, and to develop new markets in new places. I found many exciting applications for our microcomputers in a variety of companies and industries. When I returned to the United States in 1976 I worked extensively on microcomputer programming, writing, lecturing, and teaching on the subject. In fact Intel trained tens of thousands of engineers using the Intel Development System (Figure 7), in which I participated.

A Higher Level of Abstraction

In 1983, I left Intel to join a start-up, Silicon Compilers, realizing that the real potential of very large scale integration (VLSI) could be reached only by designing at a higher level of abstraction [32]. After five years I abandoned that work to join CAD start-up, Synopsys, working in logic synthesis. I had found that, although the compilation ideas weren't effective, logic synthesis from a hardware description language *was* practical. At Synopsys I managed a capable team of application engineers and trainers for more than five years. I continued my writing and teaching about these new methodologies and published a popular book on VHDL, the design language for field-programmable gate arrays and application-specific integrated circuits [33].

Staying in the CAD field, I worked at Cadabra, where we automatically generated standard cell layouts from transistor netlists. Again, the ever-changing technology meant reducing the time delay in developing an IC layout. Standard cells were a good meeting place for the logic designer and circuit and layout designer.

Later I joined Numerical Technologies to help overcome the 250-nm limit on optical photomask resolution [34]. Our solution involved phase-shift masking, which made 20-nm geometries feasible—and helped continue Moore's law.

Smaller Focus

It's amusing that, although I started as a computer programmer, I moved into logic design, then circuit design, layout, and finally photolithography. While the focus of my work continued to grow "smaller," most of my concerns were with automation tools and techniques to give designers more power and flexibility in doing their designs—for logic, circuits, layout, and photomasks.

Since I "retired," I've written two books: one on using high-level design methodologies in home construction *(Design an Expandable House)* and the other on using statistical techniques in the stock market *(Stock Market Gambling)* [35], [36]. Each year I visit two colleges and share some of my engineering experiences, and I'm active in writing short history articles about the early microcomputer days [37]. Every year I help to organize the invitational Asilomar Microcomputer Workshop. I invite interested readers to contact me at stanmazor@sbcglobal.net with comments or questions.

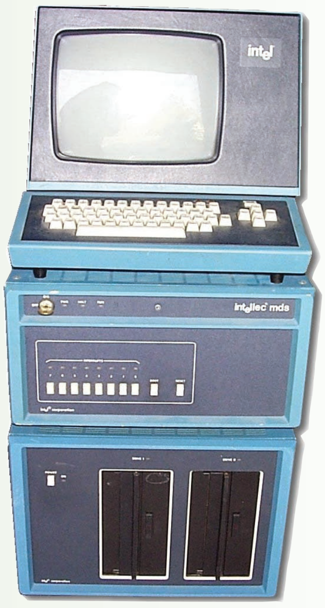


FIGURE 7: Intel development system.

Since I "retired," I've written two books: one on using high-level design methodologies in home construction and the other on using statistical techniques in the stock market.

invisible, embedded computers providing smarter machines everywhere. That Intel was a memory company certainly influenced our ability to make CPUs, and microcomputers helped Intel sell main memory chips (DRAM, EPROM, ROM).

This success was made possible by intertwined efforts. Developing new standard chips requires close cooperation among applications and marketing engineers who interpret users' needs, as well as clever process and chip designers who implement new technology. The development of businesses at Intel relied on applications engineers to define new products as well as skilled design engineers. Hoff and Mazor's early experiences with computers and programming, and Shima's and Faggin's design background were key to the creation of the microcomputer at Intel during the early 1970s. I was very lucky to be working with these talented coworkers and to participate in a great team. Although Moore's curve is aggressive, the first microcomputer was "slightly ahead of the curve."

Incidentally, my career didn't end when I left Intel—far from it. I continued to work on microcomputers, albeit in diverse ways, and I continued to see Moore's law in operation (see "From Programming to Photolithography: A Professional Odyssey").

References

[1] S. Mazor, "The history of the microcomputer," in *Readings in Computer Architecture*, M. Hill, N. Jouppi, and G. Sohi, Eds. San Francisco, CA: Morgan Kaufman, 2000, p 60. Reprinted from *Proc. IEEE*, vol. 83, pp. 1601–1608, Dec. 1995.

[2] "*Halycon days: Interviews with classic computer and video game programmers.*" Video book available at www.dadgum.com/halcyon/.

[3] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, pp. 114–117, Apr. 19, 1965.

[4] "The technical impact of Moore's law," *IEEE Solid-States Circuits Society Newsletter*, vol. 20, Sept. 2006.

[5] F. G. Heath, "Large scale integration in electronics," *Sci. Amer.*, p. 22, Feb. 1970.

[6] P. E. Haggerty, "Integrated electronics—A perspective," *Proc. IEEE*, pp. 1400–1405, Dec. 1964.

[7] "*MCS-4 micro computer set*," Data Sheet 7144, Intel Corp., 1971.

[8] Intel advertisement, *Electronic News*, Nov. 1971.

[9] L. C. Hobbs, "Effects of large arrays on machine organization and hardware/software tradeoffs," in *Proc. 1966 Fall Joint Computer Conf.*, vol. 29, p. 89.

[10] S. Mazor, "Fairchild Symbol Computer," *IEEE Ann. History Comput.*, vol. 30, pp. 92–95, Jan.–Mar. 2008.

[11] G. Bylinsky, "Little chips invade the memory market," *Fortune*, pp. 100–104, Apr. 1971.

[12] M. Hoff and S. Mazor, "Operation and application of shift registers," *Computer Design*, pp. 57–62, Feb. 1971.

[13] D. C. Hitt et al., "The mini-computer—A new approach to computer design," in *Proc. IEEE 1968 Fall Joint Comp. Conf*, pp. 655–662.

[14] R. Hooper, "The minicomputer, a programming challenge," in *Proc. 1968 Fall Joint Comp. Conf.*, pp. 649–654.

[15] M. Hoff and S. Mazor, "Standard LSI for a micro programmed processor," in *IEEE NEREM '70 Record*, Nov. 1970, pp. 92–93.

[16] J. Karp, A. Regitz, and S. Chou, "A 4096-bit dynamic MOS RAM," in *Proc. Int. Solid-State Circ. Conf.*, Feb. 1972, pp. 10–11.

[17] R. Noyce and M. Hoff, "A history of microprocessor development at Intel," *IEEE Micro*, vol. 1, no.1, pp. 8–21, 1981.

[18] M. Shima, *The Birth of the Microcomputer: My Recollections.* Tokyo: Iwanami Shoten, 1987 (in Japanese).

[19] M. E. Hoff, S. Mazor, and F. Faggin, "*Memory system for a multi-chip digital computer*," U.S. Patent 3,821,715, Intel Corp., June 1974.

[20] F. Faggin et al., "The MCS-4—An LSI micro computer system," in *Proc. IEEE Region 6 Conf.*, 1972, pp. 8–11.

[21] H. Smith, "Impact of LSI on microcomputer and calculator chips," in *IEEE NEREM '72 Rec.*, 1972.

[22] S. Mazor, "Micro to mainframe," *IEEE Ann. History Computing*, vol. 27, pp. 82–84, April–June 2005.

[23] S. Mazor, "8-bits of Irony," *IEEE Ann. History Computing*, vol. 28, pp. 73–76, April–June 2006.

[24] *Intel MCS-8 User Manual*, 1972.

[25] V. Pzoor, "Letters," *Fortune*, p. 94, Jan. 1976.

[26] G. Boone, "*Computing system CPU*," U.S. Patent 3,757,306, Texas Instruments, Sept. 1973.

[27] G. Bylinsky, "Here comes the second computer revolution," *Fortune*, Nov. 1975.

[28] S. Mazor, "Intel 8080 CPU chip development," *IEEE Ann. History Computing*, vol. 29, pp. 70–73, April–June 2007.

[29] F. Faggin, M. Shima, and S. Mazor, "Single chip CPU," U.S. Patent 4,010,499, Intel Corp., 1977.

[30] S. Morse, B. Ravenel, S. Mazor and W. Pohlman, "Intel microprocessors 8008 to 8086," *Computer*, pp. 42–60, Oct. 1980.

[31] L. Vasdasz, A. Grove, G. Moore, and T. Rowe, "Silicon gate technology," *IEEE Spectrum*, pp. 27–35, Oct. 1969.

[32] S. Mazor and L. Jack, "The validation of silicon compiler technology on a VHSIC process," in *GOMACTech Digest*, Nov. 1986, pp. 437–441.

[33] S. Mazor and P. Langstraat, A Guide to VHDL, 2nd ed. Norwell, MA: Kluwer, 1993.

[34] L. Karklin, S. Mazor, et al., "Subwavelength lithography: An impact of photomask errors on circuit performance," *Proc. SPIE*, pp. 259–267, July 2002.

[35] S. Mazor, Design an Expandable House, 2nd ed. Morrisville, NC: Unlimited Publishing, Lulu, 2003.

[36] S. Mazor, *Stock Market Gambling: Turning on a Dime.* Morrisville, NC: Unlimited Publishing, Lulu, 2007.

[37] S. Mazor, "Programming and/or logic design," in *Proc. IEEE Computer Group Conf.*, 1968, pp. 69–71.

[38] B.O. Evans, "System/360: A retrospective view," *Ann. History Computing*, vol. 8, no 2, pp. 155–179, 1986.

[39] M. Wilkes, "The genesis of microprogramming," *Ann. History Computing*, vol. 8, no. 2, pp. 115–126, 1986.

About the Author

Stanley Mazor (stanmazor@sbcglobal.net) worked on early microprocessor chips at Intel and shares patents on the 4004 and 8080 microcomputer chips. Previously he worked on the design of Symbol, a high-level language computer at Fairchild R&D (1964). He has worked in several start-up companies including: Intel, Synopsys, Silicon Compilers, Numerical Technologies, Cadabra, and BEA Systems. He studied mathematics at San Francisco State College. He has published 55 articles relating to large-scale integration chips and three books including *A Guide to VHDL* (Kluwer, 1993). For his work on Intel's microcomputers he was awarded the Kyoto Prize, the Ron Brown American Innovator Award, and the Semiconductor Industry Association Robert Noyce Award and was inducted into the Inventors' Hall of Fame. His hobby is architecture, and he recently published *Design an Expandable House* as well as *Stock Market Gambling*.