It all started with a model plane
(Faggin oral history)

Federico Faggin grew up in Vicenza, a small town in Italy, during the Second World War. After the war, when he was 11 years old, he encountered a man with a wind-up model airplane. Seeing this toy that could actually fly changed the course of Faggin's life. He decided to build one. Lacking the money for a kit, he visualized the plane, drew up plans, and then built it from balsa wood and dressmaker's tracing paper.

This was his first experience with the complete process of designing a product, building it, and testing it out. Unfortunately the first plane didn't fly - as Faggin put it, it violated all the aerodynamic laws in the universe. The man with the wind-up plane came by, and after laughing uproariously at Faggin's airplane, he explained the necessary principles.

Faggin built a second airplane. Unfortunately that airplane didn't fly either. But the third airplane flew.

This led Faggin into an interest in model planes, which led him to study aeronautical engineering in high school. (This was an intensive high school, where he studied calculus and partial differential equations.)

In this time frame, the 1950s, radio-controlled model planes were being developed using tiny vacuum tubes. Reading about these radio-control systems was the start of Faggin's interest in electronics. When his high school stopped the program in aeronautical engineering - perhaps it wasn't as popular a high school program as expected - he switched to radio electronics because of his interest in radio controlled model planes. With his interest in electronics, it wasn't long until Faggin became fascinated with computers.

The remote control model plane technology of the 1950s is interesting. Because transistors hadn't been invented yet, the transmitter and receiver used vacuum tubes. An example is the ED Boomerang, which used a DL68 pentode vacuum tube in the receiver to drive a small relay. A L-C (inductor-capacitor) circuit connected to the three-foot antenna provided the tuning.

The tube required considerable voltage, so it used a 67.5 B battery, and a 1.5 volt A battery. Another version used a XFG1 gas-filled triode thyratron. (reference http://www.modelenginenews.org/cardfile/ed_radios.html). (The DL68 was a sub-miniature tube of 9 mm x 36 mm, roughly the size of a hearing aid, and also used in hearing aids. http://www.hupse.eu/radio/tubes/DL68.htm
http://www.mif.pg.gda.pl/homepages/frank/sheets/030/d/DL68.pdf)
Control motors and servos weren't used back then. A typical system only controlled the rudder, and used a wind-up rubber band for power. The radio control would release an escapement latch, allowing the rubber band to rotate a cam that would turn the rudder to the left or right. You had to keep track of how much you wound the rubber band versus how many "clicks" of the rudder you used, or you'd run out of rudder control and the plane would fly off uncontrolled.
(Escapement reference
http://www.ctie.monash.edu.au/hargrave/tribute_essays.html)

With his highly-technical high school education and interest in electronics, Faggin went to work at Italian electronics company Olivetti at the age of 19, where he worked as an assistant engineer in the Olivetti computer R&D lab. It was there that he got his start designing computers.

In his new job he spent a few months learning about transistors and logic circuits as part of a project that was building a computer. When the leader of the project had to go into military service, Faggin took on a larger role under the guidance of a more senior engineer. After the senior engineer got into a car accident and was off work for three months, Faggin ended up at the age of 19 completing the computer design with 5 technicians working for him.

This computer was a relatively simple rack-based computer in 1960/1961. The rack was full of small PC boards, each with a couple flip flops or gates. The logic was DTL using germanium transistors. Faggin's role was designing the control logic for the computer. The computer used Nixie tubes (cold-cathode tubes containing neon that typically displayed digits) for output, as well as a teletype for input. This computer showed the feasibility of a small useful computer, and helped lead to the Olivetti Programma 101. The

computer was a decimal computer about 7 feet tall. It used ring counters for the accumulator. The computer was an adaptation of a published American design (maybe Communication and Electronics - Volumes 41-46 - Page 251 or Electronics - Volume 34 - Page 155

). It had 4K-word magnetic core memory and 12-bit word. The accumulator drove the Nixie tubes directly, since the accumulator stored decimal digits. The computer performed addition and subtraction by presetting the counters, and then counting pulses. A ripple carry handled carries from one digit to the next.

Maybe this is based on Digital counters and computers, Edward J. Bukstein, 1960

Olivetti started in 1908 as a typewriter company, starting the first Italian typewriter factory. Olivetti branched into calculators and teleprinters in the 1940s, becoming a leader in mechanical office products in the 1950s. Olivetti started moving into the new field of electronics in the 1950s, creating Italy's first computer the entirely-transistorized Elea 9003 in 1959. (This was several months before IBM came out with their first transistorized computer, the IBM 7090.) The Elea 9003 used diode-transistor logic.

Each computer needed 300,000 transistors and diodes, which convinced Olivetti to build the company SGS (Società Generale Semiconduttori) in 1957. In 1987, this company merged with Thomson Semiconducteurs of France to form SGS-THOMSON Microelectronics. (At this point, Thomson SA owned Mostek, a company that will play a significant role later.) In 1998 the company was renamed STMicroelectronics, and is currently one of the world's largest semiconductor companies. Among the products they make are ARM microcontrollers and mobile chips. It's interesting to note that one of the top semiconductor companies in the world grew out of a company built to support Olivetti's computer manufacturing in the 1950s.

The Elea 9003 led to the Elea 6000 and Elea 4000, but Olivetti sold its computer business to General Electric in 1964. One computer product that

escaped the sale to General Electric was the Programma 101, the first commercial desktop personal computer (although it could also be considered a programmable calculator). It escaped the sale when Programma 101 developer Gastone Garziera spent a couple evening changing its description in internal documents from “computer” to “calculator”. The situation in the office became awkward, though, as GE ended up owning the building in which the Programma 101 developers worked, except for their office. The Programma 101 motivated much of the design of the HP 9100, and Hewlett-Packard ended up being forced to pay royalties to Olivetti. Ref http://royal.pingdom.com/2012/08/28/the-first-pc-from-1965/
It seems that Olivetti had a bigger role in computing than people generally realize.

In 1982, Olivetti released a modern personal computer, the Olivetti M20, which used a Z8000 CPU. This CPU was, of course, built by Faggin's company Zilog, closing the circle.

Reference: http://en.wikipedia.org/wiki/Olivetti_Elea

Reference: http://www.olivetti.nu/history.htm

Faggin then studied physics at the University of Padua, distinguishing himself with his performance, finishing a six-year degree in four years. He studied physics in part because he wanted to understand more of the fundamentals of transistors. He did a thesis project on flying-spot scanners, designing an opto-feedback loop to make a scanner with enough accuracy to scan bubble chamber photographs. Within a year, he designed, built, and debugged this scanner.

In 1966, he worked for his old Olivetti boss at a startup CERES in Milan. The startup was developing thin film circuits, which were a way of building complete circuits in a package before integrated circuits. This position was important, because the company served as a representative for General Micro Electronics, a MOS company spun off from Fairchild. Faggin ended up

traveling to Silicon Valley in 1966 to learn about MOS circuits from GMe. Unfortunately MOS devices were slow, unreliable, and hard to manufacture because they used metal gate technology. Philco-Ford bought GMe and canceled CERES's rep role, and removing them from the MOS business.

At this point, Faggin left CERES for SGS-Fairchild, which was the only semiconductor company in Italy at that point. They built transistors and power transistors. They had a license to make planar transistors from Fairchild, who had invented the “planar process” for transistors in 1959. And Fairchild owned 30% of the company.

In 1966 after obtaining his university degree, Faggin worked in the SGS laboratories near Milan and developed a method of manufacturing MOS integrated circuits and designed their first integrated circuits. http://dl.ifip.org/db/conf/ifip3/histedu2008/Parolini08.pdf

Faggin took an R&D position at SGS-Fairchild, where he was given the task of designing and developing MOS circuits, based on his experience in this area (which as he points out consisted of his two-week visit to Silicon Valley). Despite his minimal experience, he had designed two MOS integrated circuits that went into production.

At the end of the year, shortly after his marriage, he went on a six-month exchange to Fairchild's Palo Alto Research Laboratory.

At Fairchild, he worked on the development of self-aligned gates, using silicon gates instead of metal. This project would turn out to have a huge impact on the success of MOS integrated circuits. Metal gates had a threshold voltage of 5 to 7 volts, so MOS transistors required large voltage swings to turn on. This lead to inconveniently large supply voltages of, say, 24 volts. Silicon gates had a much smaller threshold voltage, which made them much better. In addition, the self-aligning process made manufacture much simpler.

In addition, metal gates required a thick oxide layer to prevent an unintended parasitic transistor from forming where a metal line crosses two junctions. However, thick oxide made the chips less reliable, as the aluminum metal

traces had to traverse relatively large steps going over the oxide.

The silicon used had a Miller index of (1 1 1). This roughly means that the plane used as the surface of the silicon wafer was at an angle to the silicon lattice. This moved to (1 0 0), where the plane was parallel to the silicon lattice. (Ref: http://www.intel4004.com/mosgate.htm) The change in silicon orientation reduced that threshold voltage, saving power, but this increased the risk of parasitic transistors. To prevent the parasitic transistors required isolating parts of the chip with “channel stoppers”, which took up space and made the chip half as dense. Thus, the dilemma was bad power consumption or bad density.

Another problem was that because masks for the different layers of the chips weren't perfectly aligned, the gate needed to be made larger than desired to handle any “slop” in the alignment of the masks. If the gate metal didn't cover the entire gate, the transistor would be defective. The problem with a larger gate is it formed a capacitor where it overlapped the source or drain. This parasitic capacitance reduced the performance of the chip. Due to the Miller effect, the transistor amplified the effect of this capacitance, making it even worse.

One solution was ion implantation, which gave much more control over the doping levels, allowing the threshold voltages to be reduced.

The self-aligned gate technique was the key solution to this problem. In 1966, Robert Bower realized that if you created the gate first, the gate itself could be used as the mask for creating the source and the drain. As a result, the gate would be perfectly aligned. The problem was that Bower used aluminum for the gate, and aluminum couldn't handle the high temperatures necessary for annealing the silicon and creating a silicon dioxide layer. Bower used the new ion implantation technique, which was only available at Hughes (where he worked) at that time, but it wasn't sufficient to make the technique practical. The relevant patent is https://www.google.com/patents/US3472712

Tom Klein at Fairchild discovered in 1967 that using amorphous silicon instead of metal reduced the threshold voltage by 1.1 volts. This meant that

the threshold voltage for real transistors (with silicon gates) would be 1.1 volts lower than the threshold voltage for the parasitic transistors (with metal gates). This made it much easier to prevent the parasitic transistors. This allowed the use of 111 silicon. However, he couldn't figure out a process for making self-aligned transistors in this configuration.

This was the situation when Faggin was put in charge of developing self-aligned transistor process in 1968. Fairchild didn't have a process for creating and etching silicon gates, so Faggin developed that. Up to this point, etching raw silicon wasn't a step in chip fabrication, so Faggin had to develop appropriate chemicals to do this. He experimented with mixtures of nitric and hydrofluoric acid until he developed a mixture that etched at a reasonable rate. He also had to develop new techniques for the photo-resist that would work with this process. After a couple weeks, he had a working etch process, although he burned a hole in his shoe in the process.

Another Faggin invention was the “buried contact”, a method to create a contact directly between the silicon junctions and the amorphous silicon above that formed the gate. The standard technique was to connect these two layers with a metal contact. But by using a buried contact, the connection could be made without using the metal layer. This permitted denser chips, since interconnections could be done on two layers. Because Faggin was working both on the fabrication process and on logic design, he realized the benefits of manufacturing chips with buried contacts. Faggin's boss, Les Vadasz, didn't think buried contacts would work, but Faggin proceeded despite the objections. When he was fabricating a test chip, he added some test patterns to experiment with the poly-to-silicon connections. The reesults of this test showed that the buried contact technique worked. The technique was first used commercially by Intel for the 1103 dynamic RAM. Faggin also used it in the 4004 microprocessor.

One interesting thing about this time period is that chip fabs were relatively inexpensive, so each department at Fairchild had a separate chip fab. This made it easier for Faggin to run experiments and have hands-on supervision of the fab process.

FAggin's exchange from SGS-Fairchild to Fairchild in Palo Alto turned into a permanent move to the United States when Fairchild sold their interest in SGS-Fairchild. When he was given the chance to stay in California instead of returning to foggy Milan, Faggin jumped at the chance.

Coincidentally, the day Faggin started as an employee at Fairchild was the same day that Noyce and Moore left Fairchild to start Intel. Shortly after, Andy Grove left, followed by Faggin's boss Vadasz.

When Intel hired away Fairchild's polysilicon evaporation technician, who had the key knowledge on manufacturing silicon gate technology chips, Faggin became suspicious that Intel was going to steal their silicon gate technology. Since Fairchild was slow in moving technologies from R&D to production, Faggin was worried that Intel might jump ahead of them.

This turned out to be the case. Intel started manufacturing silicon gate technology chips. Faggin suspects that his new boss at Fairchild blocked patenting of this technology because he was hoping for an Intel job. The result was Fairchild never got a general patent on silicon gates (although they later got a narrow patent), or on buried contacts.

A few years later Vadasz received a patent on buried contacts, http://www.google.com/patents/US3699646. Needless to say, this displeased Faggin. This was one of the factors motivating Faggin to leave Intel and form Zilog.

Creating a buried contact required one additional processing step - opening a hole in the silicon oxide layer where the contact was desired, before the gate was created. The following diffusion step would then diffuse boron into both the underlying silicon and the amorphous silicon of the gate, connecting them.

Faggin had created working MOS transistors using this process by April 1968. Shortly after, he designed the Fairchild 3708, the first integrated circuit

using self-aligned gates. This chip was an update of Fairchild's 3705, an 8-bit analog multiplexer. Comparing the 3708 to the 3705, the newer chip with self-aligned gates was 5 times faster, had 1% of the leakage current, and the switch resistance was 3 times lower, showing the benefits of self-aligned gates.

The amorphous silicon gates were soon replaced by polysilicon, which was more reliable. Polysilicon became a key interconnection layer in chips such as the Z80, and will play an important role in the later discussion. Polysilicon is still used as the gate material in MOS and CMOS integrated circuits.

One side-effect of polysilicon gates is the chip could handle much higher temperatures during processing than with aluminum gates. This allowed the use of “phosphorous gettering”, a technique used to move impurities away from the wafer surface, which improves performance.

One drawback of silicon gate technology was that it made creation of capacitors on the chip difficult and expensive compared to metal gate technology. Extra processing steps would be required, increasing the cost of the chip.

A key need for capacitors was in the “bootstrap load” that pulled up the output voltage of a MOS gate. Without this, the output voltage would be lower (by the threshold voltage). This output was enough to turn on the transistor it was connected to, but just barely. If another transistor were added in the sequence (specifically a pass transistor), the voltage swing would not be sufficient. Pass transistors were critical for complex logic circuits - they are used in large number in the Z80 - so being unable to use them was a major drawback.

In more detail, metal gates with 111 silicon have typically 5 to 7 volts of threshold and 24 volts power (Vdd). Subtracting the threshold voltage from the output was a big hit, causing the output to be maybe 12 volts. In addition, this output voltage would be approached relatively slowly. The result was the transistors would switch slowly.

The bootstap capacitor: one terminal of the gate is charged to a potential close to the power supply level, and then it is capacitively bootstrapped to a level above the power supply. The voltage swing on the low side of the capacitor boosts the voltage on the high side, which is used to drive the transistor.

The lack of capacitors for bootstrap load led to resistance at Fairchild to the use of silicon gates. Another source of conflict when transferring the silicon gate technology to production was the engineers didn't believe the silicon gate devices would be smaller. Eventually Faggin discovered they were applying the old metal gate design techniques to chips with silicon gate, and that didn't work well. They needed to change their way of thinking and designing to the new technology. Even when Faggin showed them to make a compact silicon gate circuit, they'd say that it was a special case but wouldn't work in general.

After about a year, Faggin figured out how to build capacitors with silicon without requiring an additional processing step. In 1970, after moving to Intel, he used this technique in the Intel 4004 chipset. These capacitive bootstrap loads are used rarely in the 4004 - most loads are simple transistors - but they are used when the full voltage swing is needed, to drive pass transistors or superbuffers (push-pull devices where both the pull-up and pull-down transistors are switched, similar to CMOS).

Faggin developed silicon gate technology using N-channel transistors, which led to him working on CMOS. He also invented thin film MOS devices with polysilicon. However, he had trouble getting Fairchild to make products out of these technologies. In addition, Intel steadily drained Fairchild's employees, taking more than 30 people in a year and a half. Finally, Faggin wanted to do more chip design and less process development. Frustrated, he called his old boss Les Vadasz at Intel who rapidly hired Faggin for the 4004 microprocessor project. Faggin's computer design work at Olivetti was a big factor in getting this position.

The 4004

Calculator chips of the 1970s were often very strange. Although the chips were programmable (in ROM), they had peculiar instruction sets. Not surprisingly, the registers and operations were designed for decimal calculators. But this was taken to an extreme. Much of the chip (including the instruction set) was designed for an attached keyboard and a seven-segment display.

For example, the TMS0800 series of calculator chips was used in 1974 for calculators such as the Datamath and Sinclair Scientific. The opcodes were 11 bits in length, with the program stored in a ROM that was 11 bits wide. The register set consisted of three registers (A, B, and C) that were each 11 decimal digits wide, and two 11-bit flag registers (AF and BF), which were used to store bits.

The instruction set included operations such as add, subtract. The interesting thing is these instructions could add all 11 digits in one instruction. Moreover, a mask could restrict which digits to add, so if you wanted to add just the top two digits (maybe an exponent), that could be done in a single instruction. Other instructions supported decimal shifts left and right. There were no Boolean operations.

One unusual instruction that shows how tied the instruction set was to calculators was an instruction to increment accumulator while scanning for a keypress, with the key number in the accumulator at the end. Another instruction turned off the display and waited for a keypress on the keyboard.

The Texas Instruments TMS1000 4-bit microcontroller chips were used in calculators in 1972 and sold generally in 1974. While this was became a highly-popular general purpose microcontroller, its roots as a calculator chip are visible in some of its strange features.

It has 4 inputs, a small number for a microcontroller, labeled K1, K2, K4, and K8 (presumably K for Keyboard, since they read 4 columns of keys). The 11 R outputs were typically used to scan across rows of keys and strobe 11

display digits simultaneously. The O outputs are 8 bits, but generated internally from 5 bits. A PLA converts these 5 bits to 8 outputs, so they cannot be independently controlled. Typically the 8 outputs were used to drive a 7-segment display (plus period).

The TMS1000 had three 4-bit registers: A, X, and Y. The standard instruction set had the instructions you'd expect, but also a few puzzlers such as add 6 to A, add 8 to A and add 10 to A, which were probably used for binary to decimal conversion. It had a single status flag, which could indicate carry, zero, or inequality, depending on the situation. It supported bit test, set, and reset, as well as ROM-programmed constants, but didn't have any Boolean operations.

If you didn't like the instruction set of the TMS1000, it could be changed during manufacture by reprogramming the internal instruction PLA, which mapped each instruction to 16 microinstructions. The programmer's guide provides details:
http://bitsavers.trailing-edge.com/pdf/ti/TMS1000/TMS1000pgmRef_1975.pdf

The PC of the TMS1000 doesn't increment one at a time like a normal program counter. Incredible as it seems, the program counter moves through an apparently-random sequence generated by a shift register with feedback. The program instructions in the ROM are ordered in the same sequence during manufacture, so everything works out, even though the program is entirely shuffled. This shortcut saved a few gates that would have been required for the program counter incrementer.

The point of this is that calculator chips typically had a design very focused on calculator operation. Even a “general purpose” 4-bit microcontroller like the TMS1000 was designed around calculator operations and support for calculator hardware.

The instruction set of the 4004 is very different from a calculator instruction set and much closer to the instruction set of a general-purpose computer. In particular, the 4004 was designed for binary operations. It has a general-

purpose set of 16 registers, rather than registers that hold decimal numbers. (Of course, the 4-bit registers on the 4004 were conveniently sized for holding a digit.) The 4004 operated on one digit at a time. It had bitwise rotate instructions. Some instructions that are notably missing are Boolean operations such as AND, OR, and XOR.

Two instructions show the calculator roots of the 4004. DAA (decimal adjust accumulator) converts a binary value to a decimal value after addition. KBP (keyboard process) converts a one-of-four code to a binary code (i.e. when one keyboard input line of four is set, that gets converted to 2 binary bits). TCS (transfer carry subtract) loads 9 into the accumulator if carry is clear and 10 if carry is set.

One feature of the 4004 to note is that memory accesses were done by using two registers to form an address. This reappears in the 8008 in the use of the HL register. Memory accesses in the 4004 are interesting, since one instruction sends the address to RAM, and then a second instruction sends the data. In addition, the 4004 has separate address spaces for ROM and RAM, with separate instructions for each.

The 4004 uses an internal pushdown stack for subroutine calls, a feature that now seems obsolete, but was used in the 8008.

The I/O operations of the 4004 are rather bizarre, since they are designed specifically for the I/O ports built into the RAM and ROM chips.

The 4004 microprocessor was designed for Busicom, a Japanese calculator company. Busicom got its start as The Nippon Calculating Machine Corporation in 1945, making mechanical calculators. In 1965, they started making electronic desktop calculators; the Unicom 160 was crammed full of boards of diode-transistor logic. As technology developed, they moved to integrated circuits. (Reference http://www.vintagecalculators.com/html/busicom_141-pf_and_intel_4004.html)

In 1968 Masatoshi Shima was a young engineer at Busicom, who was put to work on the design of Busicom's first electronic printing calculator. His supervisor Tadashi Tanba had worked on computers at Control Data and suggested a flexible design that would be programmable in software. This would allow the calculator to be reprogrammed with new features without requiring a new hardware design. New calculators could be produced just by changing the ROM chips.

Shima came up with a complex design with hardware arithmetic units such as adders and a multiplier, controlled through a macro-instruction set. Tadashi Sasaki, an executive at Sharp Corporation, helped set up a deal between Intel and Busicom for Intel to build the necessary integrated circuits. Busicom would have an exclusive license to the chips as part of this deal.

After a year, Intel hadn't made much progress on the design. One problem was the complexity of Shima's design was too much for Intel's small design team. Another problem was Intel was much more interested in memory chips than calculators.

The leader of the Intel project was Ted Hoff, who was working with Stanley Mazor. Hoff was familiar with the PDP-8 computer. The PDP-8 had only 8 basic instructions, rather than a instruction set with many complex instructions, showing that a successful computer could have a small instruction set. (The PDP-8 could be considered an early RISC computer. Hoff's support for RISC is interesting given Intel's future trajectory towards complex CISC processors.) Hoff suggested greatly simplifying the system and using a simple instruction set. In October 1969, Busicom decided to go with the simpler design.

Shima returned to Intel in April 1970 to see how the design was progressing, but discovered that nothing had happened. Hoff had moved to another project, designing a CPU for an intelligent terminal, a CPU that would become Intel's 8008.

At this point, the paths of Shima and Faggin met up. Faggin had been hired to

complete the Busicom design.

The Busicom PF 141-PF put its output on a 15-digit narrow roll of paper. The print mechanism was build by Shinshu Seiki. Shinshu Seiki Co started producing printers for timekeeping at the 1964 Olympics in Tokyo. From this they produced the first electronic miniprinter, the EP-101. The followup to this printer, the “son of EP” led them to change the company name to Epson, the well-known printer company. (wikipedia)

Following the 4004, Busicom used the Mostek single-chip MK6010 in the “Handy”, which was the first hand-held calculator, with a LED display. Interestingly this calculator used a 12-digit display, considerably larger than the 8-digit displays common on pocket calculators now.

To get an idea of how primitive calculators were at the time, an article on calculators in Popular Science, June 1971 describes the Brother 310 with a list price of $349 (over $2000 in 2014 dollars). This calculator only had add, subtract, and multiply. To divide, you looked up the divisor on a provided list, entered the reciprocal manually, and then multiplied by that value.

Busicom ended ceasing production in 1974 due to financial difficulties. The Busicom name was bought out and is still used on calculators.

Reference:
http://archive.computerhistory.org/resources/text/Oral_History/Faggin_Federico/Faggin_Federico_1_2_3.oral_history.2004.102658025.pdf

See also http://www.ieeeghn.org/wiki/index.php/Oral-History:Federico_Faggin

Faggin didn't know until he arrived at Intel that he would be working on the 4004, having just been told that there was a challenging project awaiting him. His first day on the project, Faggin discovered that the project was not as far along in development as he had been led to believe. He also discovered that Shima was arriving from Japan the next day to check on progress.

Faggin used his one day to ramp up on the project and read through the documentation. He found a block diagram with the basic organization of the system. The system was built of four chips. First was the 4004 CPU itself. Next was a customized ROM chip; this would be the 4001. The RAM chip was the 4002. Finally was a simple 10-bit serial-in, parallel-out shift register chip that would extend the output capabilities of the system; this would be the 4003. (This chipset was known as the MCS-4, for MicroComputerSystem 4-bit.)

These chips were all connected on a 4-bit bus. The CPU generated a SYNC signal at the start of each instruction. The CPU lacked the multiplier of Shima's original design; it's arithmetic was cut down to a 4-bit adder, capable of adding a single decimal digit. The instruction set was larger than the PDP-8's; the 4004 ended up with 46 instructions compared to the PDP-8's eight instructions. However, the PDP-8 really had considerably more instructions than this in practice. The OPR (operate) instruction could perform many different operations based on how the bits were set in the operation, yielding more than 30 additional operations.)

One unusual feature of this architecture is the I/O functions were built into the RAM and ROM chips. The 4001 ROM included a 4-bit I/O port, while the 4002 RAM included a 4-bit output port. The organization of storage in the RAM is also unusual: it is organized as four registers, each containing 16 4-bit main memory characters, along with 4 4-bit status characters, for a total storage of 1280 bits per chip.

As for actual circuitry, Faggin found some basic test circuits, but quickly realized they wouldn't actually work. There wasn't any logic diagram. Intel had made more progress on the RAM and ROM chips, based on Intel's expertise in memory. The chips themselves weren't designed, but at least there were designs for the individual memory cells. Intel had a three-transistor cell for the dynamic RAM, and a serial ROM cell, which worked on slow chips like the 4001.

The next day, Mazor and Faggin picked up Shima at the airport. Shima had

been promised that logic was done, so he was shocked to see how little progress had been done. Mazor excused himself, leaving Faggin to face the anger of the disappointed Shima. Shima started complaining, “This is not what I came to do. This is idea. This is not design. I came here to check, and there is nothing to check! You bad! You were supposed to do!” Fortunately Faggin realized that Shima's English was poor and Shima was blaming Intel, not Faggin personally.

It's unclear why Intel failed to progress on the project. It was probably (according to Faggin) a combination of Intel's interest in memories rather than processors, negligence, and lack of people to work on the project. At the time, Intel had probably 35 or 40 designers and support people for both MOS and bipolar chips. In addition, the schedules were completely unrealistic, due to a lack of experience with complex chips. For example, the CPU layout was scheduled for one month. This was reasonable for a ROM or RAM chip, which was familiar and mostly consisted of a grid of repeated memory cells. But layout of a CPU with complex logic would take much longer.

After Shima reported back to Busicom, the whole project was at risk due to the lack of progress on the schedule. Faggin convinced Shima that if he helped, they could get the project done faster. Shima received permission from Busicom to stay at Intel and help out.

They started with the layout of the 4001 ROM chip, since that was the simplest. Once that was done, they moved on to the 4003 shift register and then the 4002 RAM chip. Finally, they moved to the 4004 CPU itself.

But before they could do the layout, they had to develop was the basic design rules of how to build the circuits. Since gate logic hadn't been done with silicon gates before, this was a new area. They had to figure out what rules would allow them to build logic in a power-efficient, area-efficient, and time efficient manner.

There were several alternatives for designing “random” logic. Four-phase dynamic logic was popular at Fairchild. This alternative wouldn't have fit on a single CPU, though. A static design, based on flip flops instead of dynamic

logic would also have resulted in an excessively large chip. The approach he selected was a two-phase clock with dynamic logic, but this required the bootstrap loads discussed earlier.

At Fairchild, random logic had used a four-phase clock for efficiency. This type of logic was used the late 1960s and early 1970s for efficiently implementing complex logic. http://en.wikipedia.org/wiki/Four-phase_logic The logic was implemented as dynamic logic. Interestingly, gates were not connected to power or ground, but only to the clock. The gates were dynamic, working off the capacitance of the gates, so they were very different in design from the MOS gates used in the Z80, 4004, and other processors.

The third important design concept was the buried contact, which effectively allowed two layers of interconnection (poly and metal).

Once he had the circuit designs, Faggin could simulate them. Unfortunately, simulation was very costly. Intel, in fact didn't have their own computers for simulation, but connected to a timesharing system via teletypes. As a result, Faggin didn't do any simulation on the first three chips, and just a bit on the 4004 itself. Instead, he had graphs of transistor characteristics and had developed some rules. With these, he could figure out the sizing of the transistors with a slide rule.

The design of the 4001 ROM took him about a week. Once he had the design, it was time for the layout of the chip. Unfortunately his draftsman, Rob Sayre, didn't have any experience with MOS, or even electronics, so there was a lot of learning involved. Faggin would do the planning of the chip, the high-level layout, and the structure of the cells, and then the draftsman would do the final drawing. Some of the design rules Faggin developed were how to estimate the size of a cell, to avoid finding it didn't fit when the layout happened. Other design rules were how to draw the schematics to be closer to the final layout. The layout of the 4001 took about a month. From layout, it went into a Rubylith cutting. A large sheet of Mylar had a red layer that was cut and peeled off to form a layer of the mask. The mask was made at 400 or 500x scale, and then reduced by a camera to 10x.

Finally, the step-and-repeat camera had 1x masks that made the master masks for production.

Faggin points out that this was a manual process, prone to errors. Because there was no automation, even a repeated cell needed to be drawn manually. For instance, the shift register had 10 stages that were almost identical, but they all needed to be done manually.

Cutting the Rubylith masks took three or four weeks, followed by a few more weeks to make the masks. The fab took about three weeks for a fast run to produce the chips. When they finally received the 4001, it worked for the first time, which helped prove their methodology was valid.

While layout of the 4001 took place, Faggin designed the 4003 shift register. He used a special static master-slave flip flop that he and his boss at SGS-Fairchild had co-invented. This chip took about two weeks to lay out because of its simplicity. See the layout at http://www.4004.com/mcs4-masks-schematics-sim.html (I should figure out how the flip flop works.)

When the 4003 came back from the fab, it too worked perfectly.

Following that, Faggin designed the 4002 RAM. As soon as layout of the 4003 finished, layout started on the 4002.

Layout of the 4004 started around May 1970 and took about 3 1/2 months with two or three draftsmen. By this time, Julie Hendrix had joined from Fairchild as a drafter; she had experience with MOS and silicon gate technology, so her expertise was helpful.

For the 4004, Faggin did the logic design and circuit design together. After designing key components, Faggin passed the layout on to Shima, who completed the design while Faggin was doing layout. Once the 4004 layout was complete and moved on to Rubylith cutting, Shima returned to Japan to develop the firmware.

The only computer simulation they did was on the 4-bit internal bus. Not

only was the bus used for communication between internal registers and communication with the external bus, but the internal was was also used as temporary memory - with dynamic logic, it could hold internal state. The relatively expensive simulation was done on this to make sure all the timing was correct.

Due to the chip yield curves, the chips had to be less that 140 mils large to be cost-effective. In addition, they wanted an inexpensive 16-pin package (compared to 40 pins for processors like the Z80, or 1000-2000 pins for modern Intel processors).

In November, the 4002 came back from the fab, working except for one minor mistake. Finally in December the 4004 wafers came back from fab. Faggin probed them, on the die, not even diced. The first die didn't work at all. The second die was also totally dead. He moved to a different wafer, which was also dead. None of the chips worked at all. Could his design be wrong? The first two chips had worked, so the methodology couldn't be totally flawed. He checked the wafer under the microscope and discovered that the fab process had entirely omitted the mask for the buried contact layer, ruining the chips.

This simple manufacturing error cost three weeks in the schedule, waiting for a new batch of chips to be fabbed. Mid-January the new wafers came back, Faggin probed a die, and found it worked. He ran test programs to exercise various parts of the chips, and, one by one all the tests passed. Finally, at 3:30am everything seemed to be working in the chip, and he went home.

A few minor bugs turned up in the 4004 after more extensive testing, but they were easy to correct.

Meanwhile, Shima had written the software for the calculator and had it ready to commit to ROMs. He had built a simulator out of discrete logic to simulate the 4004 so he could test the software.

By early 1971, the design of the 4004 chipset was completed and chipsets were sent to Busicom. Shima had a working calculator by March, and it went

into production the next month as the Busicom 141-PE, also sold as the NCR 18-36.

Unfortunately for Busicom, prices of calculators steadily fell, leading to financial difficulties for Busicom. To save costs, Busicom renegotiated their deal with Intel. In exchange for a reduced price on the chips, Busicom gate up their exclusive rights to the chipset, opening the door for Intel to sell the 4004 chipset.

Intel didn't have much enthusiasm for selling processors, but fortunately for the computer industry, they hired a marketing director Ed Gelbach, who had worked at Texas Instruments where there was much more enthusiasm for processors. The market was enthusiastic for the 4004. This led Intel to put more emphasis on microprocessors, even though they didn't make a lot of money off the 4004. Intel continued microprocessor development with the 8008, 8080, 8085, and the x86 series. However, it wasn't until 1983 that Andy Grove decided that microprocessors should be the focus of Intel.

The 8008

Reference: **Intel Microprocessors: 8008 to 8086**

Stephen P. Morse / Bruce W Ravenel / Stanley Mazor / William B. Pohlman

Originally published in IEEE Computer, Vol 13, No. 10, pages 42-60, October 1980

The 8008 processor, like the 4004 processor, started with an outside company that wanted a processor. And like the 4004, Intel dragged their feet on it. And like the 4004, the outside company ended up giving Intel back the rights to the chip.

In 1969, the Datapoint corporation (then called Computer Terminal Corporation) wanted a process for a “smart” CRT terminal. Their original plan was a bit-serial processor built out of TTL with shift-register memory. (A bit-serial processor processes one bit at a time, in sequence.) Intel, however, suggested a PMOS microprocessor on one chip.

The Datapoint 2200 processor architecture is fairly straightforward. It has 8-bit registers: an accumulator (A) and general-purpose registers (B, C, D, E).

It also has H and L registers that are combined to form an address that is used to reference memory. All memory accesses take place using the HL register pair. Otherwise, these can be used as general purpose registers.

The system had four flags: carry, sign, zero, and parity. Unlike most processors that put these flags into a byte, the Datapoint flags existed independently, on their own, with no bit positions.

The instruction set was straightforward and relatively orthogonal.

It supported a solid set of ALU operations: add, add with carry, subtract, subtract with borrow, and, or, exclusive or, and compare. The load operation also provided data moves. These instructions could act on any of the 7 registers or memory (which was treated as an eighth register in the opcodes). These instructions also supported immediate arguments.

Shift left and shift right were provided but only worked on the accumulator.

Control flow consisted of absolute jumps, subroutine calls, and returns, either unconditional or conditional.

The processor also has multiple I/O instructions (EX for external), which control various parts of the terminal such as the keyboard or tape drive.

Unlike modern processors, the Datapoint 2200's pushdown stack was part of the CPU, which limited its size to 16 entries. The stack was used for subroutine calls and returns. In addition the HL registers could be pushed or popped, but only in the Version II extended instruction set. Prior to that, registers couldn't be pushed in interrupt handlers, which made interrupt handling difficult.

The memory in the Datapoint 2200 was interesting: it was implemented through MOS shift registers. Each bank of 2K consisted of 32 shift registers of 512 bits each. Accessing memory addresses in order was quick, but accessing a random location could require up to 1/2 millisecond for the desired byte to circulate through the shift register.

One thing to mention is how modern the Datapoint 2200 processor is. It's no Pentium, of course. But for an 8-bit processor, it has a reasonably comprehensive instruction set, a standard register set, and few things that stick out as archaic. (The hardware stack is probably the main issue.)

Texas Instruments filed a very important patent on a CPU chip in 1971, getting patent http://www.google.com/patents/US3757306. This patent is extremely detailed, like many TI patents, with a full schematic of the chip and sample source code. It covers a CPU on a chip with four parts: and ALU, memory registers, control, and an interconnect bus.

The interesting thing about the Texas Instruments patent is it exactly describes the Datapoint 2200 architecture. The registers and flags are identical. The instructions are identical, right down to the opcodes. The only opcode difference is the eight RST instructions. Since these also appear in the 8008 in exactly the same spot, I speculate that they were in the original Datapoint 2200 plan, so Intel and Texas Instruments implemented them, but Datapoint ended up dropping them.

Is 8080 based on Faggin or on Datapoint 2200? Reference: **Datapoint: The Lost Story of the Texans Who Invented the Personal Computer**
The 8008 is almost identical to the Datapoint 2200 as far as architecture and instruction set. The register set is identical (except the 8008 has a smaller internal stack). The opcodes are also identical, with very few differences:
The 8008 adds opcodes to increment or decrement B, C, D, E, H, or L (but not memory or accumulator).
The 8008 adds RST instructions.
The 8008 adds RAL and RAR, which rotate A left or right through the carry flag.

The impact of the Datapoint 2200 on the 8080 and Z80 cannot be overstated. The instruction set of the 8080 is almost exactly an extension of the Datapoint 2200. The register set of the 8080 is identical to the Datapoint 2200. (The 2200's hardware stack was became a 16-bit stack pointer to memory.) The 8080 has the same flags as the 2200, with the addition of a

half-carry flag and making the interrupt flag explicit. Unusual instructions such as RST had their origin in the Datapoint 2200.

You might think that these similarities are to be expected with 8-bit processors. But if you look at other processors such as the 6800 or 6502, not to mention the SC/MP, you'll see that the register sets, instruction sets, addressing modes, and flags can be very different.

The instruction set of the 8080 is a superset of the Datapoint 2200. This explains the presence of instructions such as return on parity even, which don't seem particularly useful. One difference in the instruction sets is the 8080 combined the numerous input/output instructions into one input and one output instruction.

One obscure feature that carried over to the 8080 and the Z80 is that the LD instructions form a block of 64 general opcodes, with one exception. The instruction that would be a memory-to-memory move - LD (HL), (HL) - is instead the halt instruction. Note that this instruction would copy a memory location to itself, so it would be pointless. Removing it doesn't cause any inconvenience. However, the pointless copy-register-to-self instructions such as LD B, B are left as is. The Datapoint used LD A, A as the NOP instruction, while the 8080 added an explicit NOP instruction for timing reasons.

The TTL implementation of the Datapoint 2200 processor is shown in schematics at
http://bitsavers.informatik.uni-stuttgart.de/pdf/datapoint/2200/2200_Drawing_Package.pdf
This shows the Version II processor from 1973 which adds a few more instructions, two register banks (a feature that, interestingly, the Z80 used a few years later), and the 8-bit CPU.

To give an overview of the processor schematics and some of the key features:
Page 33 shows the CPU timing logic with cycles and T states which end up being somewhat similar to the Z80. The states are handled by 7495 shift

registers, which shift the active bit from position to position.

Page 34 shows the instruction decoding logic. The instruction is latched in two 7475 4-bit latches. A 74157 multiplexer selects bits 210 or 543 to select the register, as appropriate. A 7442 decimal decoder decodes the 3 bits (543) selecting an ALU operation. Another 7442 decimal decoder decodes 00nnn000 (push, pop, etc), with a 7474 to hold the interrupt enable state. Two other 7442 decimal decoders handle bits 210 and bits 76. A bunch of gates provide random logic for other decoding.

Page 35 combines timing with instruction decoding to generate control lines.

Page 36 shows the ALU itself. It is built around two 74181 4-bit ALU chips. It also has two 7489 64-bit RAM chips that provide the two banks of 8 registers. This page also has the flag logic: a 74180 parity generation chip, 8 flip flops for two banks of four flags, a 74153 multiplexer to select a flag, and random logic for various purposes including selecting the appropriate value for the carry flag. Although the Datapoint 2200 is often referred to as having a bit-serial ALU, in the version II here is is a full 8-bit ALU.

Page 37 shows the stack, built out chips including of 7489 64-bit RAM chips and a 74193 counter chip.

When Faggin joined the 8008 project, it was called the 1201: 8008 was the later marketing name. I'll refer to the project as the 8008 for simplicity. The architecture came from Datapoint (which was Computer Terminal Corporation at the time). They had a TLL design and wanted it converted to a MOS chip. The Datapoint 2200 programmable terminal was really a computer, but was called a “programmable terminal” to avoid scaring investors. It was announced in June 1970 and shipped in 1971. Shipped to General Mills May 25, 1970?

To understand the 1201 name (ref: Feeney oral history), 1 means p-channel MOS process. 2 means custom device. Finally 01 is a sequence number (later

used as size?). For the second digit, maybe 1 means RAM, 3 means ROM. 4 means shift register, 7 means EPROM. 2914 communication. 2102 static ram, 2401 shift register. Schottky bipolar: 3xxx. NMOS sram: 81xx. CMOS: 5xxx (see http://www.antiquetech.com/?page_id=804). 7xxx bubble memory. EPROM: 17xx, 27xx, 87xx.

The Datapoint could be programmed in DATABUS (also known as Programming Language for Business or PL/B was developed by Datapoint as a COBOL alternative. It used bytecode, interestingly), SCRIBE, BASIC, or assembly language.

What's the timeline? a) When did the 8008 start? Wikipedia says the contract was early 1970. b) When was the TTL Datapoint 2200 built? Was the TTL circuit first? Datapoint was announced in 1970.

The 8008 project was being run by Hal Feeney, with Ted Hoff and Stan Mazor on the project.

The legend, according to Faggin, is that the root of the 8008 project was around October 1969. The VP of Engineering at Datapoint approached Intel about getting a custom bipolar chip to manage the stack - a 64 bit memory with an integrated stack pointer, so it could act as a last-in first-out stack. When Hoff and Mazor heard what Datapoint had planned, they realized that Intel could go way beyond just a stack chip and said, “We could do everything in one chip.” At that point, Intel and Datapoint made a deal and Feeney was hired to lead the project, starting around 1970.

The standard story is that Datapoint approached both Intel and Texas Instruments to build a MOS version of the Datapoint 2200 because the TTL version took up too much space and generated too much heat.

When Faggin joined Intel, he figured that the 8008 would be done before his chip, the 4004, since it had a head start and a full design. Of course, that turned out not to be the case. Hal Feeney was pulled off the 8008 to work on a new memory chip and the 8008 project started to languish around Apil 1970. Meanwhile, Texas Instruments ran into reliability problems with their

chip, although they did get a hugely important patent out of it. Texas instruments announced their CPU-on-a-chip sometime between March and May 1971, although it never became a product. Unfortunately since the 4004 was a custom design, it wasn't publicized immediately, although there were articles such as Standard Parts and Custom Design Merge in Four-Chip Processor Kit, Electronics, April 24, 1972. Also THE MCS-4-AN LSI MICRO COMPUTER SYSTEM
by F. FAGGIN, M. SHIMA*, M.E. HOFF, H. FEENEY, S. MAZOR, IEEE 1972 Region Six Conference.

Datapoint proceeded with the TTL version.

In January 1971, Seiko asked about using the 8008 for a desktop scientific calculator similar to the HP 9100. This lead the 8008 project to be restarted under Faggin's direction. At this point, the logic for the control had been done, but none of the actual circuit design. Faggin gave guidance to Feeney on key circuits, organization, design techniques and other methodology. Feeney did most of the 8008 work himself.

Faggin had to track down one problem with the 8008's memory which was “losing its memory”. This turned out to be similar to a problem he encountered on the 4002 and 4004. Fixing the problem was straightforward but politically controversial. The substrate required a bias, which required moving from a 16-pin chip to an 18-pin chip, which Intel hated. At the time, using 16 pins was “a religion in Intel”.

Faggin viewed this as completely silly, especially since 40 and 48 pin chips were in common use. Cramming everything onto 16 pins meant losing functionality, or making the chip slower and harder to use by multiplexing multiple functions onto single pins. (By today's standards, the 4004 chips look surprisingly small in their 16-pin packages. This explains the motivation behind the packages.) The 1103 needed a pin for Vbb, the bias voltage. (It's unclear to me which other pin is the “additional” pin, maybe PRECHARGE.) A full schematic of the 1103 is at
http://maben.homeip.net/static/S100/intel/datasheet/197308%20Intel%20Memory%20Design%20Handbook.pdf

Faggin views cramming the 8008 into an 18-pin package as one of the factors that kept it from being more popular, as the external bus connections involved a lot of compromise. This required demultiplexing buses with latches and other logic. For example, the address bus and data bus used the same pins. Unlike later chips, the 8008 didn't have control lines to indicate memory reads or writes. Instead, it used three state outputs, which needed to be decoded to determine what the processor was doing. In addition, two of the data bus lines were also used to distinguish a memory fetch, memory read, memory write, or I/O command. The result is that the 8008 required extra external chips to handle these control signals and multiplexed buses. A minimal system required 15 to 20 chips, compared to the 4004 which could work with just the processor and a memory chip.

The 8080, in comparison, used a 40-pin package, as did the other popular 8-bit processors of that time (6800, 6809, 6502, Z80) Even the 8088 and 8086 used 40-pin packages.

Hal Feeney describes some of the challenges with the 8008 in his oral history (http://archive.computerhistory.org/resources/access/text/2012/07/102658338-05-01-acc.pdf). During meetings in 1969 and early 1970, CTC and Intel representatives agreed on a specification. CTC was building out of TTL, which was relatively expensive, so they hoped that MOS would reduce their costs, both for the processor and for the system as a whole. Ted Hoff and Stan Mazor made a specification that described the instructions, and how many machine cycles each instruction would take. This also described registers and interrupt handling. Feeney's job after he joined in March 1970 was to turn this into a transistor-level design, figuring out how the control signals would work, adding any necessary intermediate registers, and doing the layout.

By the summer of 1970, Feeney was pulled off the 8008 for the most part. The industry was in a cyclic downturn at the time, causing priorities to change. Feeney claims that CTC was “backing away” from the chip. In addition, Intel wanted to focus on different priorities. Feeney ended up spending most of his time on the 1201 and 1101 RAMs. He also helped design the test equipment for the 4004. In addition, as the price of TTL fell

drastically, the bene

The 16-pin package wasn't enough for all the necessary control lines, even with the multiplexing they did. Feeney needed one more line for interrupt handling, so the two extra pins from an 18-pin package made a big difference.

One challenge was putting logic and dynamic memory on the same chip. The 8008 used dynamic memory for its registers. They also needed to add voltage generators to the chip, and were able to use Faggin's solution for the 4004.

Interconnections in the chip were much more challenging than the regular interconnections of memory chips, due to the random logic in the processor. Buried contacts made a big difference with this.

One timing problem was decoding the instruction fast enough to set up all the data paths. The ALU didn't have carry-propagation timing problems because they used carry look-ahead.

A final challenge was the layout- fitting everything into the space available. Two layout designers worked on the 8008, which was the last chip Intel did without any computerized design tools. Like the 4004, it was drawn at 500 times actual size, followed by cutting and peeling Rubyliths by hand to make the masks. The 8008 was too big for the Rubylith width available, so they had to make the masks in two halves and then paste them together on a table the size of a ping-pong table. Since there were 5 layers, all 10 pieces had to be perfectly aligned by hand.