The History of the Microcomputer—Invention and Evolution

STANLEY MAZOR, SENIOR MEMBER, IEEE

Invited Paper

Intel's founder, Robert Noyce, chartered Ted Hoff's Applications Research Department in 1969 to find new applications for silicon technology—the microcomputer was the result. Hoff thought it would be neat to use MOS LSI technology to produce a computer. Because of the ever growing density of large scale integrated (LSI) circuits a "computer on a chip" was inevitable. But in 1970 we could only get about 2000 transistors on a chip, and a conventional CPU would need about 10 times that number. We developed two "microcomputers" 10 years ahead of "schedule," by scaling down the requirements and using a few other "tricks" described in this paper.

I. INTRODUCTION

Intel's first microcomputer ad appeared in November 1971:

"Announcing a new era in integrated electronics."

Intel delivered two different microcomputers five months apart: the MCS-4, emphasizing low cost, in November 1971, and the MCS-8, for versatility in April 1972. "The MCS-4 and MCS-8 CPU chip sell in quantity for less than $100 each, and are powerful alternatives to random logic" [1]. These two Micro Computer Systems (MCS) were aimed at two very different markets. One would eventually lead to the under $1 controller, the other would be the engine for a versatile personal computer (PC). By analogy it was like creating the "motorbike" and the "station wagon" at the same time. The advertised prophecy of "a new era" became fulfilled over the subsequent 20 year period.

A. Automobile Analogy

Our challenge was how to scale down a general purpose computer to fit on to a chip. Imagine that the only passenger vehicle in existence is an eight-passenger van costing $50 000. At first it would be difficult to imagine a $1000 version of this vehicle. The specifications would need to be drastically reduced to meet the price goal. Some ideas to consider:

1) reducing capacity by 75%
2) reducing speed by 90%
3) reducing range by 75%.

The golf cart might be the result. However, if golf carts are unknown at the time, it is not easy to envision how to scale down a van.

What features of a computer can be scaled down? That depends on what it will be used for. Fortunately for us, our first customer's application was for a desktop calculator; we scaled down the computer's speed and memory size to meet the needs of this particular application. As computers go, the microcomputer was not very capable; some would say that we set the computer industry back 10 years. We thought we were moving the LSI world ahead by 10 years [2]. I will share some of my recollections of the early days of Intel microprocessors.

II. INTEL MCS-4 4-B CHIP SET

Although Intel began as a memory chip company [3], in 1969 we took on a project for Busicom of Japan to design eight custom LSI chips for a desktop calculator. Each custom chip had a specialized function—keyboard, printer, display, serial arithmetic, control, etc. With only two designers, Intel didn't have the manpower to do that many custom chips. We needed to solve their problem with fewer chip designs. Ted Hoff chose a programmed computer solution using only one complex logic chip (CPU) and two memory chips; memory chips are repetitive and easier to design. Intel was a memory chip company, so we found a way to solve our problem using memory chips!

In 1970 Intel designers implemented a 4-b computer on three LSI chips (CPU, ROM, RAM) housed in 16-pin packages [4]. Reducing the data word to 4-b (for a BCD digit) was a compromise between 1-b serial calculator chips and conventional 16-b computers. The scaled down 4-b word size made the CPU chip size practical ($\sim$ 2200 transistors). We used the 16-pin package, because it was the *only one* available in our company. This limited pin count forced us to time multiplex a 4-b bus. This small bus simplified the

Manuscript received January 12, 1995; revised August 17, 1995.
The author is with C-ATS Software, Palo Alto, CA 94303 USA.
IEEE Log Number 9415184.

printed circuit board (PCB), as it used fewer connections. However, the multiplexing logic increased chip area of the specialized ROM/RAM memory chips, which then had to have built-in address registers. Increasing the transistor count to save chip connections was a novel idea. In school we learned to minimize logic, not interconnections! Later, LSI "philosophers" would preach "logic is free" [5].

1) MCS-4 Features [6], [7]:

256 × 8	Read Only Memory (2 kb ROM) with 4-b I/O port
80 × 4	Random Access Memory (320 b RAM) with 4-b output port
4-b	CPU chip with:
	16 × 4-b index registers
	45 1 and 2 byte instructions
	4-level Subroutine Address Stack
	12-b Program Counter (4 k addresses).

A. ROM Chip (4001)

Conventional calculators utilized specialized custom chips for keyboard, display, and printer control. With the MCS-4 all control logic is done in firmware, program stored in ROM [8]. A single ROM chip design is customized (with a mask during chip manufacturing) for a customer's particular program. The CPU's 12-b Program Counter addresses up to 16 ROM chips. Simple applications use only one ROM chip; the desktop calculator used four. The same chip mask also configured each ROM port bit as an input or output.

Additionally, the ROM chip had an integrated address register, an output data register, multiplexors, and control and timing logic. The specialized RAM chip had similar resources.

B. RAM Chip (4002)

Calculators need to hold several 16-digit decimal floating point numbers. We organized the RAM accordingly, and ended up with a 20-digit word (80 b):

16 digits for the fraction
2 digits for the exponent
2 digits for signs and control
20 digits × 4 b/digit

The RAM chip stored four 80-b numbers and additionally the chip had an output port. The use of three-transistor dynamic memory cells made the RAM chip feasible [9]. A built-in refresh counter was used to maintain data integrity. Refresh took place during instruction fetch cycles, when the RAM data was not being accessed. Dynamic RAM memory cells were also used inside the CPU for the 64-b index register array and 48-b Program counter/stack array. Intel expertise in dynamic memory was an enabling factor for the MCS-4!

C. Input/Output Ports

To conserve chip count and to utilize existing power/clock pins, the 16-pin ROM and RAM chips also had integrated 4-b ports for direct connection of I/O devices. To activate an output, a program selected a particular RAM/ROM chip (using an index register) and sent 4-b of Accumulator data from the CPU to the selected output port. In the desk calculator application, the display, keyboard, and printer were connected to these ports. Keyboard scanning, decoding, and debouncing [10] were all done under program control of the I/O ports; all printer and display refresh was done in firmware [11]. A small shift register (4003) was used for output port expansion. External transistors and diodes were used for amplification and isolation.

D. Microprocesor—CPU Chip (4004)

In the calculator application, each user key stroke caused thousands of CPU instructions to be executed from ROM. We wrote many subroutines which operated on 16-digit numbers stored in RAM. As an example, a 10-byte loop for digit serial addition took about 80 μs/digit (similar speed as IBM 1620 computer sold in 1960 for $100 000). In this add routine a CPU index register would address each of the 16 digits stored in the RAM memory. The program would bring in one digit at a time into the CPU's accumulator register to do arithmetic. A Decrement and Jump instruction was used to index to the next RAM location.

One major difference compared to most computers, was the MCS-4's separate program and data memories. Conventional computers ran programs from RAM (core) memory. However, our application firmware needed to be permanently stored in ROM. A major change was needed for subroutine linkage. Normally, as part of a minicomputer [12] subroutine call instruction execution (PDP-8, HP 2114) the calling program's return address would be saved at the top of the subroutine in RAM. Since MCS-4 routines were in ROM (can't write into it) we could not use this method. Instead, we used a push down stack inside the CPU for saving up to three return addresses. This was not a new idea. Stacks had been used in Burrough's computers and the IBM 1620, which Ted Hoff and I had programmed—we used our experience with large scale computers. Ultimately this limited depth of four levels (which was all we could squeeze on to this small chip) was frustrating for programmers and succeeding generations went to eight or more levels (8008, 4040, 8048). Today's computers have stacks of many megabytes; but their usage is very similar to their use in the 4004.

E. Distributed Logic Architecture

The time division multiplexing of the 4-b bus, the on-chip dynamic RAM memories, and the CPU's address stack are the highlights of the MCS-4 architecture. However, there is another interesting feature—distributed decoding of instructions. The ROM/RAM chips watched the bus, and locally decoded port instructions, as they were sent from

the ROM. This eliminated the need for the CPU to have separate signal lines to the I/O ports, and also saved CPU logic. This is not a feature used in conventional computers.

F. MCS-4 Applications

The smallest system would contain two chips—a CPU and a ROM. A typical calculator had 4 ROM's and a RAM chip—with five I/O ports, (20) wires for connecting peripheral devices. A fully loaded system could have 16 ROM and 16 RAM chips, and obviously a plethora of I/O ports. Typical applications included:

digital scales	taxi meters
gas pumps	traffic light
elevator control	vending machines
medical instruments	

Busicom of Japan produced several calculator models using the MCS-4 chip set. Ted Hoff and I made the original proposal for the MCS-4 and did the feasibility study for the first calculator. Federico Faggin did all of the logic and circuit design and implemented the layout; Busicom's M. Shima wrote most of Busicom's firmware. (Later Shima joined Intel as the 8080 designer.) The Intel patent on the MCS-4 (Hoff, Faggin, Mazor) has 17 claims, but the single chip processor is not claimed as an invention.

Intel supported the MCS-4 with a Cross assembler and later with a stand alone development system, the Intellec "blue" box. Intel's marketing efforts of H. Smith, R. Graham, and Ed Gelbach gained attention.

The MCS-4 evolved into the single chip microcomputers 8048/8051 [13]. These chips emphasized small size and low cost. These, along with a variety of other manufacturer's parts have evolved into the under $1 computer on a chip used in toys, automobiles, and appliances [14]. These chips are very pervasive—almost invisible.

III. Intel 8008 Microprocessor

Intel made a custom 512-b shift register memory chip [15] for use in (their customer) Datapoint's low cost bit-serial computer. This 8-b CPU, implemented with TTL MSI, had around 50 data processing instructions. In response to their inquiry about an 8×16 stack chip, and based upon our progress with the MCS-4, I proposed an 8-b parallel single chip CPU in 12/69 [16]. This custom chip design was never used by Datapoint, and it became a standard Intel product, which marketing dubbed the 8008 (twice 4004!).

Although the arithmetic unit and registers were twice as large as in the MCS-4, we expected that the control logic could be about the same if we deleted a few Datapoint defined instructions. Unlike the MCS4's two memory address space, the 8008 had one memory address space for program and data [17]. The symmetric and regular instruction set was attractive. However, the only memory addressing was indirect through the High-Low (HL) register pair. Today's computers have huge amounts of memory, and a plethora of memory addressing instructions.

The 8008 CPU had six 8-b general purpose registers (B,C,D,E,H,L) and an 8-b accumulator. The push down program counter stack had 8-levels. Both of these register arrays were implemented with dynamic memory cells and the CPU had built-in "hidden" refresh during instruction fetch cycles, similar to the MCS-4.

We decided that the 8008 would utilize standard memory components (not custom ROM's and RAM's as in the MCS-4). This increased the parts count on a minimum system because separate address registers, multiplexors and I/O latch chips would need to be added to make the system work; in practice about 40 additional small chips were needed. But standard memories were available in high volume at low cost, and in a larger system the extra chip overhead could be tolerated. Using memory chips with different access times requires a synchronizer scheme, and therefore ready/wait signal pins were provided to perform a handshake function. These interface signals are more sophisticated in today's processors, but the 8008 demonstrated the idea.

The availability of Electrically Programmed ROM's (EPROM) was significant in allowing customers to experiment with their software. A product synergy evolved between Intel's memory component business and the microprocessor.

Intel had an 18-pin package in volume production for the 1k dynamic RAM chip (1103); this gave two more pins for the 8008 than we had on the MCS-4, but we still had to time-multiplex an 8-b bus. By reducing the Program Counter width to 14-b we saved two package pins. The jump instruction contained a 16-b address, but two of the bits were ignored. The 8008 could have 16 k bytes of memory, and at the time, this seemed enormous. (Today, users want 16 meg.)

A. Little Endian

Some have wondered why the addresses in the 8008 were stored "backward" with the little end first, e.g., the low order byte of a two byte address is stored in the lower addressed memory location. I (regrettably) specified this ordering as part of the JUMP instruction format in the spirit of compatibility with the Datapoint 2200. Recall that their original processor was bit serial; the addresses would be stored low to high bit in the machine code (bit-backward). Other computer makers organize the addresses with the "big end" first. The lack of standardization has been a problem in the industry.

B. Applications

One of the first users of the 8008 was Seiko in Japan for a sophisticated scientific calculator. Other uses included business machines and a variety of general purpose computers.

Most of the 8008 instruction set was defined by Datapoint's H. Pyle and V. Poor [16]. Hoff and I wrote the specification for the 8008 single chip CPU. Hal Feeney did all of the chip design under Faggin's supervision. I did the logic simulation for Feeney. Les Vasdasz [18] was our

overall manager. Sandy Goldstein wrote a cross assembler; Gary Kildall (Digital Research) created PLM-8 and then CP/M. This operating system is famous and helped lead to the development of Microsoft's DOS.

Intel did not apply for a patent on the 8008. Datapoint contracted with Texas Instruments in 1970 to get a second source for this chip. TI patented their design, but never got into production [19].

After about one year of experience with programming the 8008 CPU chip [20], we had a number of requested enhancements from our users. We proposed to build the 8080 as a follow on chip; this chip was very popular and led to the microcomputer revolution and the Personal Computer. It is ironic that Datapoint ultimately competed in the marketplace with PC products based upon *their own*, Datapoint defined, architecture!

IV. 8080—More and No More

A. More

Based upon Intel's success with their new microcomputer product line Faggin convinced Vasdasz in 1972 to fund a project to convert the P-MOS 8008 into the newer N-MOS technology. This technology offered about a 2× speedup without making logic changes. After a short study, it was determined that a new mask set was needed because of the incompatibility of transistor size ratios. Faggin reckoned that since a new mask set was needed, he would fix some of the 8008's shortcomings [21].

We evolved the 8080 specification [22] to improve performance 10×. We used the greater density to put in more logic (~4500 transistors) and do more in parallel; the on chip control logic grew by 50%. We put the stack in memory, did 16-b operations, and improved memory addressing. Now 40-pin plastic packages were available, and the address bus and data bus could be brought out in parallel. This design also simplified the external circuitry and TTL voltage compatible signals were provided.

Deleting the on-chip stack saved chip area, but was a net advantage to the user—now the stack had unlimited size. I defined the stack as growing downward from the high end of memory; this facilitated indexing into the stack and simplified displaying the stack. This was abandoned on the 8086.

In the 8080, the registers were arranged as pairs of 8 b, to provide 16-b data handling. The three register pairs were designated as: BC, DE, HL—The High/Low register pair was the only way to address memory in the older 8008. This was limiting to programmers, so in the 8080 direct memory addressing instructions were added, as well as several specialized instructions for the HL register pair. One instruction XTHL provided for exchanging the top of stack with HL; another instruction, XHLD swapped the contents of HL with the DE register pair. As these special instructions were not very symmetric, applying only to HL, we optimized their logic implementation. One of Ted Hoff's tricks was the use of an *exchange flip/flop* for DE/HL This flip flop designated one of the pairs as HL and the other register pair as DE. Simply toggling this flip/flop affected an apparent exchange! This saved a lot of logic; but by mistake, the reset pin had been connected to this flip/flop. An early 8080 user manual stated: "after reset, the HL/DE register contents may be exchanged" (later the reset connection was cut). The lack of instruction set symmetry was a nuisance to programmers and later CPU's instruction sets were considerably more regular; of course there were more transistors "to burn."

B. No More

M. Shima [21] was the 8080 project manager under Faggin. My specification used all 256 operation codes, and 12th from the bottom of my list was an obscure instruction (XTHL) for exchanging the top of stack with the HL register pair. This instruction required five memory cycles to execute, and would be used to pass arguments to subroutines. I carefully explained each instruction to Shima, whose patience was tested as I detailed the XTHL operation. He drew a line under this instruction, and declared:

"No more."

This is why the last 12 instructions were never implemented and why there was room in the instructions set for the 8085 microprocessor's added instructions [23].

The 8080 was very successful in the market. Meanwhile, competition blossomed and a variety of great processors developed [24] including the Motorola 6800 and the MOS Technology 6502. Shima and Faggin (with R. Ungerman) formed Zilog, and competed with an enhanced processor, the Z-80 [25]. The 8080 CPU chip was patented by Intel (Faggin, Shima, Mazor) and has three claims [26].

C. 8085

To meet competition in 1976, Intel decided to develop a more integrated version of the 8080. This chip contained ~6500 transistors. The new N-MOS was more TTL compatible and this chip needed few external parts. There were 12 unused operation codes in the 8080 which provided room to expand the CPU's function. At Intel, a committee studied, argued, and finally decided after many months which instructions to add [27]. Although, all of these new codes were utilized by the 8085 designers, by the time this product got to market it was almost obsolete. To reduce compatibility requirements with the 8086 which was in design, 10 of the new 12 instructions were never announced in the data sheet. They have only been an interesting historical anomaly and a lesson about design by committee.

D. 8086

In 1978, Intel's W. Davidow, vice president of the microcomputer group, rushed to staff a 16 b microcomputer development project. It was to have around 30 000 transistors [28], 12 times more than the 4004. This new computer had multiplication and division and a host of other new features [29]. However, it was constrained to be upwardly compatible with the 8080 (and 8008). Accordingly, the

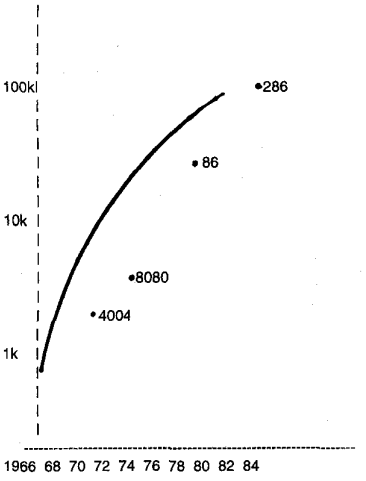

Fig. 1. MOS transistor per chip—1966 forecast (solid line) and actual (dots).

designers decided to keep the 16 b basic addresses and to use segment registers to get extended 20 b addresses. Two versions were created—the 8088 had an 8-b data bus for compatibility with 8-b memory systems, and the 16 b 8086 [30].[1] With 1 megabyte of memory addressing, this processor was a serious contender in the computer market place. This chip density required to match the 16-b minicomputers was "arriving" as had been predicted [31].

The decision by IBM to use the 8088 in a word processor and personal computer created enormous market momentum for Intel. The 186, 286, 386, 486 followed over the next 15 years, with some shadow of 8008 features still apparent. These components would be "truly pervasive" [32].

V. Historical Perspective

A. Technology Predictions

The promises of high density solid state circuitry were becoming apparent in the 1950's. In 1959, Holland contemplated large scale computers built with densities of 10^8 components per cubic foot [34]. The integrated circuit was developed in parallel at both TI and Intel. Technology forecasts were made by Fairchild's Gordon Moore and Robert Noyce in the mid-1960's—the density of IC's was doubling every year [35]. "Entire subsystems on a chip" were predicted if a high volume standard chip could be defined. By 1966 Petritz of TI was forecasting about 10 k transistors per chip for 1970 and 100 k (optimistically) by 1976 [31]. See Fig. 1 for a 1966 forecast of chip complexity. It was then estimated that abut 10 k–20 k gates would fit on a chip and that a good portion of a CPU would therefore be on one chip.

In 1966, Hobbs forecasted the reduced cost of arrays, predicting that the CPU cost would become "negligible" [5]. Practical people recognized that the issues were the "number of unique part numbers and the production volume" after all only a few thousand computers were made each year [36].

[1] The 8086 had a large staff. If a few names are to be mentioned they are S. Morse, W. Pohlman, B. Ravenel, J. McKevitt, J. Bayliss, and in Marketing, D. Gellatly [33].

B. SSI, MSI, LSI Chips

By 1968 16-b minicomputers utilized a single printed circuit board CPU containing around 200 chips. These were medium scale integrated circuits (MSI) with $\sim$ 100 transistors per chip, and small scale IC's (SSI). Obviously, the more transistors that could be put on a chip, the fewer chips needed on a PCB. Since manufacturers were trying to reduce costs, there was a constant battle to reduce the number of chips used—could a CPU be built, with 150, 80, or 25 chips?

By 1970 there were a few projects to build a 16 b minicomputer CPU using multiple LSI chips. A 1000 transistor chip would be called large scale (LSI). These projects were being done with military sponsorship at Raytheon and RCA. The air force was especially interested in light weight airborne minicomputers. These were full 16-b minicomputers and did not have a scaled down specification (like the MCS-4), except for their physical size [37], [38]. They utilized 4-b or 8-b arithmetic and register "slices"; a minimum CPU would require 8–12 LSI chips with about 6–8 different part numbers. These were R&D projects [39], [40].

C. LSI Economics

The use of custom LSI in an application required very high production volume to commercially justify the significant tooling costs. One would need to produce around 100 000 systems for commercial feasibility. The only high volume commercial applications in the early 1970's were calculators; almost every calculator manufacturer was designing custom LSI chips. These chips were invariably very specialized for arithmetic, printers, and keyboards—(Busicom's original request).

Besides tooling costs, another problem is to get an economic die size. If a die is too small it does not contain enough circuitry to justify a fair price. If a die is too ambitious and large, the manufacturing yield will be too low and the chip will be too expensive [41]. See Fig. 2 for an illustration of complexity, cost, and yield. Worst of all, at the beginning of a complex chip project it is not easy to accurately forecast the final die size. Defining standard high volume LSI chips is challenging [36], [42].

Consequently, in 1970, no one had defined general purpose LSI building blocks that were usable in a variety of applications. The only LSI building blocks available were memory chips. Honeywell tried to get multiple sources for a 64-b bipolar LSI memory chip, but that was on the leading edge of bipolar technology, and not many vendors could make them [43]. Metal gate MOS ROM's and 200-b shift registers were available from a few sources: AMI, Electronic Arrays, MOS Technology, and General Instruments. See Table 1 for 1965 LSI chip examples. Although these chips had around 1000 transistors, they were

Table 1 MOS Chip Availability in 1965

Manufacturer	Type	Transistors	Power, mW	Pads
GME	100-b shift register	600	200	12
GI	21-b static shift register	160	150	11
TI	Binary-to-digital decoder	150	25	26

very regular in structure and easy to design. Because their internal wiring was minimal, they were $2x-5x$ more dense than "random logic" chips.

D. Partitioning into Packages

One difficulty implementing any system on a set of LSI chips is partitioning [44] into pieces with a reasonable number of I/O pins on each. It was very expensive to get more than 20 pins. Around 1970 there were very few commercially available low cost packages. The most common had only 14-pins and sold for around $1. Cost sensitive applications such as desk calculators could not afford 48-pin packages which were then selling for around $10.

Optimization consists of maximizing the number of gates inside compared to the number of pins outside—the *gate to pin ratio*. Memory chips with 1 kb in an 18-pin package gave an excellent gate/pin ratio of about 100:1. Each time the technology allowed a doubling of bits on a chip, only one more address pin was needed. A shift register was even better, because regardless of the number of bits added, the input/output pin count stayed constant.

If a CPU were to be built of LSI chips it was not obvious how to break it into pieces with a small number of I/O pin connections and a high gate/pin ratio. Simply put, if you cut an ordinary CPU into two pieces you would have hundreds of signals which would need to cross the chip boundaries.

Each package pin also required a lot of MOS chip "real estate" for amplifiers to drive the heavy off chip capacitive loads and for the wire bonding pads which go from the chip to the package. Besides the cost, placing more pins on an LSI chip also lowered the reliability. Hence, most commercial LSI applications were constrained by the few leads available on IC packages. This is why the early microprocessors were in 16 and 18 pin packages.

E. Semiconductor Technology

On-chip interconnections are also a major problem. A CPU chip contains "random logic" requiring many interconnection wires. Prior to 1980 most semiconductor chips had only one layer of metal. This metal was used for global connections such as power, ground, clocks, and major busses. Local connections were made using poorer quality, higher resistance, lines of poly-silicon or diffusion.

The silicon gate process [18], developed originally at Fairchild Semiconductor in around 1967, provided slightly better local interconnections and crossovers. This technology also offered lower capacitance, smaller size (self-aligned structures) and lower voltage operation. This was a key technology enabler for microprocessor development at Intel. The 8008 chip for Datapoint was implemented using silicon gate technology. In contrast, TI, was at that time, using metal gate MOS technology and used about twice the amount of silicon area for a similar chip.

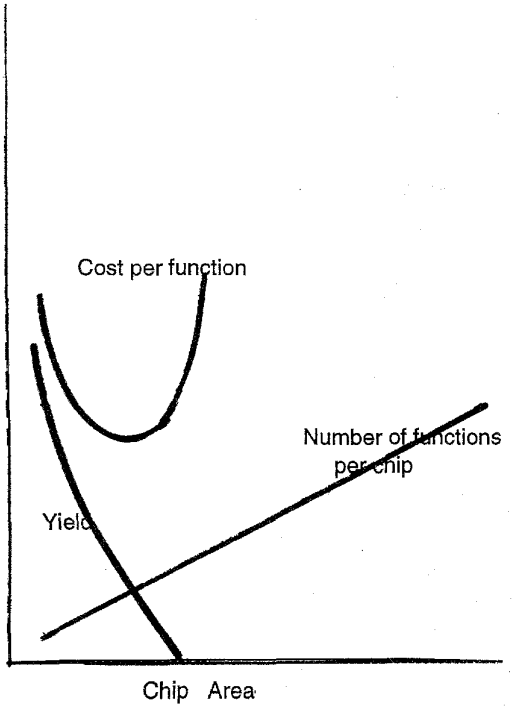

Fig. 2. Relationships of complexity, cost, and yield to chip area.

Silicon gate P-MOS needed a 14 V supply, and was often biased between +5 V and −9 V to give pseudo-TTL compatibility. This relatively high voltage aggravated the severe power budget facing the circuit designer. Small IC packages cannot dissipate more than 1/2 W of power in normal air cooled systems. The compromise was to use dynamic logic operating at low duty cycles to reduce heat. In 1995, power dissipation is still a major design factor in commercial system design. It has been one of the driving factors toward 1.5–2.5 V technology; battery operation is another factor.

F. Circuit Factors

The P-MOS transistors in 1970 required 14 V to operate. To reduce the overall power dissipation most of the circuits were operated dynamically in a two phase operation. First a circuit was precharged using an on-chip amplifier, and then the circuit was conditionally discharged, based upon logic decisions. Previously, "bootstrap" amplifiers were built using the gate "overlap capacitance" as part of the circuit. However, silicon gate self-aligned geometry eliminated this capacitor. F. Faggin innovated a new and efficient bootstrap amplifier as part of his early circuit design of Intel's chips.

Another element which made micro's feasible was the dynamic RAM cell. The memory storage is obtained by storing a charge on a small capacitor. This capacitor is usually integrated into a three-transistor memory cell. However, the memory starts to fade after about 5 ms, so that an external "refresh" circuit needs to read, test, and restore the charge on a periodic basis. Static memory cells required twice the chip area and used much more power; they were impractical for use inside the CPU. Recall that Intel was only a memory company in 1970. Hoff had done research on memory cell design and, proposed to use dynamic RAM inside the CPU for index registers and stack. Hoff's insight was essential for enabling the first microprocessor chip.

G. CAD Tools

Since the mid-1960's computer makers had been doing circuit analysis using "home grown" tools. Hoff and I developed a transient analysis program (PULS) to help with MOS circuit design. Intel's Dov Frohman, who invented the EPROM (he didn't call it a FROM), provided the transistor model. Intel's designers used PULS to help them achieve the desired ac/dc performance. Hoff wrote our first logic simulator for the PDP-8; later I used a commercial (Applicon) tool for the PDP-10. I abused the DEC macro assembler to get the first MCS-4 code assembled and into the 4001 ROM bit map. We developed the early calculator firmware with this assembler. The availability of these CAD tools allowed our designers to catch design errors early and were essential to Intel's success. A few Silicon Valley CAD companies were spawned from these in house CAD groups.

H. The Microcomputer Name

In the mid-1960's midicomputers and minicomputers were selling in the marketplace. Some computers used a microprogram, stored in ROM; the inner part of such a computer was called an "engine" or "microengine" or "microprocessor." In 1970, a microcomputer was normally interpreted as a computer considerably smaller than a minicomputer, possibly using ROM for program storage. By extension, the terms "nano-computer" and "pico-computer" have also been used by computer engineers indicating relative size and performance of computers.

In the late 1960's Fairchild had a logic family called μ-logic, so the prefix was also used for "micrologic" in IC's. (Since most of the Intel guys had come from Fairchild we avoided references to their product line; Intel did not use the Greek letter).

Lo [45] mentions "the computer on a chip" in 1968, and *Scientific American* also featured "Computer on a Chip," with 400 gates in 1970 [2]. IBM looked for ways of simplifying computers. In 1968, Hitt proposed a 4-b computer with no arithmetic unit and no registers (CADET—can't add doesn't even try). But this very simple computer was not built with LSI and was still called a minicomputer [11].

The single chip central processor unit (CPU) has been commonly called a microprocessor. With off chip memory, it is usually called a microcomputer. Single chip computers are often called microcontrollers. The 4004 specification was for a microcomputer.

VI. Summary

Integrated circuit technology has been evolving in a predictable manner for the past 30 years. Although a computer on a chip was eventually realizable, it was problematical how to use LSI chips which had fewer than 20 000 transistors. Most of the work focused on partitioning 16-b computers into multiple chips, but few of these projects were successful. Early Intel microprocessors succeeded because they were scaled down computers. Like a golf cart, they were very limited, "but got across the green." When the densities reached 200 k+ transistors per chip, microprocessors became the dominant computer technology.

References

[1] "The alternative," Intel Corp. brochure, 1971.
[2] F. G. Heath, "Large scale integration in electronics," *Scientif. Amer.*, p. 22, Feb. 1970.
[3] G. Blynsky, "Little chips invade the memory market," *Fortune Mag.*, pp. 100–104, Apr. 1971.
[4] M. E. Hoff, S. Mazor, and F. Faggin, "Memory system for a multi-chip digital computer," US Patent #3,821,715, Intel Corp., June 1974.
[5] L. C. Hobbs, "Effects of large arrays on machine organization and hardware/software tradeoffs," *1966 FJCC*, vol. 29, p. 89.
[6] "MCS-4 micro computer set," data sheet #7144, Intel Corp., 1971.
[7] F. Faggin *et al.*, "The MCS-4—An LSI micro computer system," *IEEE Region 6 Conf. 1972*, pp. 8–11.
[8] H. Smith, "Impact of LSI on microcomputer and calculator chips," *IEEE NEREM '72 Rec.*
[9] J. Karp, A. Regitz, and S. Chou, "A 4096-bit dynamic MOS RAM," *ISSCC Dig. Papers*, pp. 10–11, Feb. 1972.
[10] S. Mazor and D. Hall, "Microprocessor software debounces input switches," *Design News*, vol. 34, pp. 109–114, June 1978.
[11] D. C. Hitt *et al.*, "The mini-computer—A new approach to computer design," *IEEE 1968 FJCC*, IBM Corp., pp. 655–662.
[12] R. Hooper, "The minicomputer, a programming challenge," *1968 FJCC*, pp. 649–654.
[13] S. Mazor and L. Goss, "A new single chip microcomputer for control applications," *IECI 77*, pp. 109–112, Mar. 1977.
[14] S. Mazor and J. Haynes, "Blood analyzer with one-chip microcomputer," *Medical Electron.*, vol. 10, no. 5, pp. 49–51, Oct. 1979.
[15] M. Hoff and S. Mazor, "Operation and application of shift registers," *Comput. Design*, pp. 57–62, Feb. 1971.
[16] V. Poor, "Letters," Datapoint, *Fortune Mag.*, p. 94, Jan. 1976.
[17] M. Hoff, "The new LSI components," *6th Annu. IEEE Comput. Soc. Int. Conf.*, 1972.
[18] L. Vasdasz, A. Grove, G. Moore, and T. Rowe, "Silicon gate technology," *IEEE Spectrum*, pp. 27–35, Oct. 1969.
[19] G. Boone, "Computing system CPU," US Patent #3,757,306, Texas Instrum., Sept. 1973.
[20] *Intel MCS-8 User Manual*, 1972.
[21] M. Shima, F. Faggin, and S. Mazor, "An N-channel 8-bit single chip microprocessor," *IEEE ISSCC*, pp. 56–57, Feb. 1974.
[22] *Intel MCS-80 User Manual*, 1974.
[23] M. Shima, *The Birth of the Microcomputer: My Recollections*. Tokyo: Iwanami Shoten, 190 pp.
[24] G. Bylinsky, "Here comes the second computer revolution," *Fortune Mag.*, Nov. 1975.
[25] F. Faggin, "Letters," Zilog, *Fortune Mag.*, p. 94, Jan. 1976.
[26] F. Faggin, M. Shima, and S. Mazor, "Single chip CPU," US Patent #4,010,449, Intel Corp.
[27] *Intel MCS-85 Users Manual*, Mar. 1977.
[28] S. Morse, W. Pohlman, and B. Ravenel, "The Intel 8086 microprocessor," *Comput.*, pp. 18–27, June 1978.
[29] S. Mazor, "Programming the 8086," *Comput. Design*, Dec. 1980 to Feb. 1981 (3 parts).

[30] *Intel MCS-86/88 Users Manual*, July 1978.
[31] R. Petritz, "Large-scale integrated electronics," in *Proc. FJCC*, 1966, pp. 65–87.
[32] P. E. Haggerty, "Integrated electronics—A perspective," *Proc. IEEE*, vol. PROC–52, pp. 1400–1405, Dec. 1964.
[33] S. Morse, B. Ravenel, S. Mazor, and W. Pohlman, "Intel microprocessors 8008 to 8086," *Comput.*, pp. 42–60, Oct. 1980.
[34] J. Holland, "A universal computer capable of executing an arbitrary number of sub-programs simultaneously," in *Proc. 1959 EJCC*.
[35] R. Noyce, "A look at future costs of large integrated arrays," in *Proc. 1966 FJCC*, pp. 111–115.
[36] M. E. Conway and L. M. Spandorfer, "A computer designer's view of large scale integration," *1968 FJCC*, p. 835.
[37] R. K. Booher, "MOS GP computer," *1968 FJCC*, vol. 33, pp. 877–886.
[38] J. J. Pariser and H. E. Maurer, "Implementation of the NASA modular computer with functional characters," *1969 FJCC*, p. 231.
[39] F. D. Erwin and J. F. McKevitt, "Characters—Universal architecture for LSI," *1969 FJCC*, vol. 35, p. 69.
[40] A. Alaspa and A. Dingwall, "COS/MOS parallel processor array," *IEEE J. Solid-State Circ.*, vol. SC-5, Oct. 1970.
[41] Philco Ford, "Large scale integrated circuit arrays," TR# AFAL-TR-69-23, Air Force Avionics Labs, June 1969.
[42] H. G. Rudenberg, "Large scale integration: Promises versus accomplishments—The dilemma of our industry," *1969 FJCC*, vol. 35, p. 359.
[43] Intel 3101 64-bit Static RAM Data Sheet, 1970.
[44] N. Cserhalmi *et al.*, "Efficient partitioning for the batch-fabricated fourth generation computer," *1968 FJCC*, vol. 33, pp. 857–866.
[45] A. W. Lo, "High-speed logic and memory—past, present, and future," *1968 FJCC*, vol. 33, pp. 1459–1465.
[46] R. Noyce and M. Hoff, "A history of microprocessor development at Intel," *IEEE Micro*, vol. 1, no. 1, pp. 8–21, Feb. 1981.
[47] S. Mazor, "Programming and/or logic design," *IEEE Comput. Group Conf.*, June 1968, pp. 69–71.
[48] M. E. Hoff and S. Mazor, "Standard LSI for a microprogrammed processor," in *NEREM '70 Proc.*, Nov. 1970, pp. 92–93.
[49] S. Mazor, "A new single chip CPU," *Compcon*, pp. 177–180, Feb. 1974.
[50] "Intel advertisement," *Electron. News*, Nov. 1971.
[51] S. Mazor, "VLSI computer architecture issues," *Process. and Devices Symp.*, Electron Devices Group, Santa Clara Valley, CA, Apr. 1981.
[52] S. Mazor and S. Wharton, "Compact code—IAPX 432 addressing techniques," *Comput. Design*, p. 249, May 1982.
[53] L. Jack and S. Mazor, "Rapid VHSIC insertion through the application of silicon compiler technology," in *Dig. 1985 GOMAC Conf.*, Nov. 1985, p. 117.
[54] S. Mazor and P. Langstraat, *A Guide to VHDL*. New York: Kluwer, 1994.
[55] G. Hyatt, "Single chip integrated circuit computer architecture," US Patent #4,942,516, July 1990.

Stanley Mazor (Senior Member, IEEE) studied mathematics at San Francisco State College.

In 1964 he joined Fairchild Semiconductor R&D, where he helped specify and implement the Symbol high-level language computer. From 1969 to 1984 he was with Intel Corporation, where he worked on the specification of the early Intel microprocessors. He also supervised Intel's microcomputer training development group. He also spent two years in Brussels as an Applications Engineer supporting European customers. From 1984 to 1988 he was the Director of Customer Engineering Services at Silicon Compiler Systems (SCS), where he developed application-specific IC's. In 1988 he joined Synopsys, where he was Technical Training Manager. He is presently Director of Technical Services at C-ATS, Palo Alto, CA. He also taught courses at the University of Santa Clara and Stanford University, and has been a Guest Professor in China, Finland, and Sweden. He has published over 45 articles and papers on the design and application of VLSI, including signal processing, instrumentation, security, and optimization.

Mr. Mazor received the Best Paper Award for his GOMAC contribution on VHSIC insertion in 1986. He is active in the COMPCON program committee and the Asilomar Microcomputer Workshop.