Architecture 2

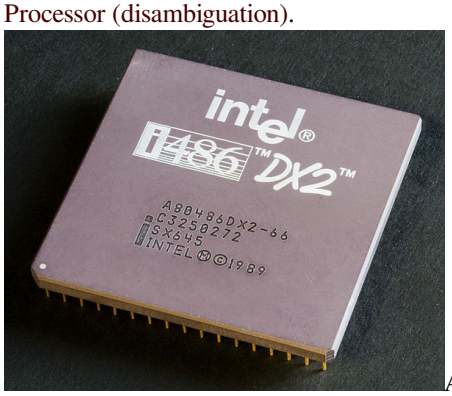
CPU, DSP, GPU, NPU

Contents

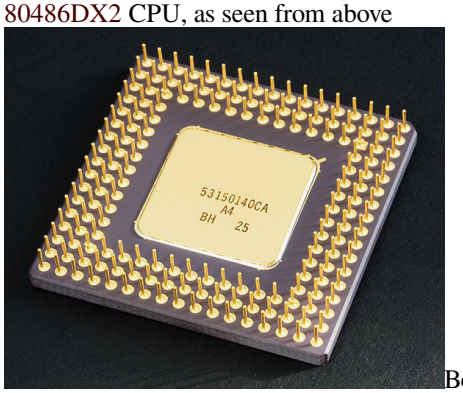
Chapter 1

Central processing unit

"CPU" redirects here. For other uses, see CPU (disambiguation).
"Computer processor" redirects here. For other uses, see Processor (disambiguation).



An Intel 80486DX2 CPU, as seen from above



Bottom side of an Intel 80486DX2

A **central processing unit** (**CPU**) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions. The term has been used in the computer industry at least since the early 1960s.[1] Traditionally, the term "CPU" refers to a processor, more specifically to its processing unit and control unit (CU), distinguishing these core elements of a computer from external components such as main memory and I/O circuitry.[2]

The form, design and implementation of CPUs have changed over the course of their history, but their fundamental operation remains almost unchanged. Principal components of a CPU include the arithmetic logic unit (ALU) that performs arithmetic and logic operations, processor registers that supply operands to the ALU and store the results of ALU operations, and a control unit that fetches instructions from memory and "executes" them by directing the coordinated operations of the ALU, registers and other components.

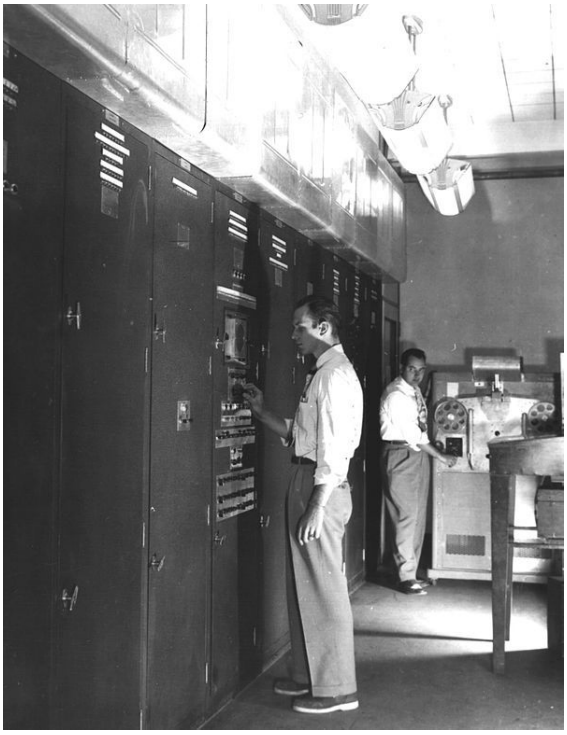
Most modern CPUs are microprocessors, meaning they are contained on a single integrated circuit (IC) chip. An IC that contains a CPU may also contain memory, peripheral interfaces, and other components of a computer; such integrated devices are variously called microcontrollers or systems on a chip (SoC). Some computers employ a multi-core processor, which is a single chip containing two or more CPUs called "cores"; in that context, single chips are sometimes referred to as "sockets".[3] Array processors or vector processors have multiple processors that operate in parallel, with no unit considered central.

1.1 History

Main article: History of general-purpose CPUs

Computers such as the ENIAC had to be physically rewired to perform different tasks, which caused these machines to be called "fixed-program computers".[4] Since the term "CPU" is generally defined as a device for software (computer program) execution, the earliest devices that could rightly be called CPUs came with the advent of the stored-program computer.

The idea of a stored-program computer was already present in the design of J. Presper Eckert and John William Mauchly's ENIAC, but was initially omitted so that it could be finished sooner.[5] On June 30, 1945, before ENIAC was made, mathematician John von Neumann distributed the paper entitled *First Draft of a Report on the EDVAC*. It was the outline of a stored-program computer that would eventually be completed in August 1949.[6] EDVAC was designed to perform a certain number of instructions (or operations) of various types. Significantly, the programs written for EDVAC were to be stored in high-speed computer memory rather than spec-



EDVAC, one of the first stored-program computers

ified by the physical wiring of the computer.[7] This overcame a severe limitation of ENIAC, which was the considerable time and effort required to reconfigure the computer to perform a new task. With von Neumann's design, the program that EDVAC ran could be changed simply by changing the contents of the memory. EDVAC, however, was not the first stored-program computer; the Manchester Small-Scale Experimental Machine, a small prototype stored-program computer, ran its first program on 21 June 1948[8] and the Manchester Mark 1 ran its first program during the night of 16–17 June 1949.[9]

Early CPUs were custom designs used as part of a larger and sometimes distinctive computer.[10] However, this method of designing custom CPUs for a particular application has largely given way to the development of multi-purpose processors produced in large quantities. This standardization began in the era of discrete transistor mainframes and minicomputers and has rapidly accelerated with the popularization of the integrated circuit (IC). The IC has allowed increasingly complex CPUs to be designed and manufactured to tolerances on the order of nanometers.[11] Both the miniaturization and standardization of CPUs have increased the presence of digital devices in modern life far beyond the limited application of dedicated computing machines. Modern microprocessors appear in electronic devices ranging from automobiles[12] to cellphones,[13] and sometimes even in toys.[14]

While von Neumann is most often credited with the design of the stored-program computer because of his design of EDVAC, and the design became known as the von Neumann architecture, others before him, such as Konrad Zuse, had suggested and implemented similar ideas.[15] The so-called Harvard architecture of the Harvard Mark I, which was completed before EDVAC,[16][17] also utilized a stored-program design using punched paper tape rather than electronic memory.[18] The key difference between the von Neumann and Harvard architectures is that the latter separates the storage and treatment of CPU instructions and data, while the former uses the same memory space for both.[19] Most modern CPUs are primarily von Neumann in design, but CPUs with the Harvard architecture are seen as well, especially in embedded applications; for instance, the Atmel AVR microcontrollers are Harvard architecture processors.[20]

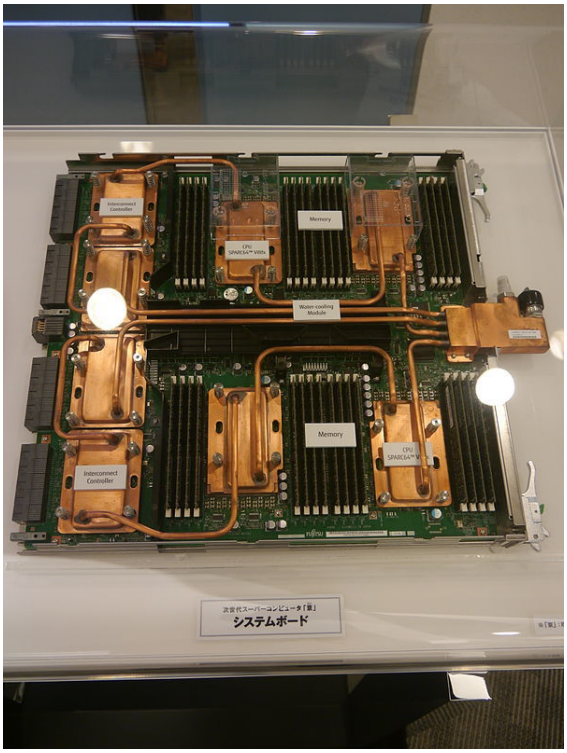
Relays and vacuum tubes (thermionic tubes) were commonly used as switching elements;[21][22] a useful computer requires thousands or tens of thousands of switching devices. The overall speed of a system is dependent on the speed of the switches. Tube computers like EDVAC tended to average eight hours between failures, whereas relay computers like the (slower, but earlier) Harvard Mark I failed very rarely.[1] In the end, tube-based CPUs became dominant because the significant speed advantages afforded generally outweighed the reliability problems. Most of these early synchronous CPUs ran at low clock rates compared to modern microelectronic designs. Clock signal frequencies ranging from 100 kHz to 4 MHz were very common at this time, limited largely by the speed of the switching devices they were built with.

1.1.1 Transistor CPUs

The design complexity of CPUs increased as various technologies facilitated building smaller and more reliable electronic devices. The first such improvement came with the advent of the transistor. Transistorized CPUs

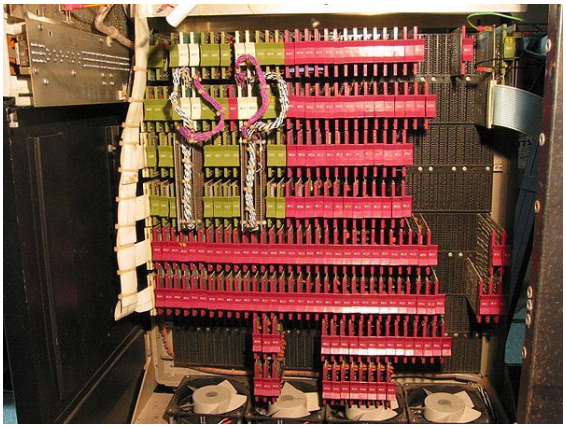
during the 1950s and 1960s no longer had to be built out of bulky, unreliable, and fragile switching elements like vacuum tubes and relays. With this improvement more complex and reliable CPUs were built onto one or several printed circuit boards containing discrete (individual) components.

In 1964, IBM introduced its System/360 computer architecture that was used in a series of computers capable of running the same programs with different speed and performance.[23] This was significant at a time when most electronic computers were incompatible with one another, even those made by the same manufacturer. To facilitate this improvement, IBM utilized the concept of a microprogram (often called "microcode"), which still sees widespread usage in modern CPUs.[24] The System/360 architecture was so popular that it dominated the mainframe computer market for decades and left a legacy that is still continued by similar modern computers like the IBM zSeries.[25][26] In 1965, Digital Equipment Corporation (DEC) introduced another influential computer aimed at the scientific and research markets, the PDP-8.[27]



Transistor-based computers had several distinct advantages over their predecessors. Aside from facilitating increased reliability and lower power consumption, transistors also allowed CPUs to operate at much higher speeds because of the short switching time of a transistor in comparison to a tube or relay.[28] Thanks to both the increased reliability as well as the dramatically increased speed of the switching elements (which were almost exclusively transistors by this time), CPU clock rates in the tens of megahertz were obtained during this period. Additionally while discrete transistor and IC CPUs were in heavy usage, new high-performance designs like SIMD (Single Instruction Multiple Data) vector processors began to appear.[29] These early experimental designs later gave rise to the era of specialized supercomputers like those made by Cray Inc and Fujitsu Ltd.

1.1.2 Small-scale integration CPUs



CPU, core memory, and external bus interface of a DEC PDP-8/I. Made of medium-scale integrated circuits.

During this period, a method of manufacturing many interconnected transistors in a compact space was developed. The integrated circuit (IC) allowed a large number of transistors to be manufactured on a single semiconductor-based die, or "chip". At first only very basic non-specialized digital circuits such as NOR gates were miniaturized into ICs. CPUs based upon these "building block" ICs are generally referred to as "small-scale integration" (SSI) devices. SSI ICs, such as the ones used in the Apollo guidance computer, usually contained up to a few score transistors. To build an entire CPU out of SSI ICs required thousands of individual chips, but still consumed much less space and power than earlier discrete transistor designs.

IBM's System/370 follow-on to the System/360 used SSI ICs rather than Solid Logic Technology discrete-transistor modules. DEC's PDP-8/I and KI10 PDP-10 also switched from the individual transistors used by the PDP-8 and PDP-10 to SSI ICs, and their extremely popular PDP-11 line was originally built with SSI ICs but was eventually implemented with LSI components once these became practical.

1.1.3 Large-scale integration CPUs

Lee Boysel published influential articles, including a 1967 "manifesto", which described how to build the equivalent of a 32-bit mainframe computer from a relatively small number of large-scale integration circuits (LSI).[30][31] At the time, the only way to build LSI chips, which are chips

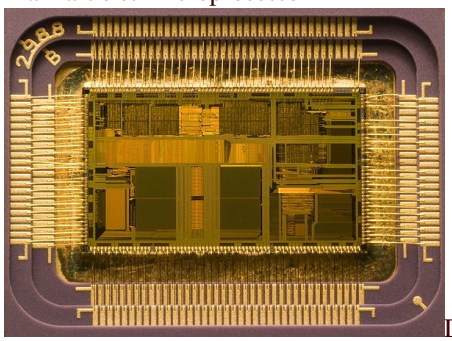
with a hundred or more gates, was to build them using a MOS process (i.e., PMOS logic, NMOS logic, or CMOS logic). However, some companies continued to build processors out of bipolar chips because bipolar junction transistors were so much faster than MOS chips; for example, Datapoint built processors out of TTL chips until the early 1980s.[31]

People building high-speed computers wanted them to be fast, so in the 1970s they built the CPUs from small-scale integration (SSI) and medium-scale integration (MSI) 7400 series TTL gates. At the time, MOS ICs were so slow that they were considered useful only in a few niche applications that required low power.[32][33]

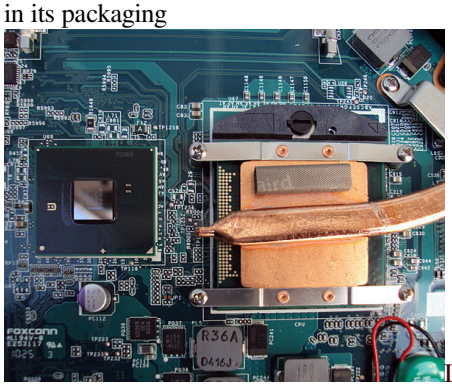
As the microelectronic technology advanced, an increasing number of transistors were placed on ICs, decreasing the quantity of individual ICs needed for a complete CPU. MSI and LSI ICs increased transistor counts to hundreds, and then thousands. By 1968, the number of ICs required to build a complete CPU had been reduced to 24 ICs of eight different types, with each IC containing roughly 1000 MOSFETs.[34] In stark contrast with its SSI and MSI predecessors, the first LSI implementation of the PDP-11 contained a CPU composed of only four LSI integrated circuits.[35]

1.1.4 Microprocessors

Main article: Microprocessor



Die of an Intel 80486DX2 microprocessor (actual size: 12×6.75 mm) in its packaging



Intel Core i5 CPU on a Vaio E series laptop motherboard (on the right, beneath the heat pipe)

In the 1970s, the fundamental inventions by Federico Faggin (Silicon Gate MOS ICs with self-aligned gates along with his new random logic design methodology) changed the design and implementation of CPUs forever. Since the introduction of the first commercially available microprocessor (the Intel 4004) in 1970, and the first widely used microprocessor (the Intel 8080) in 1974, this class of CPUs has almost completely overtaken all other central processing unit implementation methods. Mainframe and minicomputer manufacturers of the time launched proprietary IC development programs to upgrade their older computer architectures, and eventually produced instruction set compatible microprocessors that were backward-compatible with their older hardware and software. Combined with the advent and eventual success of the ubiquitous personal computer, the term *CPU* is now applied almost exclusively[lower-alpha 1] to microprocessors. Several CPUs (denoted *cores*) can be combined in a single processing chip.[36]

Previous generations of CPUs were implemented as discrete components and numerous small integrated circuits (ICs) on one or more circuit boards.[37] Microprocessors, on the other hand, are CPUs manufactured on a very small number of ICs; usually just one.[38] The overall smaller CPU size, as a result of being implemented on a single die, means faster switching time because of physical factors like decreased gate parasitic capacitance.[39][40] This has allowed synchronous microprocessors to have clock rates ranging from tens of megahertz to several gigahertz. Additionally, as the ability to construct exceedingly small transistors on an IC has increased, the complexity and number of transistors in a single CPU has increased many fold. This widely observed trend is described by Moore's law, which has proven to be a fairly accurate predictor of the growth of CPU (and other IC) complexity.[41]

While the complexity, size, construction, and general form of CPUs have changed enormously since 1950,[42] it is notable that the basic design and function has not changed much at all. Almost all common CPUs today can be very accurately described as von Neumann stored-program machines.[lower-alpha 2] As the aforementioned Moore's law continues to hold true,[41] concerns have arisen about the limits of integrated circuit transistor technology. Extreme miniaturization of electronic gates is causing the effects of phenomena like electromigration and subthreshold leakage to become much more significant. These newer concerns are among the many factors causing researchers to investigate new methods of computing such as the quantum computer, as well as to expand the usage of parallelism and other methods that extend the usefulness of the classical von Neumann model.

1.2 Operation

The fundamental operation of most CPUs, regardless of the physical form they take, is to execute a sequence of

stored instructions that is called a program. The instructions to be executed are kept in some kind of computer memory. Nearly all CPUs follow the fetch, decode and execute steps in their operation, which are collectively known as the instruction cycle.

After the execution of an instruction, the entire process repeats, with the next instruction cycle normally fetching the next-in-sequence instruction because of the incremented value in the program counter. If a jump instruction was executed, the program counter will be modified to contain the address of the instruction that was jumped to and program execution continues normally. In more complex CPUs, multiple instructions can be fetched, decoded, and executed simultaneously. This section describes what is generally referred to as the "classic RISC pipeline", which is quite common among the simple CPUs used in many electronic devices (often called microcontroller). It largely ignores the important role of CPU cache, and therefore the access stage of the pipeline.

Some instructions manipulate the program counter rather than producing result data directly; such instructions are generally called "jumps" and facilitate program behavior like loops, conditional program execution (through the use of a conditional jump), and existence of functions.[lower-alpha 3] In some processors, some other instructions change the state of bits in a "flags" register. These flags can be used to influence how a program behaves, since they often indicate the outcome of various operations. For example, in such processors a "compare" instruction evaluates two values and sets or clears bits in the flags register to indicate which one is greater or whether they are equal; one of these flags could then be used by a later jump instruction to determine program flow.

1.2.1 Fetch

The first step, fetch, involves retrieving an instruction (which is represented by a number or sequence of numbers) from program memory. The instruction's location (address) in program memory is determined by a program counter (PC), which stores a number that identifies the address of the next instruction to be fetched. After an instruction is fetched, the PC is incremented by the length of the instruction so that it will contain the address of the next instruction in the sequence.[lower-alpha 4] Often, the instruction to be fetched must be retrieved from relatively slow memory, causing the CPU to stall while waiting for the instruction to be returned. This issue is largely addressed in modern processors by caches and pipeline architectures (see below).

1.2.2 Decode

The instruction that the CPU fetches from memory determines what the CPU will do. In the decode step, performed by the circuitry known as the **instruction decoder**, the instruction is converted into signals that control other parts of the CPU.

The way in which the instruction is interpreted is defined by the CPU's instruction set architecture (ISA).[lower-alpha 5] Often, one group of bits (that is, a "field") within the instruction, called the opcode, indicates which operation is to be performed, while the remaining fields usually provide supplemental information required for the operation, such as the operands. Those operands may be specified as a constant value (called an immediate value), or as the location of a value that may be a processor register or a memory address, as determined by some addressing mode.

In some CPU designs the instruction decoder is implemented as a hardwired, unchangeable circuit. In others, a microprogram is used to translate instructions into sets of CPU configuration signals that are applied sequentially over multiple clock pulses. In some cases the memory that stores the microprogram is rewritable, making it possible to change the way in which the CPU decodes instructions.

1.2.3 Execute

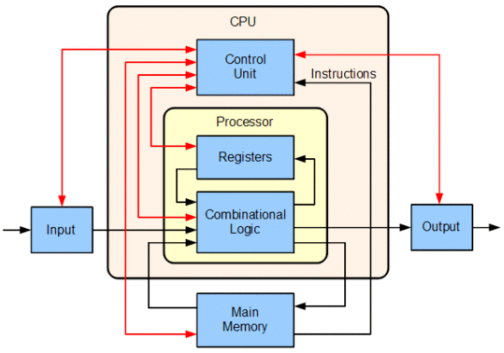
After the fetch and decode steps, the execute step is performed. Depending on the CPU architecture, this may consist of a single action or a sequence of actions. During each action, various parts of the CPU are electrically connected so they can perform all or part of the desired operation and then the action is completed, typically in response to a clock pulse. Very often the results are written to an internal CPU register for quick access by subsequent instructions. In other cases results may be written to slower, but less expensive and higher capacity main memory.

For example, if an addition instruction is to be executed, the arithmetic logic unit (ALU) inputs are connected to a pair of operand sources (numbers to be summed), the ALU is configured to perform an addition operation so that the sum of its operand inputs will appear at its output, and the ALU output is connected to storage (e.g., a register or memory) that will receive the sum. When the clock pulse occurs, the sum will be transferred to storage and, if the resulting sum is too large (i.e., it is larger than the ALU's output word size), an arithmetic overflow flag will be set.

1.3 Structure and implementation

See also: Processor design

Hardwired into a CPU's circuitry is a set of basic operations it can perform, called an instruction set. Such operations may involve, for example, adding or subtracting two numbers, comparing two numbers, or jumping to a



Block diagram of a basic uniprocessor-CPU computer. Black lines indicate data flow, whereas red lines indicate control flow; arrows indicate flow directions.

different part of a program. Each basic operation is represented by a particular combination of bits, known as the machine language opcode; while executing instructions in a machine language program, the CPU decides which operation to perform by "decoding" the opcode. A complete machine language instruction consists of an opcode and, in many cases, additional bits that specify arguments for the operation (for example, the numbers to be summed in the case of an addition operation). Going up the complexity scale, a machine language program is a collection of machine language instructions that the CPU executes.

The actual mathematical operation for each instruction is performed by a combinational logic circuit within the CPU's processor known as the arithmetic logic unit or ALU. In general, a CPU executes an instruction by fetching it from memory, using its ALU to perform an operation, and then storing the result to memory. Beside the instructions for integer mathematics and logic operations, various other machine instructions exist, such as those for loading data from memory and storing it back, branching operations, and mathematical operations on floating-point numbers performed by the CPU's floating-point unit (FPU).[44]

1.3.1 Control unit

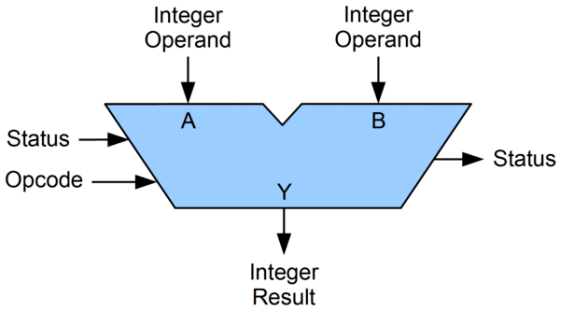
Main article: Control unit

The control unit of the CPU contains circuitry that uses electrical signals to direct the entire computer system to carry out stored program instructions. The control unit does not execute program instructions; rather, it directs other parts of the system to do so. The control unit communicates with both the ALU and memory.

1.3.2 Arithmetic logic unit

Main article: Arithmetic logic unit

The arithmetic logic unit (ALU) is a digital circuit within



Symbolic representation of an ALU and its input and output signals

the processor that performs integer arithmetic and bitwise logic operations. The inputs to the ALU are the data words to be operated on (called operands), status information from previous operations, and a code from the control unit indicating which operation to perform. Depending on the instruction being executed, the operands may come from internal CPU registers or external memory, or they may be constants generated by the ALU itself.

When all input signals have settled and propagated through the ALU circuitry, the result of the performed operation appears at the ALU's outputs. The result consists of both a data word, which may be stored in a register or memory, and status information that is typically stored in a special, internal CPU register reserved for this purpose.

1.3.3 Memory management unit

Main article: Memory management unit

Most high-end microprocessors (in desktop, laptop, server computers) have a memory management unit, translating logical addresses into physical RAM addresses, providing memory protection and paging abilities, useful for virtual memory. Simpler processors, especially microcontrollers usually don't include an MMU.

1.3.4 Integer range

Every CPU represents numerical values in a specific way. For example, some early digital computers represented numbers as familiar decimal (base 10) numeral system values, and others have employed more unusual representations such as ternary (base three). Nearly all modern CPUs represent numbers in binary form, with each digit being represented by some two-valued physical quantity such as a "high" or "low" voltage.[lower-alpha 6]

A six-bit word containing the binary encoded representation of decimal value 40. Most modern CPUs employ word sizes that are a power of two, for example 8, 16, 32 or 64 bits.

Related to numeric representation is the size and precision of integer numbers that a CPU can represent. In the case of a binary CPU, this is measured by the number of bits (significant digits of a binary encoded integer) that the CPU can process in one operation, which is commonly called "word size", “bit width”, “data path width”, “integer precision”, or “integer size”. A CPU’s integer size determines the range of integer values it can directly operate on.[lower-alpha 7] For example, an 8-bit CPU can directly manipulate integers represented by eight bits, which have a range of 256 (2^8) discrete integer values.

Integer range can also affect the number of memory locations the CPU can directly address (an address is an integer value representing a specific memory location). For example, if a binary CPU uses 32 bits to represent a memory address then it can directly address 2^{32} memory locations. To circumvent this limitation and for various other reasons, some CPUs use mechanisms (such as bank switching) that allow additional memory to be addressed.

CPUs with larger word sizes require more circuitry and consequently are physically larger, cost more, and consume more power (and therefore generate more heat). As a result, smaller 4- or 8-bit microcontrollers are commonly used in modern applications even though CPUs with much larger word sizes (such as 16, 32, 64, even 128-bit) are available. When higher performance is required, however, the benefits of a larger word size (larger data ranges and address spaces) may outweigh the disadvantages. A CPU can have internal data paths shorter than the word size to reduce size and cost. For example, even though the IBM System/360 instruction set was a 32-bit instruction set, the System/360 Model 30 and Model 40 had 8-bit data paths in the arithmetic logical unit, so that a 32-bit add required four cycles, one for each 8 bits of the operands, and, even though the Motorola 68k instruction set was a 32-bit instruction set, the Motorola 68000 and Motorola 68010 had 16-bit data paths in the arithmetic logical unit, so that a 32-bit add required two cycles.

To gain some of the advantages afforded by both lower and higher bit lengths, many instruction sets have different bit widths for integer and floating-point data, allowing CPUs implementing that instruction set to have different bit widths for different portions of the device. For example, the IBM System/360 instruction set was primarily 32 bit, but supported 64-bit floating point values to facilitate greater accuracy and range in floating point numbers.[24] The System/360 Model 65 had an 8-bit adder for decimal and fixed-point binary arithmetic and a 60-bit adder for floating-point arithmetic.[45] |Many later CPU designs use similar mixed bit width, especially when the processor is meant for general-purpose usage where a reasonable balance of integer and floating point capability is required.

1.3.5 Clock rate

Main article: Clock rate

Most CPUs are synchronous circuits, which means they employ a clock signal to pace their sequential operations. The clock signal is produced by an external oscillator circuit that generates a consistent number of pulses each second in the form of a periodic square wave. The frequency of the clock pulses determines the rate at which a CPU executes instructions and, consequently, the faster the clock, the more instructions the CPU will execute each second.

To ensure proper operation of the CPU, the clock period is longer than the maximum time needed for all signals to propagate (move) through the CPU. In setting the clock period to a value well above the worst-case propagation delay, it is possible to design the entire CPU and the way it moves data around the “edges” of the rising and falling clock signal. This has the advantage of simplifying the CPU significantly, both from a design perspective and a component-count perspective. However, it also carries the disadvantage that the entire CPU must wait on its slowest elements, even though some portions of it are much faster. This limitation has largely been compensated for by various methods of increasing CPU parallelism (see below).

However, architectural improvements alone do not solve all of the drawbacks of globally synchronous CPUs. For example, a clock signal is subject to the delays of any other electrical signal. Higher clock rates in increasingly complex CPUs make it more difficult to keep the clock signal in phase (synchronized) throughout the entire unit. This has led many modern CPUs to require multiple identical clock signals to be provided to avoid delaying a single signal significantly enough to cause the CPU to malfunction. Another major issue, as clock rates increase dramatically, is the amount of heat that is dissipated by the CPU. The constantly changing clock causes many components to switch regardless of whether they are being used at that time. In general, a component that is switching uses more energy than an element in a static state. Therefore, as clock rate increases, so does energy consumption, causing the CPU to require more heat dissipation in the form of CPU cooling solutions.

One method of dealing with the switching of unneeded components is called clock gating, which involves turning off the clock signal to unneeded components (effectively disabling them). However, this is often regarded as difficult to implement and therefore does not see common usage outside of very low-power designs. One no-

table recent CPU design that uses extensive clock gating is the IBM PowerPC-based Xenon used in the Xbox 360; that way, power requirements of the Xbox 360 are greatly reduced.[46] Another method of addressing some of the problems with a global clock signal is the removal of the clock signal altogether. While removing the global clock signal makes the design process considerably more complex in many ways, asynchronous (or clockless) designs carry marked advantages in power consumption and heat dissipation in comparison with similar synchronous designs. While somewhat uncommon, entire asynchronous CPUs have been built without utilizing a global clock signal. Two notable examples of this are the ARM compliant AMULET and the MIPS R3000 compatible MiniMIPS.

Rather than totally removing the clock signal, some CPU designs allow certain portions of the device to be asynchronous, such as using asynchronous ALUs in conjunction with superscalar pipelining to achieve some arithmetic performance gains. While it is not altogether clear whether totally asynchronous designs can perform at a comparable or better level than their synchronous counterparts, it is evident that they do at least excel in simpler math operations. This, combined with their excellent power consumption and heat dissipation properties, makes them very suitable for embedded computers.[47]

1.3.6 Parallelism

Main article: Parallel computing

The description of the basic operation of a CPU offered

Model of a subscalar CPU, in which it takes fifteen clock cycles to complete three instructions.

in the previous section describes the simplest form that a CPU can take. This type of CPU, usually referred to as *subscalar*, operates on and executes one instruction on one or two pieces of data at a time, that is less than one instruction per clock cycle (IPC < 1).

This process gives rise to an inherent inefficiency in subscalar CPUs. Since only one instruction is executed at a time, the entire CPU must wait for that instruction to complete before proceeding to the next instruction. As a result, the subscalar CPU gets "hung up" on instructions which take more than one clock cycle to complete execution. Even adding a second execution unit (see below) does not improve performance much; rather than one pathway being hung up, now two pathways are hung up and the number of unused transistors is increased. This design, wherein the CPU's execution resources can operate on only one instruction at a time, can only possibly reach *scalar* performance (one instruction per clock cycle, IPC = 1). However, the performance is nearly always subscalar (less than one instruction per clock cycle, IPC < 1).

Attempts to achieve scalar and better performance have resulted in a variety of design methodologies that cause the CPU to behave less linearly and more in parallel. When referring to parallelism in CPUs, two terms are generally used to classify these design techniques:

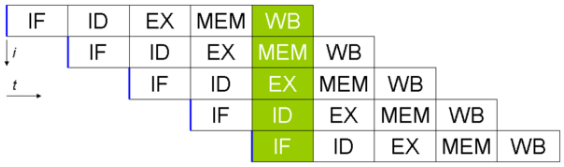
- *instruction-level parallelism* (ILP), which seeks to increase the rate at which instructions are executed within a CPU (that is, to increase the utilization of on-die execution resources);
- *task-level parallelism* (TLP), which purposes to increase the number of threads or processes that a CPU can execute simultaneously.

Each methodology differs both in the ways in which they are implemented, as well as the relative effectiveness they afford in increasing the CPU's performance for an application.[lower-alpha 8]

Instruction-level parallelism

Main articles: Instruction pipelining and Superscalar processor

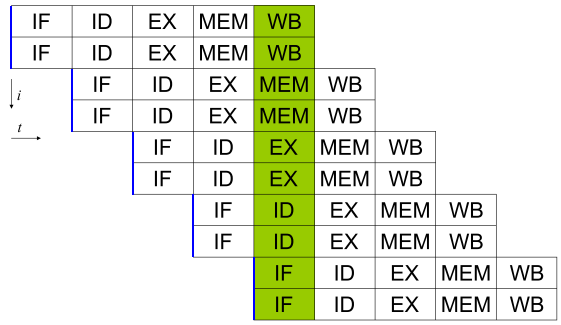
One of the simplest methods used to accomplish increased parallelism is to begin the first steps of instruction fetching and decoding before the prior instruction finishes executing. This is the simplest form of a technique known as instruction pipelining, and is utilized in almost all modern general-purpose CPUs. Pipelining allows more than one instruction to be executed at any given time by breaking down the execution pathway into discrete stages. This separation can be compared to an assembly line, in which an instruction is made more complete at each stage until it exits the execution pipeline and is retired.



Basic five-stage pipeline. In the best case scenario, this pipeline can sustain a completion rate of one instruction per clock cycle.

Pipelining does, however, introduce the possibility for a situation where the result of the previous operation is needed to complete the next operation; a condition often termed data dependency conflict. To cope with this, additional care must be taken to check for these sorts of conditions and delay a portion of the instruction pipeline if this occurs. Naturally, accomplishing this requires additional circuitry, so pipelined processors are more complex than subscalar ones (though not very significantly so). A

pipelined processor can become very nearly scalar, inhibited only by pipeline stalls (an instruction spending more than one clock cycle in a stage).



A simple superscalar pipeline. By fetching and dispatching two instructions at a time, a maximum of two instructions per clock cycle can be completed.

Further improvement upon the idea of instruction pipelining led to the development of a method that decreases the idle time of CPU components even further. Designs that are said to be *superscalar* include a long instruction pipeline and multiple identical execution units.[48] In a superscalar pipeline, multiple instructions are read and passed to a dispatcher, which decides whether or not the instructions can be executed in parallel (simultaneously). If so they are dispatched to available execution units, resulting in the ability for several instructions to be executed simultaneously. In general, the more instructions a superscalar CPU is able to dispatch simultaneously to waiting execution units, the more instructions will be completed in a given cycle.

Most of the difficulty in the design of a superscalar CPU architecture lies in creating an effective dispatcher. The dispatcher needs to be able to quickly and correctly determine whether instructions can be executed in parallel, as well as dispatch them in such a way as to keep as many execution units busy as possible. This requires that the instruction pipeline is filled as often as possible and gives rise to the need in superscalar architectures for significant amounts of CPU cache. It also makes hazard-avoiding techniques like branch prediction, speculative execution, and out-of-order execution crucial to maintaining high levels of performance. By attempting to predict which branch (or path) a conditional instruction will take, the CPU can minimize the number of times that the entire pipeline must wait until a conditional instruction is completed. Speculative execution often provides modest performance increases by executing portions of code that may not be needed after a conditional operation completes. Out-of-order execution somewhat rearranges the order in which instructions are executed to reduce delays due to data dependencies. Also in case of single instruction stream, multiple data stream—a case when a lot of data from the same type has to be processed—, modern processors can disable parts of the pipeline so that when a single instruction is executed many times, the CPU skips the fetch and decode phases and thus greatly increases performance on certain occasions, especially in highly monotonous program engines such as video creation software and photo processing.

In the case where a portion of the CPU is superscalar and part is not, the part which is not suffers a performance penalty due to scheduling stalls. The Intel P5 Pentium had two superscalar ALUs which could accept one instruction per clock cycle each, but its FPU could not accept one instruction per clock cycle. Thus the P5 was integer superscalar but not floating point superscalar. Intel's successor to the P5 architecture, P6, added superscalar capabilities to its floating point features, and therefore afforded a significant increase in floating point instruction performance.

Both simple pipelining and superscalar design increase a CPU's ILP by allowing a single processor to complete execution of instructions at rates surpassing one instruction per clock cycle.[lower-alpha 9] Most modern CPU designs are at least somewhat superscalar, and nearly all general purpose CPUs designed in the last decade are superscalar. In later years some of the emphasis in designing high-ILP computers has been moved out of the CPU's hardware and into its software interface, or ISA. The strategy of the very long instruction word (VLIW) causes some ILP to become implied directly by the software, reducing the amount of work the CPU must perform to boost ILP and thereby reducing the design's complexity.

Task-level parallelism

Main articles: Multithreading and Multi-core processor

Another strategy of achieving performance is to execute multiple threads or processes in parallel. This area of research is known as parallel computing.[49] In Flynn's taxonomy, this strategy is known as multiple instruction stream, multiple data stream (MIMD).[50]

One technology used for this purpose was multiprocessing (MP).[51] The initial flavor of this technology is known as symmetric multiprocessing (SMP), where a small number of CPUs share a coherent view of their memory system. In this scheme, each CPU has additional hardware to maintain a constantly up-to-date view of memory. By avoiding stale views of memory, the CPUs can cooperate on the same program and programs can migrate from one CPU to another. To increase the number of cooperating CPUs beyond a handful, schemes such as non-uniform memory access (NUMA) and directory-based coherence protocols were introduced in the 1990s. SMP systems are limited to a small number of CPUs while NUMA systems have been built with thousands of processors. Initially, multiprocessing was built using multiple discrete CPUs and boards to implement the interconnect between the processors. When the processors and their interconnect

are all implemented on a single chip, the technology is known as chip-level multiprocessing (CMP) and the single chip as a multi-core processor.

It was later recognized that finer-grain parallelism existed with a single program. A single program might have several threads (or functions) that could be executed separately or in parallel. Some of the earliest examples of this technology implemented input/output processing such as direct memory access as a separate thread from the computation thread. A more general approach to this technology was introduced in the 1970s when systems were designed to run multiple computation threads in parallel. This technology is known as multi-threading (MT). This approach is considered more cost-effective than multiprocessing, as only a small number of components within a CPU is replicated to support MT as opposed to the entire CPU in the case of MP. In MT, the execution units and the memory system including the caches are shared among multiple threads. The downside of MT is that the hardware support for multithreading is more visible to software than that of MP and thus supervisor software like operating systems have to undergo larger changes to support MT. One type of MT that was implemented is known as temporal multithreading, where one thread is executed until it is stalled waiting for data to return from external memory. In this scheme, the CPU would then quickly context switch to another thread which is ready to run, the switch often done in one CPU clock cycle, such as the UltraSPARC Technology. Another type of MT is known as simultaneous multithreading, where instructions of multiple threads are executed in parallel within one CPU clock cycle.

For several decades from the 1970s to early 2000s, the focus in designing high performance general purpose CPUs was largely on achieving high ILP through technologies such as pipelining, caches, superscalar execution, out-of-order execution, etc. This trend culminated in large, power-hungry CPUs such as the Intel Pentium 4. By the early 2000s, CPU designers were thwarted from achieving higher performance from ILP techniques due to the growing disparity between CPU operating frequencies and main memory operating frequencies as well as escalating CPU power dissipation owing to more esoteric ILP techniques.

CPU designers then borrowed ideas from commercial computing markets such as transaction processing, where the aggregate performance of multiple programs, also known as throughput computing, was more important than the performance of a single thread or process.

This reversal of emphasis is evidenced by the proliferation of dual and more core processor designs and notably, Intel's newer designs resembling its less superscalar P6 architecture. Late designs in several processor families exhibit CMP, including the x86-64 Opteron and Athlon 64 X2, the SPARC UltraSPARC T1, IBM POWER4 and POWER5, as well as several video game console CPUs like the Xbox 360's triple-core PowerPC design, and the PS3's 7-core Cell microprocessor.

Data parallelism

Main articles: Vector processor and SIMD

A less common but increasingly important paradigm of processors (and indeed, computing in general) deals with data parallelism. The processors discussed earlier are all referred to as some type of scalar device.[lower-alpha 10] As the name implies, vector processors deal with multiple pieces of data in the context of one instruction. This contrasts with scalar processors, which deal with one piece of data for every instruction. Using Flynn's taxonomy, these two schemes of dealing with data are generally referred to as single instruction stream, multiple data stream (SIMD) and single instruction stream, single data stream (SISD), respectively. The great utility in creating processors that deal with vectors of data lies in optimizing tasks that tend to require the same operation (for example, a sum or a dot product) to be performed on a large set of data. Some classic examples of these types of tasks are multimedia applications (images, video, and sound), as well as many types of scientific and engineering tasks. Whereas a scalar processor must complete the entire process of fetching, decoding, and executing each instruction and value in a set of data, a vector processor can perform a single operation on a comparatively large set of data with one instruction. Of course, this is only possible when the application tends to require many steps which apply one operation to a large set of data.

Most early vector processors, such as the Cray-1, were associated almost exclusively with scientific research and cryptography applications. However, as multimedia has largely shifted to digital media, the need for some form of SIMD in general-purpose processors has become significant. Shortly after inclusion of floating-point units started to become commonplace in general-purpose processors, specifications for and implementations of SIMD execution units also began to appear for general-purpose processors. Some of these early SIMD specifications like HP's Multimedia Acceleration eXtensions (MAX) and Intel's MMX were integer-only. This proved to be a significant impediment for some software developers, since many of the applications that benefit from SIMD primarily deal with floating-point numbers. Progressively, these early designs were refined and remade into some of the common, modern SIMD specifications, which are usually associated with one ISA. Some notable modern examples are Intel's SSE and the PowerPC-related AltiVec (also known as VMX).[lower-alpha 11]

1.4 Performance

Further information: Computer performance and Benchmark (computing)

The *performance* or *speed* of a processor depends on, among many other factors, the clock rate (generally given in multiples of hertz) and the instructions per clock (IPC), which together are the factors for the instructions per second (IPS) that the CPU can perform.[52] Many reported IPS values have represented "peak" execution rates on artificial instruction sequences with few branches, whereas realistic workloads consist of a mix of instructions and applications, some of which take longer to execute than others. The performance of the memory hierarchy also greatly affects processor performance, an issue barely considered in MIPS calculations. Because of these problems, various standardized tests, often called "benchmarks" for this purpose—such as SPECint—have been developed to attempt to measure the real effective performance in commonly used applications.

Processing performance of computers is increased by using multi-core processors, which essentially is plugging two or more individual processors (called *cores* in this sense) into one integrated circuit.[53] Ideally, a dual core processor would be nearly twice as powerful as a single core processor. In practice, the performance gain is far smaller, only about 50%, due to imperfect software algorithms and implementation.[54] Increasing the number of cores in a processor (i.e. dual-core, quad-core, etc.) increases the workload that can be handled. This means that the processor can now handle numerous asynchronous events, interrupts, etc. which can take a toll on the CPU when overwhelmed. These cores can be thought of as different floors in a processing plant, with each floor handling a different task. Sometimes, these cores will handle the same tasks as cores adjacent to them if a single core is not enough to handle the information.

Due to specific capabilities of modern CPUs, such as hyper-threading and uncore, which involve sharing of actual CPU resources while aiming at increased utilization, monitoring performance levels and hardware utilization gradually became a more complex task.[55] As a response, some CPUs implement additional hardware logic that monitors actual utilization of various parts of a CPU and provides various counters accessible to software; an example is Intel's *Performance Counter Monitor* technology.[3]

1.5 See also

- Addressing mode
- AMD Accelerated Processing Unit
- CISC
- Computer bus
- Computer engineering
- CPU core voltage
- CPU socket
- Digital signal processor
- Hyper-threading
- List of CPU architectures
- Microprocessor
- Multi-core processor
- Protection ring
- RISC
- Stream processing
- True Performance Index
- Wait state

1.6 Notes

[1] Integrated circuits are now used to implement all CPUs, except for a few machines designed to withstand large electromagnetic pulses, say from a nuclear weapon.

[2] The so-called "von Neumann" memo expounded the idea of stored programs,[43] which for example may be stored on punched cards, paper tape, or magnetic tape.

[3] Some early computers like the Harvard Mark I did not support any kind of "jump" instruction, effectively limiting the complexity of the programs they could run. It is largely for this reason that these computers are often not considered to contain a proper CPU, despite their close similarity to stored-program computers.

[4] Since the program counter counts *memory addresses* and not *instructions*, it is incremented by the number of memory units that the instruction word contains. In the case of simple fixed-length instruction word ISAs, this is always the same number. For example, a fixed-length 32-bit instruction word ISA that uses 8-bit memory words would always increment the PC by four (except in the case of jumps). ISAs that use variable-length instruction words increment the PC by the number of memory words corresponding to the last instruction's length.

[5] Because the instruction set architecture of a CPU is fundamental to its interface and usage, it is often used as a classification of the "type" of CPU. For example, a "PowerPC CPU" uses some variant of the PowerPC ISA. A system can execute a different ISA by running an emulator.

[6] The physical concept of voltage is an analog one by nature, practically having an infinite range of possible values. For the purpose of physical representation of binary numbers, two specific ranges of voltages are defined, one for logic '0' and another for logic '1'. These ranges are dictated by design considerations such as noise margins and characteristics of the devices used to create the CPU.

[7] While a CPU's integer size sets a limit on integer ranges, this can (and often is) overcome using a combination of software and hardware techniques. By using additional memory, software can represent integers many magnitudes larger than the CPU can. Sometimes the CPU's instruction set will even facilitate operations on integers larger than it can natively represent by providing instructions to make large integer arithmetic relatively quick. This method of dealing with large integers is slower than utilizing a CPU with higher integer size, but is a reasonable trade-off in cases where natively supporting the full integer range needed would be cost-prohibitive. See Arbitrary-precision arithmetic for more details on purely software-supported arbitrary-sized integers.

[8] Neither ILP nor TLP is inherently superior over the other; they are simply different means by which to increase CPU parallelism. As such, they both have advantages and disadvantages, which are often determined by the type of software that the processor is intended to run. High-TLP CPUs are often used in applications that lend themselves well to being split up into numerous smaller applications, so-called "embarrassingly parallel problems". Frequently, a computational problem that can be solved quickly with high TLP design strategies like symmetric multiprocessing takes significantly more time on high ILP devices like superscalar CPUs, and vice versa.

[9] Best-case scenario (or peak) IPC rates in very superscalar architectures are difficult to maintain since it is impossible to keep the instruction pipeline filled all the time. Therefore, in highly superscalar CPUs, average sustained IPC is often discussed rather than peak IPC.

[10] Earlier the term scalar was used to compare the IPC count afforded by various ILP methods. Here the term is used in the strictly mathematical sense to contrast with vectors. See scalar (mathematics) and Vector (geometric).

[11] Although SSE/SSE2/SSE3 have superseded MMX in Intel's general-purpose processors, later IA-32 designs still support MMX. This is usually accomplished by providing most of the MMX functionality with the same hardware that supports the much more expansive SSE instruction sets.

1.7 References

[1] Weik, Martin H. (1961). "A Third Survey of Domestic Electronic Digital Computing Systems". Ballistic Research Laboratory.

[2] Kuck, David (1978). *Computers and Computations, Vol 1*. John Wiley & Sons, Inc. p. 12. ISBN 0471027162.

[3] Thomas Willhalm; Roman Dementiev; Patrick Fay (December 18, 2014). "Intel Performance Counter Monitor – A better way to measure CPU utilization". *software.intel.com*. Retrieved February 17, 2015.

[4] Regan, Gerard. *A Brief History of Computing*. p. 66. ISBN 1848000839. Retrieved 26 November 2014.

[5] "Bit By Bit". Haverford College. Archived from the original on October 13, 2012. Retrieved August 1, 2015.

[6] "First Draft of a Report on the EDVAC" (PDF). Moore School of Electrical Engineering, University of Pennsylvania. 1945.

[7] Stanford University. "The Modern History of Computing". The Stanford Encyclopedia of Philosophy. Retrieved September 25, 2015.

[8] Enticknap, Nicholas (Summer 1998), "Computing's Golden Jubilee", *Resurrection* (The Computer Conservation Society) (20), ISSN 0958-7403, retrieved 19 April 2008

[9] "The Manchester Mark 1". *The University of Manchester*. Retrieved September 25, 2015.

[10] "The First Generation". Computer History Museum. Retrieved September 29, 2015.

[11] "The History of the Integrated Circuit". *Nobelprize.org*. Retrieved September 29, 2015.

[12] Turley, Jim. "Motoring with microprocessors". Embedded. Retrieved November 15, 2015.

[13] "Mobile Processor Guide – Summer 2013". Android Authority. Retrieved November 15, 2015.

[14] "ARM946 Processor". ARM. Retrieved November 15, 2015.

[15] "Konrad Zuse". Computer History Museum. Retrieved September 29, 2015.

[16] "Timeline of Computer History: Computers". Computer History Museum. Retrieved November 21, 2015.

[17] White, Stephen. "A Brief History of Computing - First Generation Computers". Retrieved November 21, 2015.

[18] "Harvard University Mark - Paper Tape Punch Unit". Computer History Museum. Retrieved November 21, 2015.

[19] "What is the difference between a von Neumann architecture and a Harvard architecture?". ARM. Retrieved November 22, 2015.

[20] "Advanced Architecture Optimizes the Atmel AVR CPU". Atmel. Retrieved November 22, 2015.

[21] "Switches, transistors and relays". BBC. Retrieved 7 February 2016.

[22] "Introducing the Vacuum Transistor: A Device Made of Nothing". IEEE Spectrum. Retrieved 7 February 2016.

[23] "IBM System/360 Dates and Characteristics". IBM.

[24] Amdahl, G. M.; Blaauw, G. A.; Brooks, F. P. Jr. (April 1964). "Architecture of the IBM System/360". *IBM Journal of Research and Development* (IBM) **8** (2): 87–101. doi:10.1147/rd.82.0087. ISSN 0018-8646.

[25] Brodkin, John. "50 years ago, IBM created mainframe that helped send men to the Moon". Ars Technica. Retrieved 9 April 2016.

[26] Clarke, Gavin. "Why won't you DIE? IBM's S/360 and its legacy at 50". The Register. Retrieved 9 April 2016.

[27] "Online PDP-8 Home Page, Run a PDP-8". *PDP8*. Retrieved September 25, 2015.

[28] "Transistors, Relays, and Controlling High-Current Loads". *New York University*. ITP Physical Computing. Retrieved 9 April 2016.

[29] Patterson, David A.; Hennessy, John L.; Larus, James R. (1999). *Computer organization and design : the hardware/software interface* (2. ed., 3rd print. ed.). San Francisco: Kaufmann. p. 751. ISBN 1558604286.

[30] Ross Knox Bassett. "To the Digital Age: Research Labs, Start-up Companies, and the Rise of MOS Technology". 2007. p. 127-128, 256, and 314.

[31] Ken Shirriff. "The Texas Instruments TMX 1795: the first, forgotten microprocessor".

[32] "Speed & Power in Logic Families".

[33] T. J. Stonham. "Digital Logic Techniques: Principles and Practice". 1996. p. 174.

[34] R. K. Booher. "MOS GP Computer". afips, pp.877, 1968 Proceedings of the Fall Joint Computer Conference, 1968 doi:10.1109/AFIPS.1968.126

[35] "LSI-11 Module Descriptions". *LSI-11, PDP-11/03 user's manual* (PDF) (2nd ed.). Maynard, Massachusetts: Digital Equipment Corporation. November 1975. pp. 4–3.

[36] Margaret Rouse (March 27, 2007). "Definition: multi-core processor". TechTarget. Retrieved March 6, 2013.

[37] Richard Birkby. "A Brief History of the Microprocessor". *computermuseum.li*. Retrieved October 13, 2015.

[38] Osborne, Adam (1980). *An Introduction to Microcomputers*. Volume 1: Basic Concepts (2nd ed.). Berkeley, California: Osborne-McGraw Hill. ISBN 0-931988-34-9.

[39] Zhislina, Victoria. "Why has CPU frequency ceased to grow?". Intel. Retrieved October 14, 2015.

[40] "MOS Transistor - Electrical Engineering & Computer Science" (PDF). University of California. Retrieved October 14, 2015.

[41] "Excerpts from A Conversation with Gordon Moore: Moore's Law" (PDF). Intel. 2005. Retrieved 2012-07-25.

[42] Lilly, Paul. "A Brief History of CPUs: 31 Awesome Years of x86". Maximum PC. Retrieved December 10, 2015.

[43] Aspray, William. "The stored program concept" (PDF). Spectrum, IEEE. Retrieved September 29, 2015.

[44] Ian Wienand (September 3, 2013). "Computer Science from the Bottom Up, Chapter 3. Computer Architecture" (PDF). *bottomupcs.com*. Retrieved January 7, 2015.

[45] "IBM System/360 Model 65 Functional Characteristics" (PDF). IBM. September 1968. pp. 8–9. A22-6884-3.

[46] Brown, Jeffery (2005). "Application-customized CPU design". IBM developerWorks. Retrieved 2005-12-17.

[47] Garside, J. D., Furber, S. B., & Chung, S-H (1999). "AMULET3 Revealed". University of Manchester Computer Science Department. Archived from the original on December 10, 2005.

[48] Huynh, Jack (2003). "The AMD Athlon XP Processor with 512KB L2 Cache" (PDF). University of Illinois, Urbana-Champaign. pp. 6–11. Retrieved 2007-10-06.

[49] Gottlieb, Allan; Almasi, George S. (1989). *Highly parallel computing*. Redwood City, Calif.: Benjamin/Cummings. ISBN 0-8053-0177-1.

[50] Flynn, M. J. (September 1972). "Some Computer Organizations and Their Effectiveness". *IEEE Trans. Comput.* **C–21** (9): 948–960. doi:10.1109/TC.1972.5009071.

[51] "Parallelism exploitation in superscalar multiprocessing" (PDF). IEEE Xplore. Retrieved 25 April 2016.

[52] "CPU Frequency". *CPU World Glossary*. CPU World. 25 March 2008. Retrieved 1 January 2010.

[53] "What is (a) multi-core processor?". *Data Center Definitions*. SearchDataCenter.com. 27 March 2007. Retrieved 1 January 2010.

[54] "Quad Core Vs. Dual Core". Buzzle. Retrieved 26 November 2014.

[55] Tegtmeier, Martin. "CPU utilization of multi-threaded architectures explained". Oracle. Retrieved September 29, 2015.

1.8 External links

- How Microprocessors Work at HowStuffWorks.
- 25 Microchips that shook the world – an article by the Institute of Electrical and Electronics Engineers.

Chapter 2

Digital signal processor

See also: Digital signal processing

A **digital signal processor** (**DSP**) is a specialized

A digital signal processor chip found in a guitar effects unit.

microprocessor (or a SIP block), with its architecture optimized for the operational needs of digital signal processing.[1][2]

The goal of DSPs is usually to measure, filter and/or compress continuous real-world analog signals. Most general-purpose microprocessors can also execute digital signal processing algorithms successfully, but dedicated DSPs usually have better power efficiency thus they are more suitable in portable devices such as mobile phones because of power consumption constraints.[3] DSPs often use special memory architectures that are able to fetch multiple data and/or instructions at the same time.

2.1 Overview

A typical digital processing system

Digital signal processing algorithms typically require a large number of mathematical operations to be performed quickly and repeatedly on a series of data samples. Signals (perhaps from audio or video sensors) are constantly converted from analog to digital, manipulated digitally, and then converted back to analog form. Many DSP applications have constraints on latency; that is, for the system to work, the DSP operation must be completed within some fixed time, and deferred (or batch) processing is not viable.

Most general-purpose microprocessors and operating systems can execute DSP algorithms successfully, but are not suitable for use in portable devices such as mobile phones and PDAs because of power efficiency constraints.[3] A specialized digital signal processor, however, will tend to provide a lower-cost solution, with better performance, lower latency, and no requirements for specialized cooling or large batteries.

The architecture of a digital signal processor is optimized specifically for digital signal processing. Most also support some of the features as an applications processor or microcontroller, since signal processing is rarely the only task of a system. Some useful features for optimizing DSP algorithms are outlined below.

2.2 Architecture

2.2.1 Software architecture

By the standards of general-purpose processors, DSP instruction sets are often highly irregular. One implication for software architecture is that hand-optimized assembly-code routines are commonly packaged into li-

braries for re-use, instead of relying on advanced compiler technologies to handle essential algorithms.

Instruction sets

- multiply–accumulates (MACs, including fused multiply–add, FMA) operations
  - used extensively in all kinds of matrix operations
    - convolution for filtering
    - dot product
    - polynomial evaluation
  - Fundamental DSP algorithms depend heavily on multiply–accumulate performance
    - FIR filters
    - Fast Fourier transform (FFT)
- Instructions to increase parallelism:
  - SIMD
  - VLIW
  - superscalar architecture
- Specialized instructions for modulo addressing in ring buffers and bit-reversed addressing mode for FFT cross-referencing
- Digital signal processors sometimes use time-stationary encoding to simplify hardware and increase coding efficiency.
- Multiple arithmetic units may require memory architectures to support several accesses per instruction cycle
- Special loop controls, such as architectural support for executing a few instruction words in a very tight loop without overhead for instruction fetches or exit testing

Data instructions

- Saturation arithmetic, in which operations that produce overflows will accumulate at the maximum (or minimum) values that the register can hold rather than wrapping around (maximum+1 doesn't overflow to minimum as in many general-purpose CPUs, instead it stays at maximum). Sometimes various sticky bits operation modes are available.
- Fixed-point arithmetic is often used to speed up arithmetic processing
- Single-cycle operations to increase the benefits of pipelining

Program flow

- Floating-point unit integrated directly into the datapath
- Pipelined architecture
- Highly parallel multiplier–accumulators (MAC units)
- Hardware-controlled looping, to reduce or eliminate the overhead required for looping operations

2.2.2 Hardware architecture

Memory architecture

DSPs are usually optimized for streaming data and use special memory architectures that are able to fetch multiple data and/or instructions at the same time, such as the Harvard architecture or Modified von Neumann architecture, which use separate program and data memories (sometimes even concurrent access on multiple data buses).

DSPs can sometimes rely on supporting code to know about cache hierarchies and the associated delays. This is a tradeoff that allows for better performance. In addition, extensive use of DMA is employed.

Addressing and virtual memory DSPs frequently use multi-tasking operating systems, but have no support for virtual memory or memory protection. Operating systems that use virtual memory require more time for context switching among processes, which increases latency.

- Hardware modulo addressing
  - Allows circular buffers to be implemented without having to test for wrapping
- Bit-reversed addressing, a special addressing mode
  - useful for calculating FFTs
- Exclusion of a memory management unit
- Memory-address calculation unit

2.3 History

Prior to the advent of stand-alone DSP chips discussed below, most DSP applications were implemented using bit-slice processors. The AMD 2901 bit-slice chip with its family of components was a very popular choice.

There were reference designs from AMD, but very often the specifics of a particular design were application specific. These bit slice architectures would sometimes include a peripheral multiplier chip. Examples of these multipliers were a series from TRW including the TDC1008 and TDC1010, some of which included an accumulator, providing the requisite multiply–accumulate (MAC) function.

In 1976, Richard Wiggins proposed the Speak & Spell concept to Paul Breedlove, Larry Brantingham, and Gene Frantz at Texas Instrument's Dallas research facility. Two years later in 1978 they produced the first Speak & Spell, with the technological centerpiece being the TMS5100,[4] the industry's first digital signal processor. It also set other milestones, being the first chip to use Linear predictive coding to perform speech synthesis.[5]

In 1978, Intel released the 2920 as an "analog signal processor". It had an on-chip ADC/DAC with an internal signal processor, but it didn't have a hardware multiplier and was not successful in the market. In 1979, AMI released the S2811. It was designed as a microprocessor peripheral, and it had to be initialized by the host. The S2811 was likewise not successful in the market.

In 1980 the first stand-alone, complete DSPs – the NEC μPD7720 and AT&T DSP1 – were presented at the International Solid-State Circuits Conference '80. Both processors were inspired by the research in PSTN telecommunications.

The Altamira DX-1 was another early DSP, utilizing quad integer pipelines with delayed branches and branch prediction.

Another DSP produced by Texas Instruments (TI), the TMS32010 presented in 1983, proved to be an even bigger success. It was based on the Harvard architecture, and so had separate instruction and data memory. It already had a special instruction set, with instructions like load-and-accumulate or multiply-and-accumulate. It could work on 16-bit numbers and needed 390 ns for a multiply–add operation. TI is now the market leader in general-purpose DSPs.

About five years later, the second generation of DSPs began to spread. They had 3 memories for storing two operands simultaneously and included hardware to accelerate tight loops, they also had an addressing unit capable of loop-addressing. Some of them operated on 24-bit variables and a typical model only required about 21 ns for a MAC. Members of this generation were for example the AT&T DSP16A or the Motorola 56000.

The main improvement in the third generation was the appearance of application-specific units and instructions in the data path, or sometimes as coprocessors. These units allowed direct hardware acceleration of very specific but complex mathematical problems, like the Fourier-transform or matrix operations. Some chips, like the Motorola MC68356, even included more than one processor core to work in parallel. Other DSPs from 1995 are the TI TMS320C541 or the TMS 320C80.

The fourth generation is best characterized by the changes in the instruction set and the instruction encoding/decoding. SIMD extensions were added, VLIW and the superscalar architecture appeared. As always, the clock-speeds have increased, a 3 ns MAC now became possible.

2.4 Modern DSPs

Modern signal processors yield greater performance; this is due in part to both technological and architectural advancements like lower design rules, fast-access two-level cache, (E)DMA circuitry and a wider bus system. Not all DSPs provide the same speed and many kinds of signal processors exist, each one of them being better suited for a specific task, ranging in price from about US$1.50 to US$300

Texas Instruments produces the C6000 series DSPs, which have clock speeds of 1.2 GHz and implement separate instruction and data caches. They also have an 8 MiB 2nd level cache and 64 EDMA channels. The top models are capable of as many as 8000 MIPS (instructions per second), use VLIW (very long instruction word), perform eight operations per clock-cycle and are compatible with a broad range of external peripherals and various buses (PCI/serial/etc). TMS320C6474 chips each have three such DSPs, and the newest generation C6000 chips support floating point as well as fixed point processing.

Freescale produces a multi-core DSP family, the MSC81xx. The MSC81xx is based on StarCore Architecture processors and the latest MSC8144 DSP combines four programmable SC3400 StarCore DSP cores. Each SC3400 StarCore DSP core has a clock speed of 1 GHz.

XMOS produces a multi-core multi-threaded line of processor well suited to DSP operations, They come in various speeds ranging from 400 to 1600 MIPS. The processors have a multi-threaded architecture that allows up to 8 real-time threads per core, meaning that a 4 core device would support up to 32 real time threads. Threads communicate between each other with buffered channels that are capable of up to 80 Mbit/s. The devices are easily programmable in C and aim at bridging the gap between conventional micro-controllers and FPGAs

CEVA, Inc. produces and licenses three distinct families of DSPs. Perhaps the best known and most widely deployed is the CEVA-TeakLite DSP family, a classic memory-based architecture, with 16-bit or 32-bit word-widths and single or dual MACs. The CEVA-X DSP family offers a combination of VLIW and SIMD architectures, with different members of the family offering dual or quad 16-bit MACs. The CEVA-XC DSP family targets Software-defined Radio (SDR) modem designs

and leverages a unique combination of VLIW and Vector architectures with 32 16-bit MACs.

Analog Devices produce the SHARC-based DSP and range in performance from 66 MHz/198 MFLOPS (million floating-point operations per second) to 400 MHz/2400 MFLOPS. Some models support multiple multipliers and ALUs, SIMD instructions and audio processing-specific components and peripherals. The Blackfin family of embedded digital signal processors combine the features of a DSP with those of a general use processor. As a result, these processors can run simple operating systems like μCLinux, velOSity and Nucleus RTOS while operating on real-time data.

NXP Semiconductors produce DSPs based on TriMedia VLIW technology, optimized for audio and video processing. In some products the DSP core is hidden as a fixed-function block into a SoC, but NXP also provides a range of flexible single core media processors. The TriMedia media processors support both fixed-point arithmetic as well as floating-point arithmetic, and have specific instructions to deal with complex filters and entropy coding.

CSR produces the Quatro family of SoCs that contain one or more custom Imaging DSPs optimized for processing document image data for scanner and copier applications.

Most DSPs use fixed-point arithmetic, because in real world signal processing the additional range provided by floating point is not needed, and there is a large speed benefit and cost benefit due to reduced hardware complexity. Floating point DSPs may be invaluable in applications where a wide dynamic range is required. Product developers might also use floating point DSPs to reduce the cost and complexity of software development in exchange for more expensive hardware, since it is generally easier to implement algorithms in floating point.

Generally, DSPs are dedicated integrated circuits; however DSP functionality can also be produced by using field-programmable gate array chips (FPGAs).

Embedded general-purpose RISC processors are becoming increasingly DSP like in functionality. For example, the OMAP3 processors include a ARM Cortex-A8 and C6000 DSP.

In Communications a new breed of DSPs offering the fusion of both DSP functions and H/W acceleration function is making its way into the mainstream. Such Modem processors include ASOCS ModemX and CEVA's XC4000.

2.5 See also

- Digital signal controller
- Graphics processing unit
- Video processing unit
- Vision processing unit
- MDSP - a multiprocessor DSP

2.6 References

[1] Dyer, S. A.; Harms, B. K. (1993). "Digital Signal Processing". In Yovits, M. C. *Advances in Computers* **37**. Academic Press. pp. 104–107. doi:10.1016/S0065-2458(08)60403-9. ISBN 9780120121373.

[2] Liptak, B. G. (2006). *Process Control and Optimization*. Instrument Engineers' Handbook **2** (4th ed.). CRC Press. pp. 11–12. ISBN 9780849310812.

[3] Ingrid Verbauwhede; Patrick Schaumont; Christian Piguet; Bart Kienhuis (2005-12-24). "Architectures and Design techniques for energy efficient embedded DSP and multimedia processing" (PDF). rijndael.ece.vt.edu. Retrieved 2014-06-11.

[4] "Speak & Spell, the First Use of a Digital Signal Processing IC for Speech Generation, 1978". *IEEE Milestones*. IEEE. Retrieved 2012-03-02.

[5] Bogdanowicz, A. (2009-10-06). "IEEE Milestones Honor Three". *The Institute*. IEEE. Retrieved 2012-03-02.

2.7 External links

- DSP Online Book
- Pocket Guide to Processors for DSP - Berkeley Design Technology, INC

Chapter 3

Graphics processing unit

Not to be confused with Graphics card.
"GPU" redirects here. For other uses, see GPU (disambiguation).

A **graphics processing unit** (**GPU**), also occasionally called **visual processing unit** (**VPU**), is a specialized electronic circuit designed to rapidly manipulate and alter memory to accelerate the creation of images in a frame buffer intended for output to a display. GPUs are used in embedded systems, mobile phones, personal computers, workstations, and game consoles. Modern GPUs are very efficient at manipulating computer graphics and image processing, and their highly parallel structure makes them more efficient than general-purpose CPUs for algorithms where the processing of large blocks of data is done in parallel. In a personal computer, a GPU can be present on a video card, or it can be embedded on the motherboard or—in certain CPUs—on the CPU die.[1]

GeForce 6600GT (NV43) GPU

The term GPU was popularized by Nvidia in 1999, who marketed the GeForce 256 as "the world's first GPU", or Graphics Processing Unit.[2] It was presented as a "single-chip processor with integrated transform, lighting, triangle setup/clipping, and rendering engines that are capable of processing a minimum of 10 million polygons per second".[3] Rival ATI Technologies coined the term visual processing unit or VPU with the release of the Radeon 9700 in 2002.[4]

3.1 History

See also: Video Display Controller, List of home computers by video hardware, and Sprite (computer graphics)

3.1.1 1970s

Arcade system boards have been using specialized graphics chips since the 1970s. The key to understanding early video games hardware is that the RAM for frame buffers was too expensive, so video chips composited data together as the display was being scanned out on the monitor.[5]

Fujitsu's MB14241 video shifter was used to accelerate the drawing of sprite graphics for various 1970s arcade games from Taito and Midway, such as *Gun Fight* (1975), *Sea Wolf* (1976) and *Space Invaders* (1978).[6][7][8] The Namco Galaxian arcade system in 1979 used specialized graphics hardware supporting RGB color, multi-colored sprites and tilemap backgrounds.[9] The Galaxian hardware was widely used during the golden age of arcade video games, by game companies such as Namco, Centuri, Gremlin, Irem, Konami, Midway, Nichibutsu, Sega and Taito.[10][11]

In the home market, the Atari 2600 in 1977 used a video shifter called the Television Interface Adaptor.[12] The Atari 8-bit computers (1979) had ANTIC, a video processor which interpreted instructions describing a "display list"—the way the scan lines map to specific bitmapped or character modes and where the memory is stored (so there did not need to be a contiguous frame buffer).[13] 6502 machine code subroutines could be triggered on scan lines by setting a bit on a display list instruction.[14] ANTIC also supported smooth vertical and horizontal scrolling independent of the CPU.[15]

3.1.2 1980s

The Williams Electronics arcade games *Robotron: 2084* , *Joust*, *Sinistar*, and *Bubbles*, all released in 1982, contain custom blitter chips for operating on 16-color bitmaps.[16][17]

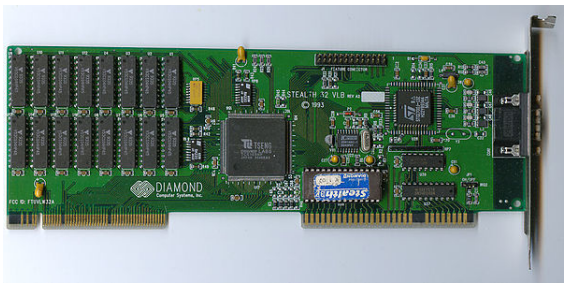
In 1985, the Commodore Amiga featured a custom graphics chip, with a blitter unit accelerating bitmap manipulation, line draw, and area fill functions. Also included is a coprocessor with its own primitive instruction set, capable of manipulating graphics hardware registers in sync with the video beam (e.g. for per-scanline palette switches, sprite multiplexing, and hardware windowing), or driving the blitter.

In 1986, Texas Instruments released the TMS34010, the first microprocessor with on-chip graphics capabilities. It could run general-purpose code, but it had a very graphics-oriented instruction set. In 1990-1991, this chip would become the basis of the Texas Instruments Graphics Architecture ("TIGA") Windows accelerator cards.

In 1987, the IBM 8514 graphics system was released as one of the first video cards for IBM PC compatibles to implement fixed-function 2D primitives in electronic hardware. The same year, Sharp released the X68000, which used a custom graphics chipset[18] that was powerful for a home computer at the time, with a 65,536 color palette and hardware support for sprites, scrolling and multiple playfields,[19] eventually serving as a development machine for Capcom's CP System arcade board. Fujitsu later competed with the FM Towns computer, released in 1989 with support for a full 16,777,216 color palette.[20]

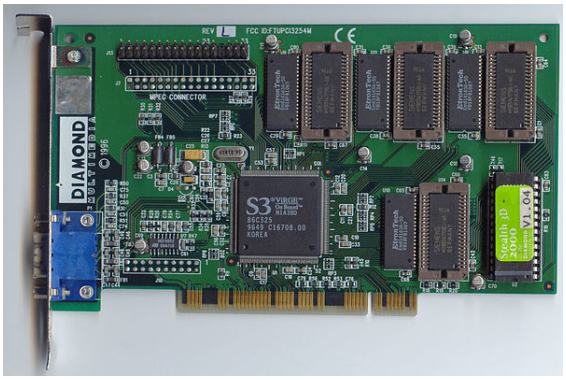
In 1988, the first dedicated polygonal 3D graphics boards were introduced in arcades with the Namco System 21[21] and Taito Air System.[22]

3.1.3 1990s

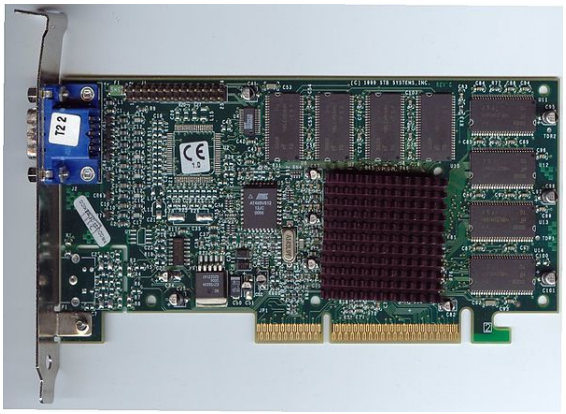


Tseng Labs ET4000/W32p

In 1991, S3 Graphics introduced the *S3 86C911*, which its designers named after the Porsche 911 as an implication of the performance increase it promised.[23] The 86C911 spawned a host of imitators: by 1995, all major PC graphics chip makers had added 2D acceleration support to their chips.[24][25] By this time, fixed-function *Windows accelerators* had surpassed expensive general-purpose graphics coprocessors in Windows performance, and these coprocessors faded away from the PC market.



S3 Graphics ViRGE



Voodoo3 2000 AGP card

Throughout the 1990s, 2D GUI acceleration continued to evolve. As manufacturing capabilities improved, so did the level of integration of graphics chips. Additional application programming interfaces (APIs) arrived for a variety of tasks, such as Microsoft's WinG graphics library for Windows 3.x, and their later DirectDraw interface for hardware acceleration of 2D games within Windows 95 and later.

In the early- and mid-1990s, CPU-assisted real-time 3D graphics were becoming increasingly common in arcade, computer and console games, which led to an increasing public demand for hardware-accelerated 3D graphics. Early examples of mass-market 3D graphics hardware can be found in arcade system boards such as the Sega Model 1, Namco System 22, and Sega Model 2, and the fifth-generation video game consoles such as the Saturn, PlayStation and Nintendo 64. Arcade systems such as the Sega Model 2 and Namco Magic Edge Hornet Simulator in 1993 were capable of hardware T&L (transform, clipping, and lighting) years before appearing in consumer graphics cards.[26][27]Other systems used DSPs to accelerate transformations.Fujitsu, which worked on the Sega Model 2 arcade system,[28] began working on integrating T&L into a single LSI solution for use in home computers in 1995;[29][30] the Fujitsu Pinolite, the first 3D

geometry processor for personal computers, released in 1997.[31] The first hardware T&L GPU on home video game consoles was the Nintendo 64's Reality Coprocessor, released in 1996.[32] In 1997, Mitsubishi released the 3Dpro/2MP, a fully featured GPU capable of transformation and lighting, for workstations and Windows NT desktops;[33] AMD utilized it for their FireGL 4000 graphics card, released in 1997.[34]

In the PC world, notable failed first tries for low-cost 3D graphics chips were the S3 *ViRGE*, ATI *Rage*, and Matrox *Mystique*. These chips were essentially previous-generation 2D accelerators with 3D features bolted on. Many were even pin-compatible with the earlier-generation chips for ease of implementation and minimal cost. Initially, performance 3D graphics were possible only with discrete boards dedicated to accelerating 3D functions (and lacking 2D GUI acceleration entirely) such as the PowerVR and the 3dfx *Voodoo*. However, as manufacturing technology continued to progress, video, 2D GUI acceleration and 3D functionality were all integrated into one chip. Rendition's *Verite* chipsets were among the first to do this well enough to be worthy of note. In 1997, Rendition went a step further by collaborating with Hercules and Fujitsu on a "Thriller Conspiracy" project which combined a Fujitsu FXG-1 Pinolite geometry processor with a Vérité V2200 core to create a graphics card with a full T&L engine years before Nvidia's GeForce 256. This card, designed to reduce the load placed upon the system's CPU, never made it to market.

OpenGL appeared in the early '90s as a professional graphics API, but originally suffered from performance issues which allowed the Glide API to step in and become a dominant force on the PC in the late '90s.[35] However, these issues were quickly overcome and the Glide API fell by the wayside. Software implementations of OpenGL were common during this time, although the influence of OpenGL eventually led to widespread hardware support. Over time, a parity emerged between features offered in hardware and those offered in OpenGL. DirectX became popular among Windows game developers during the late 90s. Unlike OpenGL, Microsoft insisted on providing strict one-to-one support of hardware. The approach made DirectX less popular as a standalone graphics API initially, since many GPUs provided their own specific features, which existing OpenGL applications were already able to benefit from, leaving DirectX often one generation behind. (See: Comparison of OpenGL and Direct3D.)

Over time, Microsoft began to work more closely with hardware developers, and started to target the releases of DirectX to coincide with those of the supporting graphics hardware. Direct3D 5.0 was the first version of the burgeoning API to gain widespread adoption in the gaming market, and it competed directly with many more-hardware-specific, often proprietary graphics libraries, while OpenGL maintained a strong following. Direct3D 7.0 introduced support for hardware-accelerated transform and lighting (T&L) for Direct3D, while OpenGL had this capability already exposed from its inception. 3D accelerator cards moved beyond being just simple rasterizers to add another significant hardware stage to the 3D rendering pipeline. The Nvidia *GeForce 256* (also known as NV10) was the first consumer-level card released on the market with hardware-accelerated T&L, while professional 3D cards already had this capability. Hardware transform and lighting, both already existing features of OpenGL, came to consumer-level hardware in the '90s and set the precedent for later pixel shader and vertex shader units which were far more flexible and programmable.

3.1.4 2000 to 2006

Nvidia was first to produce a chip capable of programmable shading, the *GeForce 3* (code named NV20). Each pixel could now be processed by a short program that could include additional image textures as inputs, and each geometric vertex could likewise be processed by a short program before it was projected onto the screen. Used in the Xbox console, it competed with the PlayStation 2, which used a custom vector DSP for hardware accelerated vertex processing.

By October 2002, with the introduction of the ATI *Radeon 9700* (also known as R300), the world's first Direct3D 9.0 accelerator, pixel and vertex shaders could implement looping and lengthy floating point math, and were quickly becoming as flexible as CPUs, yet orders of magnitude faster for image-array operations. Pixel shading is often used for bump mapping, which adds texture, to make an object look shiny, dull, rough, or even round or extruded.[36]

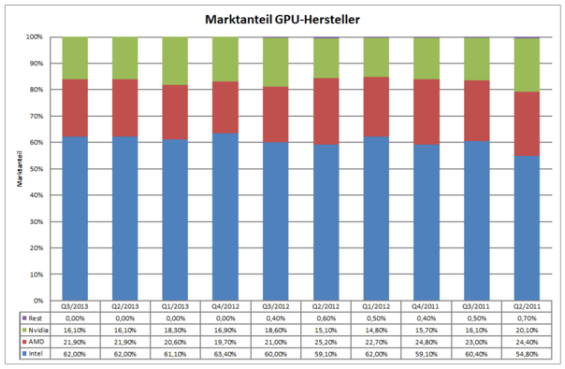
3.1.5 2006 to present

With the introduction of the GeForce 8 series, which was produced by Nvidia, and then new generic stream processing unit GPUs became a more generalized computing device. Today, parallel GPUs have begun making computational inroads against the CPU, and a subfield of research, dubbed GPU Computing or GPGPU for *General Purpose Computing on GPU*, has found its way into fields as diverse as machine learning,[37] oil exploration, scientific image processing, linear algebra,[38] statistics,[39] 3D reconstruction and even stock options pricing determination. Over the years, the energy consumption of GPUs has increased and to manage it, several techniques have been proposed.[40]

Nvidia's CUDA platform was the earliest widely adopted programming model for GPU computing. More recently OpenCL has become broadly supported. OpenCL is an open standard defined by the Khronos Group which allows for the development of code for both GPUs and

CPUs with an emphasis on portability.[41] OpenCL solutions are supported by Intel, AMD, Nvidia, and ARM, and according to a recent report by Evan's Data, OpenCL is the GPGPU development platform most widely used by developers in both the US and Asia Pacific.

3.1.6 GPU companies



GPU manufacturers market share

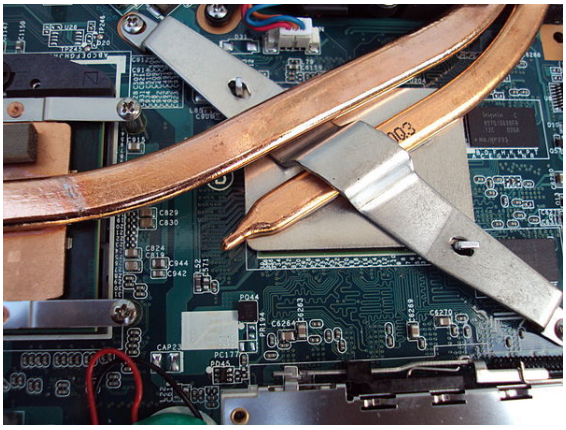
Many companies have produced GPUs under a number of brand names. In 2009, Intel, Nvidia and AMD/ATI were the market share leaders, with 49.4%, 27.8% and 20.6% market share respectively. However, those numbers include Intel's integrated graphics solutions as GPUs. Not counting those numbers, Nvidia and ATI control nearly 100% of the market as of 2008.[42] In addition, S3 Graphics[43] (owned by VIA Technologies) and Matrox[44] produce GPUs.

3.2 Computational functions

Modern GPUs use most of their transistors to do calculations related to 3D computer graphics. They were initially used to accelerate the memory-intensive work of texture mapping and rendering polygons, later adding units to accelerate geometric calculations such as the rotation and translation of vertices into different coordinate systems. Recent developments in GPUs include support for programmable shaders which can manipulate vertices and textures with many of the same operations supported by CPUs, oversampling and interpolation techniques to reduce aliasing, and very high-precision color spaces. Because most of these computations involve matrix and vector operations, engineers and scientists have increasingly studied the use of GPUs for non-graphical calculations; they are especially suited to other embarrassingly parallel problems.

In addition to the 3D hardware, today's GPUs include basic 2D acceleration and framebuffer capabilities (usually with a VGA compatibility mode). Newer cards like AMD/ATI HD5000-HD7000 even lack 2D acceleration; it has to be emulated by 3D hardware.

3.2.1 GPU accelerated video decoding



The ATI HD5470 GPU (above) features UVD 2.1 which enables it to decode AVC and VC-1 video formats

Most GPUs made since 1995 support the YUV color space and hardware overlays, important for digital video playback, and many GPUs made since 2000 also support MPEG primitives such as motion compensation and iDCT. This process of hardware accelerated video decoding, where portions of the video decoding process and video post-processing are offloaded to the GPU hardware, is commonly referred to as "GPU accelerated video decoding", "GPU assisted video decoding", "GPU hardware accelerated video decoding" or "GPU hardware assisted video decoding".

More recent graphics cards even decode high-definition video on the card, offloading the central processing unit. The most common APIs for GPU accelerated video decoding are DxVA for Microsoft Windows operating system and VDPAU, VAAPI, XvMC, and XvBA for Linux-based and UNIX-like operating systems. All except XvMC are capable of decoding videos encoded with MPEG-1, MPEG-2, MPEG-4 ASP (MPEG-4 Part 2), MPEG-4 AVC (H.264 / DivX 6), VC-1, WMV3/WMV9, Xvid / OpenDivX (DivX 4), and DivX 5 codecs, while XvMC is only capable of decoding MPEG-1 and MPEG-2.

Video decoding processes that can be accelerated

The video decoding processes that can be accelerated by today's modern GPU hardware are:

- Motion compensation (mocomp)
- Inverse discrete cosine transform (iDCT)
  - Inverse telecine 3:2 and 2:2 pull-down correction
- Inverse modified discrete cosine transform (iMDCT)

- In-loop deblocking filter
- Intra-frame prediction
- Inverse quantization (IQ)
- Variable-length decoding (VLD), more commonly known as slice-level acceleration
- Spatial-temporal deinterlacing and automatic interlace/progressive source detection
- Bitstream processing (Context-adaptive variable-length coding/Context-adaptive binary arithmetic coding) and perfect pixel positioning.

3.3 GPU forms

3.3.1 Dedicated graphics cards

Main article: Video card

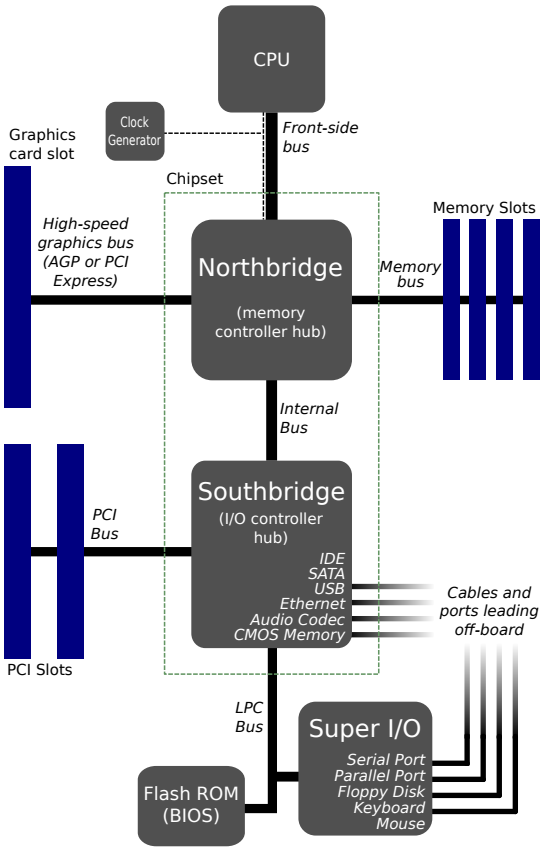
The GPUs of the most powerful class typically interface with the motherboard by means of an expansion slot such as PCI Express (PCIe) or Accelerated Graphics Port (AGP) and can usually be replaced or upgraded with relative ease, assuming the motherboard is capable of supporting the upgrade. A few graphics cards still use Peripheral Component Interconnect (PCI) slots, but their bandwidth is so limited that they are generally used only when a PCIe or AGP slot is not available.

A dedicated GPU is not necessarily removable, nor does it necessarily interface with the motherboard in a standard fashion. The term "dedicated" refers to the fact that dedicated graphics cards have RAM that is dedicated to the card's use, not to the fact that *most* dedicated GPUs are removable. Dedicated GPUs for portable computers are most commonly interfaced through a non-standard and often proprietary slot due to size and weight constraints. Such ports may still be considered PCIe or AGP in terms of their logical host interface, even if they are not physically interchangeable with their counterparts.

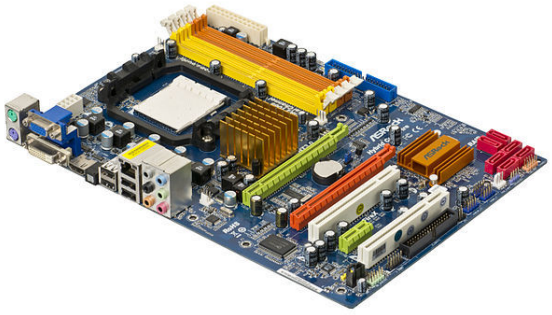
Technologies such as SLI by Nvidia and CrossFire by AMD allow multiple GPUs to draw images simultaneously for a single screen, increasing the processing power available for graphics.

3.3.2 Integrated graphics solutions

Integrated graphics solutions, *shared graphics solutions*, or *integrated graphics processors* (IGP) utilize a portion of a computer's system RAM rather than dedicated graphics memory. IGPs can be integrated onto the motherboard as part of the chipset, or within the same die as CPU (like AMD APU or Intel HD Graphics). On certain motherboards [45] AMD's IGPs can use dedicated sideport memory. This is a separate fixed block of high performance memory that is dedicated for use by the GPU. In early 2007, computers with integrated graphics account for about 90% of all PC shipments.[46] These solutions are less costly to implement than dedicated graphics solutions, but tend to be less capable. Historically, integrated solutions were often considered unfit to play 3D games or run graphically intensive programs but could run less intensive programs such as Adobe Flash. Examples of such IGPs would be offerings from SiS and VIA circa 2004.[47] However, modern integrated graphics processors such as AMD Accelerated Processing Unit and Intel HD Graphics are more than capable of handling 2D graphics or low



The position of an integrated GPU in a northbridge/southbridge system layout



A motherboard with integrated graphics, which has HDMI, VGA and DVI outs.

stress 3D graphics.

As a GPU is extremely memory intensive, an integrated solution may find itself competing for the already relatively slow system RAM with the CPU, as it has minimal or no dedicated video memory. IGPs can have up to 29.856 GB/s of memory bandwidth from system RAM, however graphics cards can enjoy up to 264 GB/s of bandwidth between its RAM and GPU core. This bandwidth is what is referred to as the memory bus and can be performance limiting. Older integrated graphics chipsets lacked hardware transform and lighting, but newer ones include it.[48][49]

3.3.3 Hybrid solutions

This newer class of GPUs competes with integrated graphics in the low-end desktop and notebook markets. The most common implementations of this are ATI's HyperMemory and Nvidia's TurboCache.

Hybrid graphics cards are somewhat more expensive than integrated graphics, but much less expensive than dedicated graphics cards. These share memory with the system and have a small dedicated memory cache, to make up for the high latency of the system RAM. Technologies within PCI Express can make this possible. While these solutions are sometimes advertised as having as much as 768MB of RAM, this refers to how much can be shared with the system memory.

3.3.4 Stream Processing and General Purpose GPUs (GPGPU)

Main articles: GPGPU and Stream processing

It is becoming increasingly common to use a general purpose graphics processing unit (GPGPU) as a modified form of stream processor (or a vector processor), running compute kernels. This concept turns the massive computational power of a modern graphics accelerator's shader pipeline into general-purpose computing power, as opposed to being hard wired solely to do graphical operations. In certain applications requiring massive vector operations, this can yield several orders of magnitude higher performance than a conventional CPU. The two largest discrete (see "Dedicated graphics cards" above) GPU designers, ATI and Nvidia, are beginning to pursue this approach with an array of applications. Both Nvidia and ATI have teamed with Stanford University to create a GPU-based client for the Folding@home distributed computing project, for protein folding calculations. In certain circumstances the GPU calculates forty times faster than the conventional CPUs traditionally used by such applications.[50][51]

GPGPU can be used for many types of embarrassingly parallel tasks including ray tracing. They are generally suited to high-throughput type computations that exhibit data-parallelism to exploit the wide vector width SIMD architecture of the GPU.

Furthermore, GPU-based high performance computers are starting to play a significant role in large-scale modelling. Three of the 10 most powerful supercomputers in the world take advantage of GPU acceleration.[52]

NVIDIA cards support API extensions to the C programming language such as CUDA and OpenCL. CUDA is specifically for NVIDIA GPUs whilst OpenCL is designed to work across a multitude of architectures including GPU, CPU and DSP (using vendor specific SDKs). These technologies allow specified functions (kernels) from a normal C program to run on the GPU's stream processors. This makes C programs capable of taking advantage of a GPU's ability to operate on large buffers in parallel, while still making use of the CPU when appropriate. CUDA is also the first API to allow CPU-based applications to directly access the resources of a GPU for more general purpose computing without the limitations of using a graphics API.

Since 2005 there has been interest in using the performance offered by GPUs for evolutionary computation in general, and for accelerating the fitness evaluation in genetic programming in particular. Most approaches compile linear or tree programs on the host PC and transfer the executable to the GPU to be run. Typically the performance advantage is only obtained by running the single active program simultaneously on many example problems in parallel, using the GPU's SIMD architecture.[53][54] However, substantial acceleration can also be obtained by not compiling the programs, and instead transferring them to the GPU, to be interpreted there.[55][56] Acceleration can then be obtained by either interpreting multiple programs simultaneously, simultaneously running multiple example problems, or combinations of both. A modern GPU (*e.g.* 8800 GTX or later) can readily simultaneously interpret hundreds of thousands of very small programs.

3.3.5 External GPU (eGPU)

An external GPU is a graphics processor located outside of the housing of the computer. External graphics processors are sometimes used with laptop computers. Laptops might have a substantial amount of RAM and a sufficiently powerful central processing unit (CPU), but often lack a powerful graphics processor (and instead have a less powerful but more energy-efficient on-board graphics chip). On-board graphics chips are often not powerful enough for playing the latest games, or for other tasks (video editing, ...).

Therefore, it is desirable to be able to attach a GPU to some external bus of a notebook. PCI Express is the only bus commonly used for this purpose. The port may be, for example, an ExpressCard or mPCIe port (PCIe ×1,

up to 5 or 2.5 Gbit/s respectively) or a Thunderbolt 1, 2, or 3 port (PCIe ×4, up to 10, 20, or 40 Gbit/s respectively). Those ports are only available on certain notebook systems.[57][58]

External GPUs have had little official vendor support. This has not stopped enthusiasts from creating their own DIY eGPU solutions.[59][60]

3.4 Sales

In 2013, 438.3 million GPUs were shipped globally and the forecast for 2014 was 414.2 million.[61]

3.5 See also

- Brute force attack
- Computer graphics
- Computer hardware
- Computer monitor
- Central processing unit
- Vision processing unit
- GPU cache
- Physics processing unit (PPU)
- Ray tracing hardware
- Video card
- Video Display Controller
- Video game console
- Virtualized GPU
- vector processor
- manycore

3.5.1 Hardware

- Comparison of AMD graphics processing units
- Comparison of Nvidia graphics processing units
- Comparison of Intel graphics processing units
- Intel GMA
- Larrabee
- Nvidia PureVideo - the bit-stream technology from Nvidia used in their graphics chips to accelerate video decoding on hardware GPU with DXVA.
- SoC
- UVD (Unified Video Decoder) - is the video decoding bit-stream technology from ATI Technologies to support hardware (GPU) decode with DXVA.

3.5.2 APIs

- OpenGL API
- DirectX Video Acceleration (DxVA) API for Microsoft Windows operating-system.
- Mantle (API)
- Vulkan (API)
- Video Acceleration API (VA API)
- VDPAU (Video Decode and Presentation API for Unix)
- X-Video Bitstream Acceleration (XvBA), the X11 equivalent of DXVA for MPEG-2, H.264, and VC-1
- X-Video Motion Compensation, the X11 equivalent for MPEG-2 video codec only

3.5.3 Applications

- GPU cluster
- Mathematica includes built-in support for CUDA and OpenCL GPU execution
- MATLAB acceleration using the Parallel Computing Toolbox and MATLAB Distributed Computing Server,[62] as well as 3rd party packages like Jacket.
- Molecular modeling on GPU
- Deeplearning4j, open-source, distributed deep learning for Java. Machine vision and textual topic modelling toolkit.

3.6 References

[1] Denny Atkin. “Computer Shopper: The Right GPU for You”. Retrieved 2007-05-15.

[2] “NVIDIA Launches the World’s First Graphics Processing Unit: GeForce 256”. Nvidia. 31 August 1999. Retrieved 28 March 2016.

[3] “Graphics Processing Unit (GPU)". Nvidia. Retrieved 29 March 2016.

[4] Pabst, Thomas (18 July 2002). “ATi Takes Over 3D Technology Leadership With Radeon 9700”. Tom’s Hardware. Retrieved 29 March 2016.

[5] Hague, James (September 10, 2013). "Why Do Dedicated Game Consoles Exist?". *Programming in the 21st Century.*}

[6] "mame/8080bw.c at master · mamedev/mame · GitHub". *GitHub*.

[7] "mame/mw8080bw.c at master · mamedev/mame · GitHub". *GitHub*.

[8] "Arcade/SpaceInvaders – Computer Archeology". *computerarcheology.com*.

[9] "mame/galaxian.c at master · mamedev/mame · GitHub". *GitHub*.

[10] "mame/galaxian.c at master · mamedev/mame · GitHub". *GitHub*.

[11] "MAME - src/mame/drivers/galdrvr.c". *archive.org*. Archived from the original on 3 January 2014.

[12] Springmann, Alessondra. "Atari 2600 Teardown: What?s Inside Your Old Console?". The Washington Post. Retrieved July 14, 2015.

[13] "What are the 6502, ANTIC, CTIA/GTIA, POKEY, and FREDDIE chips?". *Atari8.com*.

[14] Wiegers, Karl E. (April 1984). "Atari Display List Interrupts". *COMPUTE!* (47): 161.

[15] Wiegers, Karl E. (December 1985). "Atari Fine Scrolling". *COMPUTE!* (67): 110.

[16] Riddle, Sean. "Blitter Information".

[17] Wolf, Mark J.P. (June 2012). *Before the Crash: Early Video Game History*. Wayne State University Press. p. 185.

[18] http://nfggames.com/games/x68k/

[19] "museum ~ Sharp X68000". Old-computers.com. Retrieved 2015-01-28.

[20] "Hardcore Gaming 101: Retro Japanese Computers: Gaming's Final Frontier". *hardcoregaming101.net*.

[21] "System 16 - Namco System 21 Hardware (Namco)". *system16.com*.

[22] "System 16 - Taito Air System Hardware (Taito)". *system16.com*.

[23] "S3 Video Boards". *InfoWorld* **14** (20): 62. May 18, 1992. Retrieved July 13, 2015.

[24] "What the numbers mean". *PC Magazine* **12**: 128. 23 February 1993. Retrieved 29 March 2016.

[25] Singer, Graham. "The History of the Modern Graphics Processor". Techspot. Retrieved 29 March 2016.

[26] "System 16 - Namco Magic Edge Hornet Simulator Hardware (Namco)". *system16.com*.

[27] "MAME - src/mame/video/model2.c". *archive.org*. Archived from the original on 4 January 2013.

[28] "System 16 - Sega Model 2 Hardware (Sega)". *system16.com*.

[29] http://www.hotchips.org/wp-content/uploads/hc_archives/hc07/3_Tue/HC7.S5/HC7.5.1.pdf

[30] http://www.fujitsu.com/downloads/MAG/vol33-2/paper08.pdf

[31] "Fujitsu Develops World's First Three Dimensional Geometry Processor". *fujitsu.com*.

[32] xenol. "The Nintendo 64 is one of the greatest gaming devices of all time". *xenol*.

[33] "Mitsubishi's 3DPro/2mp Chipset Sets New Records for Fastest 3D Graphics Accelerator for Windows NT Systems; 3DPro/2mp grabs Viewperf performance lead; other high-end benchmark tests clearly show that 3DPro's performance outdistances all Windows NT competitors.".

[34] Vlask. "VGA Legacy MKIII - Diamond Fire GL 4000 (Mitsubishi 3DPro/2mp)".

[35] 3dfx Glide API

[36] Søren Dreijer. "Bump Mapping Using CG (3rd Edition)". Retrieved 2007-05-30.

[37] "Large-scale deep unsupervised learning using graphics processors". Dl.acm.org. 2009-06-14. doi:10.1145/1553374.1553486. Retrieved 2014-01-21.

[38] "Linear algebra operators for GPU implementation of numerical algorithms", Kruger and Westermann, International Conf. on Computer Graphics and Interactive Techniques, 2005

[39] "ABC-SysBio—approximate Bayesian computation in Python with GPU support", Liepe et al., Bioinformatics, (2010), 26:1797-1799

[40] "A Survey of Methods for Analyzing and Improving GPU Energy Efficiency", Mittal et al., ACM Computing Surveys, 2014.

[41] "OpenCL - The open standard for parallel programming of heterogeneous systems". *khronos.org*.

[42] "GPU sales strong as AMD gains market share". *techreport.com*.

[43] "Products". S3 Graphics. Retrieved 2014-01-21.

[44] "Matrox Graphics - Products - Graphics Cards". Matrox.com. Retrieved 2014-01-21.

[45] "GA-890GPA-UD3H overview".

[46] Gary Key. "AnandTech - µATX Part 2: Intel G33 Performance Review". *anandtech.com*.

[47] Tim Tscheblockov. "Xbit Labs: Roundup of 7 Contemporary Integrated Graphics Chipsets for Socket 478 and Socket A Platforms". Retrieved 2007-06-03.

[48] Bradley Sanford. "Integrated Graphics Solutions for Graphics-Intensive Applications" (PDF). Retrieved 2007-09-02.

[49] Bradley Sanford. "Integrated Graphics Solutions for Graphics-Intensive Applications". Retrieved 2007-09-02.

[50] Darren Murph. "Stanford University tailors Folding@home to GPUs". Retrieved 2007-10-04.

[51] Mike Houston. "Folding@Home - GPGPU". Retrieved 2007-10-04.

[52] "Top500 List - June 2012 | TOP500 Supercomputer Sites". Top500.org. Retrieved 2014-01-21.

[53] John Nickolls. "Stanford Lecture: Scalable Parallel Programming with CUDA on Manycore GPUs".

[54] S Harding and W Banzhaf. "Fast genetic programming on GPUs". Retrieved 2008-05-01.

[55] W Langdon and W Banzhaf. "A SIMD interpreter for Genetic Programming on GPU Graphics Cards". Retrieved 2008-05-01.

[56] V. Garcia and E. Debreuve and M. Barlaud. Fast k nearest neighbor search using GPU. In Proceedings of the CVPR Workshop on Computer Vision on GPU, Anchorage, Alaska, USA, June 2008.

[57] "eGPU candidate system list". *Tech-Inferno Forums*.

[58] Neil Mohr. "How to make an external laptop graphics adaptor". *TechRadar*.

[59] "DIY eGPU on Tablet PC's: experiences, benchmarks, setup, ect...". *tabletpcreview.com*.

[60] "Implementations Hub: TB, EC, mPCIe". *Tech-Inferno Forums*.

[61] "Graphics chips market is showing some life". *TG Daily*. August 20, 2014. Retrieved August 22, 2014.

[62] "MATLAB Adds GPGPU Support". 2010-09-20.

3.7 External links

- NVIDIA - What is GPU computing?
- The *GPU Gems* book series
- - a Graphics Hardware History
- General-Purpose Computation Using Graphics Hardware
- How GPUs work
- GPU Caps Viewer - Video card information utility
- OpenGPU-GPU Architecture(In Chinese)
- ARM Mali GPUs Overview
- GPU Rendering Magazine

Chapter 4

Network processor

A **network processor** is an integrated circuit which has a feature set specifically targeted at the networking application domain.

Network processors are typically software programmable devices and would have generic characteristics similar to general purpose central processing units that are commonly used in many different types of equipment and products.

4.1 History of development

In modern telecommunications networks, information (voice, video, data) is transferred as packet data (termed packet switching) which is in contrast to older telecommunications networks that carried information as analog signals such as in the public switched telephone network (PSTN) or analog TV/Radio networks. The processing of these packets has resulted in the creation of integrated circuits (IC) that are optimised to deal with this form of packet data. Network Processors have specific features or architectures that are provided to enhance and optimise packet processing within these networks.

Network processors have evolved into ICs with specific functions. This evolution has resulted in more complex and more flexible ICs being created. The newer circuits are programmable and thus allow a single hardware IC design to undertake a number of different functions, where the appropriate software is installed.

Network processors are used in the manufacture of many different types of network equipment such as:

- Routers, software routers and switches
- Firewalls
- Session border controllers
- Intrusion detection devices
- Intrusion prevention devices
- Network monitoring systems

4.2 Generic functions

In the generic role as a packet processor, a number of optimised features or functions are typically present in a network processor, these include:

- Pattern matching - the ability to find specific patterns of bits or bytes within packets in a packet stream.
- Key lookup - the ability to quickly undertake a database lookup using a key (typically an address in a packet) to find a result, typically routing information.
- Computation
- Data bitfield manipulation - the ability to change certain data fields contained in the packet as it is being processed.
- Queue management - as packets are received, processed and scheduled to be sent onwards, they are stored in queues.
- Control processing - the micro operations of processing a packet are controlled at a macro level which involves communication and orchestration with other nodes in a system.
- Quick allocation and re-circulation of packet buffers.

4.3 Architectural paradigms

In order to deal with high data-rates, several architectural paradigms are commonly used:

- Pipeline of processors - each stage of the pipeline consisting of a processor performing one of the functions listed above.
- Parallel processing with multiple processors, often including multithreading.
- Specialized microcoded engines to more efficiently accomplish the tasks at hand.

- Recently, multicore architectures are used for higher layer (L4-L7), application processing.

Additionally, traffic management, which is a critical element in L2-L3 network processing and used to be executed by a variety of co-processors, has become an integral part of the network processor architecture, and a substantial part of its silicon area ("real estate") is devoted to the integrated traffic manager[1]

4.4 Applications

Using the generic function of the network processor, a software program implements an application that the network processor executes, resulting in the piece of physical equipment performing a task or providing a service. Some of the applications types typically implemented as software running on network processors are:[2]

- Packet or frame discrimination and forwarding, that is, the basic operation of a router or switch.
- Quality of service (QoS) enforcement - identifying different types or classes of packets and providing preferential treatment for some types or classes of packet at the expense of other types or classes of packet.
- Access Control functions - determining whether a specific packet or stream of packets should be allowed to traverse the piece of network equipment.
- Encryption of data streams - built in hardware-based encryption engines allow individual data flows to be encrypted by the processor.
- TCP offload processing

4.5 See also

- Content processor
- Multi core Processor
- Knowledge based processor
- Active networking
- Computer engineering
- Internet
- List of defunct Network Processor companies
- Network Processing Forum
- Queueing theory

4.5.1 Manufacturers

- Agere Systems
- Alcatel Lucent
- Altera
- AMD
- Analog Devices
- Applied Micro Circuits Corporation
- Atheros
- Bay Microsystems
- Broadcom
- BroadLight
- Cavium Networks
- Conexant
- EZchip
- Freescale
- Hifn
- Infineon
- Intel - Intel has ceased all development in the area of network processors in 2006, but its market share still grew in 2007 and 2008, topping at 38%, due to previously developed products. Netronome currently has the license to develop and manufacture IXP processors with more than 16 cores.[3]
- Lantiq
- LSI Corporation
- Marvell Technology Group
- Mindspeed
- Motorola
- Netronome
- Raza Microelectronics Inc
- SiberCore
- Solidum
- Tilera
- PMC-Sierra
- Xelerated
- Greenfiled
- Ubicom
- Xilinx
- Fortinet

4.6 References

[1] Ran Giladi (2008). *Network processors: architecture, programming, and implementation*. Morgan Kaufmann (Elsevier). ISBN 978-0-12-370891-5.

[2] Douglas E. Comer (2005). *Network Systems Design Using Network Processors: Intel 2XXX Version*. Addison-Wesley. ISBN 978-0-13-187286-8.

[3] *Intel shifts network chip to startup*

4.7 Text and image sources, contributors, and licenses

4.7.1 Text

- **Central processing unit** *Source:* https://en.wikipedia.org/wiki/Central_processing_unit?oldid=719039431 *Contributors:* Chuck Smith, Brion VIBBER, Mav, The Anome, Tarquin, AlexWasFirst, Wayne Hardman, Andre Engels, XJaM, JeLuF, Mudlock, SimonP, BenZin~enwiki, Heron, Chuq, Stevertigo, Frecklefoot, Edward, RTC, Michael Hardy, Booyabazooka, Mahjongg, Nixdorf, Liftarn, SGBailey, Wapcaplet, Ixfd64, TakuyaMurata, 7265, Minesweeper, Ahoerstemeier, Mac, Nanshu, Elano, Muriel Gottrop~enwiki, Angela, Yaronf, Glenn, Nikai, Cimon Avaro, Mxn, Hashar, Emperorbma, Crusadeonilliteracy, Dmsar, Tpbradbury, Morwen, Saltine, Nv8200pa, Taxman, K1Bond007, Tempshill, ZeWrestler, Traroth, Shizhao, Bloodshedder, Ebricca, Secretlondon, Phil Boswell, Chuunen Baka, Donarreiskoffer, Robbot, Murray Langton, Fredrik, Jmabel, Modulatum, Mayooranathan, Merovingian, Sverdrup, Blainster, Jondel, Hadal, JesseW, Wikibot, Mushroom, Vikingstad, Cyrius, Iain.mcclatchie, David Gerard, Wjbeaty, Cedars, Ancheta Wis, Giftlite, Jacoplane, DavidCary, Netoholic, Popup~enwiki, Tom harrison, Lupin, Aphaia, Lmno, Peruvianllama, Everyking, Curps, Guanaco, Solipsist, Fanf, VampWillow, SWAdair, Uzume, Bobblewik, Kudz75, Neilc, Chowbok, Jpkoester1, Quadell, Antandrus, Beland, Pearcej, Rdsmith4, Kesac, Epic matt, Tooki, Joyous!, Positron, Mike Rosoft, PZFUN, Freakofnurture, Archer3, Jiy, Discospinster, Solitude, Rhobite, Jpk, Pixel8, Mjpieters, Berkut, Mani1, Deelkar, Paul August, Dyl, Bender235, ZeroOne, Neko-chan, BACbKA, Ht1848, CanisRufus, Zippedmartin, Diego UFCG~enwiki, Lankiveil, Shanes, Art LaPella, Bookofjude, Bobo192, NetBot, Longhair, Fir0002, Smalljim, Duk, R. S. Shaw, .:Ajvol:., Johnteslade, Elipongo, Matt Britt, Cohesion, Richi, Timl, Giraffedata, Sasquatch, Pearle, Justinc, Nsaa, Espoo, Jumbuck, Alansohn, Liao, Guy Harris, Arthena, Atlant, Wouterstomp, AzaToth, Lectonar, Axl, YDZ, Snowolf, Klaser, Angelic Wraith, Velella, Wtshymanski, Rick Sidwell, Suruena, Bsadowski1, Karderio, Dan100, Richwales, Oleg Alexandrov, Feezo, Snowmanmelting, Iwan rahabok, Morkork, LOL, Nuggetboy, Matey~enwiki, Robert K S, Mjbt, Pol098, Ruud Koot, MONGO, Eleassar777, Damicatz, Terence, GregorB, Wayward, Toussaint, Mandarax, Tslocum, Graham87, Cuchullain, Arunib, MC MasterChef, Kbdank71, Jclemens, Reisio, Sjö, Саша Стефановић, Quale, MarSch, JHMM13, Tawker, HandyAndy, Ligulem, Ttwaring, GeorgeBills, Sango123, Tommy Kronkvist, Naraht, RobertG, Windchaser, Nihiltres, JiFish, Who, Xenobog, LevelCheck, RexNL, Gurch, Alphachimp, SteveBaker, Synchrite, King of Hearts, CJLL Wright, DVdm, Ahpook, Cactus.man, Gwernol, Tone, Elfguy, Wavelength, TexasAndroid, Spacepotato, Sceptre, Adam1213, Phantomsteve, Rowan Moore, Fabartus, John Quincy Adding Machine, Anonymous editor, Noypi380, Stephenb, Shell Kinney, Rsrikanth05, Wimt, NawlinWiki, DragonHawk, Wiki alf, Malmis, Spike Wilbury, NickBush24, Jaxl, Mmccalpin, Vanished user 1029384756, Icelight, RazorICE, J128, Joelr31, Dureo, Banes, Matticus78, Misza13, Killdevil, Tony1, Jeh, Graham Jones, Jeremy Visser, Mcicogni, Mike92591, Avraham, Robost, Fallout boy, Phgao, Zzuuzz, Demus Wiesbaden, Dr.alf, GraemeL, JoanneB, Shawnc, Smurrayinchester, Willtron, Spliffy, Imdaking, Garion96, Rwwww, MansonP, Finell, Soir, Arcadie, AndrewWTaylor, SmackBot, Tuoreco, Bobet, Prodego, KnowledgeOfSelf, Hydrogen Iodide, Pgk, Phaldo, Anastrophe, Jab843, Cessator, Frymaster, Canthusus, Edonovan, Info lover, Ekilfeather, Xaosflux, Yamaguchi??, Gilliam, Ohnoitsjamie, Hmains, Lg1223, Chris the speller, Persian Poet Gal, Jprg1966, Tree Biting Conspiracy, Zelphar, Dlohcierekim's sock, Ikiroid, Whispering, Bowmanjj, ToobMug, Manta7, Harpastum, Jeffreyarcand, Can't sleep, clown will eat me, Mitsuhirato, Frap, Ashawley, Vulcanstar6, OrphanBot, Nixeagle, JonHarder, Yidisheryid, TheKMan, TonySt, RipFire12901, Celarnor, Krich, Gohst, Emre D., Cybercobra, Nakon, AlyM, Fatal-, Slavy13, Acdx, Luigi.a.cruz, Ligulembot, Ck lostsword, Pilotguy, Qmwne235, Zac67, Dave314159, Kuru, Kipala, Edwy, CaptainVindaloo, Goodnightmush, Jcoy, Ben Moore, 16@r, Andypandy.UK, Aceofskies05, George The Dragon, Timmy2, Optakeover, Doczilla, Dhp1080, Ambuj.Saxena, Ryulong, LaMenta3, Phuzion, DwightKingsbury, Levineps, BranStark, Iridescent, Tophtucker, ToastyMallows, Kabu~enwiki, Jacopone~enwiki, Tawkerbot2, Scriptfan, YourUserHere, Adamwisky, Bob121, Electron20, Fvasconcellos, Olie93, JForget, FleetCommand, Unixguy, CmdrObot, Anakata, Sax Russell, Neelix, Sahrin, Akaka~enwiki, Danrok, Steel, Gogo Dodo, ST47, MysticMetal, Chasingsol, Tawkerbot4, DumbBOT, Ameliorate!, Sp, Kozuch, Mtpaley, Omicronpersei8, Davo123, Gimmetrow, S raghu20, Quantyz, Epbr123, Kubanczyk, Kelpherder, Qwyrxian, Newton2, Marek69, TheJosh, James086, Renamed user 1752, Doyley, Java13690, Leon7, Druiloor, Big Bird, Sam42, Dawnseeker2000, Escarbot, Ilion2, Mentifisto, Hmrox, AntiVandalBot, Luna Santin, Widefox, Seaphoto, CZmarlin, Bigtimepeace, Sir Hat, SEG88, Farosdaughter, Malcolm, MECU, DarthShrine, Res2216firestar, Savemeto2, JAnDbot, Tigga, Fiskars007, Barek, MER-C, Instinct, Arch dude, Wiz126, Sitethief, LittleOldMe, Acroterion, Xoneca, Akuyume, Bongwarrior, VoABot II, AuburnPilot, JamesBWatson, Mbarbier, Think outside the box, Omiks3, CTF83!, Dulciana, LaughingMan42, Upholder, ArchStanton69, Marmoulak, 28421u2232nfenfcenc, Allstarecho, P.B. Pilhet, Vssun, DerHexer, GermanX, Gwern, MartinBot, Dinshoupang, Frak, R'n'B, CommonsDelinker, Buletproofbrit, Thesalus, LittleOldMe old, Lilac Soul, J.delanoy, Pharaoh of the Wizards, Trusilver, Grungerz, Hans Dunkelberg, Uncle Dick, Jesant13, MooresLaw, Eliz81, Benscripps, Hakufu Sonsaku, Mikael Häggström, R twell27, AntiSpamBot, Cpusweden, NewEnglandYankee, Shoessss, Camoxide, Littleog, KylieTastic, Cometstyles, Bonadea, Rjclaudio, The dragon123, Useight, Hmdz105, Meiskam, VolkovBot, ABF, Brheed Zonabp84, Gseandiyh79, Jeff G., Indubitably, RightSideNov, 132qwerty, Philip Trueman, DoorsAjar, TXiKiBoT, Oshwah, Zidonuke, Triggerhappy412, Miranda, NPrice, Nxavar, Z.E.R.O., Qxz, Someguy1221, Personline, Oxfordwang, Anna Lincoln, Mandetory, Melsaran, Priver312, Martin451, The Ice Inside, Don4of4, Headbangerbuggy, Lou.weird, Jackfork, LeaveSleaves, Monkeynoze, Sychen, Wiae, Srce, Andy Dingley, Michelle192837, Haseo9999, Wasted Sapience, WJetChao, Synthebot, Enviroboy, Turgan, Nagy, Dogah, SieBot, Gmb1994, Fnagaton, Sonicology, AlphaPyro, Santhosh.thottingal, Krawi, Smsarmad, Yintan, Keilana, Bentogoa, Breawycker, Flyer22 Reborn, Tiptoety, Bobanater, Wilson44691, Paolo.dL, Oxymoron83, Bichito, Jdaloner, Lightmouse, Techman224, Hobartimus, Ks0stm, RyanParis, Dillard421, Sean.hoyland, Mygerardromance, WikiLaurent, Pinkadelica, Denisarona, Jbray3179, WikipedianMarlith, Seanm924, Loren.wilton, Martarius, ClueBot, Ano onymis, Snigbrook, Foxj, Wikievil666, The Thing That Should Not Be, Rilak, Jan1nad, Knepflerle, Pheeror, Arakunem, Starcraft.nut, Myhellhereorunder, Yabeeno, Matt 118118, Blanchardb, Ridge Runner, Rprpr, Rockfang, Pointillist, Sensiblemayank, Hallo990, Catfish Jim and the soapdish, Excirial, Goodone121, Tomeasy, Koishii1521, Alejandrocaro35, Tyler, Jjtennisman, Ben ben ben ben ben jerry, Dekisugi, XTerminator2000, Netanel h, Deerstop, Aitias, Flipper344, Versus22, Lambtron, SoxBot III, Ginbot86, XLinkBot, Hotcrocodile, Fastily, Gwandoya, Stickee, Rror, Feinoha, Little Mountain 5, Rreagan007, SilvonenBot, NellieBly, Galzigler, Badgernet, Jd027, Torahjerus14, JinJian, Zodon, Lanky217, Dsimic, Tim1980tim, Addbot, Winstonliang6758, Pyfan, Raghavkvp, Mortense, Sdfsakjdhfuioaheif283, Manuel Trujillo Berges, Landon1980, Non-dropframe, Captaintucker, LHvU, Peti1212, Turnerj, Ronhjones, Fieldday-sunday, Ironholds, D0762, Darklightning1, Scientus, CanadianLinuxUser, MrOllie, Glane23, Matthewirwin28693, WikiDegausser, Jasper Deng, Beastathon, Brainmachine, Numbo3-bot, Bwrs, DNA to RNA, Tide rolls, Krano, Teles, Gail, Zorrobot, Jarble, Ettrig, Luckas-bot, Yobot, Mcdennis13, Fraggle81, Phatstakks, Pikachu~enwiki, ArchonMagnus, Mmxx, Knockwood, KamikazeBot, Timir Saxa, AnakngAraw, Radiopathy, Deicool, AnomieBOT, Rubinbot, Jim1138, IRP, Galoubet, Piano non troppo, AdjustShift, Fahadsadah, Kingpin13, RandomAct, Flewis, Materialscientist, The High Fin Sperm Whale, Citation bot, Juhuang, X360Silent, Georgeb92, Wx4sno, Milf&cookies, GB fan, JohnFromPinckney, 04satvinderbi, Xqbot, 4I7.4I7, Cureden, The sock that should not be, Capricorn42, 4twenty42o, Knowitall44, Jsharpminor, Blargarg, Coretheapple, GrouchoBot, Nayvik, Solphusion~enwiki, Wizardist, RibotBOT, Mpgenius, Genius1789, SCARECROW, Luciandrei, Sicklight, Jarred.rop1, Bstarynk, Full-hyperion, Shadowjams, Dougofborg, Cekli829, GT5162, FrescoBot, Bighead01753, LucienBOT, Rome109, X5UPR3ME STEV3x, VI, Jamesooders, Citation bot

1, Maggyero, Wdfowty, Pooosihole, I dream of horses, FMAlchemist36, HRoestBot, Edderso, A412, Coekon, Calmer Waters, Tinton5, Hamtechperson, BRUTE, RedBot, SpaceFlight89, Footwarrior, Lineslarge, Rzęsor, Pranayrocks23, WardMuylaert, RMartin-2, ئاراس نورى, Peterlunde, TheStrayCat, Ravenperch, Vrenator, Defender of torch, Momma69, Aiken drum, Bitolado, Jeffrd10, Specs112, Diannaa, Tbhotch, Reach Out to the Truth, Jesse V., Minimac, Hornlitz, 11james22, DARTH SIDIOUS 2, Onel5969, Dmytheus, TjBot, Ihateblazing, FNQ, Dhiraj1984, Noommos, Lauri.pirttiaho, Instigate cjsc (Narine), Thisisafakeaccountfordeletingpages, EmausBot, Orphan Wiki, WikitanvirBot, Mynameiswa, BillyPreset, Akjar13, Racerx11, Wikipelli, Mz7, ZéroBot, John Cline, Cogiati, Daonguyen95, Ida Shaw, Fæ, Imperial Monarch, Pololei, Espin2, Fred Gandt, Stas3717, Wayne Slam, Ferry24.Milan, Arman Cagle, LordJeff, Shrikanthv, GeorgeBarnick, L Kensington, Donner60, Wikiloop, Orange Suede Sofa, Ipsign, Jrstern29, LikeLakers2, Erin2003, 28bot, ClueBot NG, Bionik276, TomBridle, Satellizer, Andreas.Persson, Movses-bot, Millermk, Rajayush78, Cntras, O.Koslowski, Kasirbot, Masssly, Widr, Greg.Kerr01, Ashish Gaikwad, Jorgenev, Mapolat, Adam.Amory.97, Yucel114, Hazal018, HMSSolent, Wbm1058, Nashhinton, Dogan900, Wiki13, MusikAnimal, Amp71, IraChesterfield, Jonas weepel, Kinnngg, Rm1271, Dauntless28, Anchit singla, Camomen3000, Glacialfox, Klilidiplomus, MantridFreeman, Mikepabell, Chip123456, Fylbecatulous, BattyBot, Justincheng12345-bot, WhiteNebula, Computerwoman417, Benjaminjkim, CKsquid, YFdyh-bot, Ekonomka, Modwizcode, MadGuy7023, Gdrg22, Tow, Sniper-ass, Dexbot, KuraAbyss, Taylorclem, ZaferXYZ, Frosty, Graphium, Jamesx12345, Trollilols, Botusharov, EdwardJK, Jazzee8845, Andypandyjay, Reatlas, Greengreengreenred, Sancho .308, Thatginernonce, Mndunne, Bobobob1231, DavidLeighEllis, Edwinblack1, Comp.arch, Ugog Nizdast, Fwiki225, Riehutunut, AddWittyNameHere, OccultZone, JammedWords, Lopsidedtriangle, Djgolam, JaconaFrere, FlashDave, UltraFireFX, Sonyoo1, GeorgeAhad, TKer193, Mr. Smart LION, Monkbot, Rudyjuliyanto, MatthewBuchwalder, Jim Carter, Trax support, Sperkowsky, Keonnikpour, Vinaya.mente, QueenFan, TerryAlex, Siraj Khurshied, Anish057, BlackSeaSailor, ChamithN, Crystallizedcarbon, Liance, Darknesssnow, KOool23456, Joeseph kake, FourViolas, Sivaprasadmsr, ComsciStudent, Israelg99, TurkeyWriter77, Benji877, TehBacon, Lumpy247, Troll42069, Thefreeencyclopedia1231, Robolamiah, Superbug1000, Dou14208038, Territory war 3 pwner, Ruftas, Jakobletroll, Stealthyhyena, VETICTClass, England12345, Shadowhunter46, Chazer23, Ttt74, Cpu, calor and Anonymous: 1393

- **Digital signal processor** *Source:* https://en.wikipedia.org/wiki/Digital_signal_processor?oldid=716649648 *Contributors:* Mav, SimonP, Maury Markowitz, Michael Hardy, Nixdorf, Nickrusnov, Oyd11, Minesweeper, Typhoon, Sphl~enwiki, Whkoh, Tristanb, Jrousseau, Dysprosia, Mrand, Wernher, Bevo, Topbanana, Murray Langton, RedWolf, Jondel, SpellBott, Giftlite, DavidCary, Mcapdevila, Chowbok, Pyle, Lockeownzj00, Sam Hocevar, Biot, Abdull, Flex, Alistair1978, Harriv, Violetriga, Matt Britt, Hooperbloob, Guy Harris, Cburnett, Suruena, GOKULAK, Nuno Tavares, Ruud Koot, Jeff3000, AlbertCahalan~enwiki, Meneth, MarkusHagenlocher, Toussaint, Palica, Zephyrxero, Kbdank71, Ketiltrout, Joe Decker, Alll~enwiki, Maxim Razin, Arnero, Webshared, Jidan, Chobot, Bgwhite, YurikBot, Borgx, Dmccarty, Toffile, Gaius Cornelius, TheMandarin, Darksidex, Kkmurray, Ninly, LeonardoRob0t, Curpsbot-unicodify, RichardYoung, Henriok, DomQ, Ohnoitsjamie, Oli Filth, Firetrap9254, Zvar, A5b, The undertow, Db1145, JimisDose, 16@r, Saerdnaer, Dicklyon, TeryKing, Kvng, Martin Kozák, Requestion, AstroPig7, Thijs!bot, WillMak050389, Electron9, Hcobb, Jauricchio, Jamesgor13579, Cevadsp, JAnDbot, Jahoe, Rivertorch, Soulbot, Yewyew66, GermanX, Jvienneau, Glrx, RockMFR, Jcurie, Harobikes34, STBotD, RobOnKnowledge, PNG crusade bot, Ultim, Starrymessenger, JhsBot, Broadbot, Andy Dingley, Haseo9999, SieBot, Sonicology, AlphaPyro, Jerryobject, Travelingseth, GAMER 20999, Dspanalyst, OSP Editor, Martarius, ClueBot, Rilak, Niceguyedc, DragonBot, Alexbot, LJanardhan, Jotterbot, Sebastien.m, GlasGhost, Pantech solutions, Johnuniq, Semitransgenic, Skarebo, Dsimic, Addbot, Mortense, Anschel, MrVanBot, Lightbot, Luckas-bot, Themfromspace, Lehuma, AnomieBOT, Adeliine, Thisara.d.m, Witguiota, Dspmandavid, RibotBOT, Kyng, Insomnia64, Prari, FrescoBot, BenzolBot, Jonathandeamer, Jschnur, RedBot, Serols, SpaceFlight89, Overjive, RjwilmsiBot, Bento00, 10 goal payne, DSP-user, EmausBot, Orphan Wiki, Sumudufdo, GoingBatty, Nrobbo, Mulletsrokkify, 28bot, ClueBot NG, Jaanus.kalde, Matarasephotos, BG19bot, Minsbot, Wikpoint, ChrisGualtieri, Tagremover, Poolborges, Frosty, Nutaq, Pkunk, Spencer.mccormick, Melonkelon, Yardimsever, ArmbrustBot, YiFeiBot, ScotXW, KasparBot, Fmadd and Anonymous: 186
- **Graphics processing unit** *Source:* https://en.wikipedia.org/wiki/Graphics_processing_unit?oldid=719490379 *Contributors:* Taw, Wayne Hardman, Heron, Edward, DopefishJustin, Mahjongg, Nixdorf, Karada, Egil, Andres, Harvester, Lee Cremeans, Furrykef, Tempshill, Wernher, Thue, Topbanana, Stormie, Optim, Lumos3, Robbot, Chealer, Vespristiano, Playwrite, Academic Challenger, Tea2min, Alan Liefting, Alf Boggis, Paul Pogonyshev, Everyking, Alison, Lurker, DJSupreme23, Gracefool, Rchandra, AlistairMcMillan, Egomaniac, Khalid hassani, Gadfium, Utcursch, Pgan002, Aughtandzero, Quadell, Lockeownzj00, Beland, MFNickster, Simoneau, Trilobite, Imroy, Pixel8, AlexKepler, Berkut, Alistair1978, Pavel Vozenilek, Gronky, Indrian, Evice, Billlion, TOR, CanisRufus, RoyBoy, Dgpop, Drhex, Polluks, Matt Britt, Richi, Kjkolb, Markpapadakis, Kaf, Varuna, Murphykieran, Mc6809e, Hohum, Angelic Wraith, Velella, Suruena, Sciurinæ, Bjorke, Freyr, Marasmusine, Kelly Martin, Woohookitty, Jannex, Ae-a, Macronyx~enwiki, Tabletop, SCEhardt, Isnow, M412k, Toussaint, Kbdank71, Josh Parris, Tbird20d, Sdornan, Sango123, StuartBrady, FlaBot, Mirror Vax, Arnero, Viznut, Chobot, ShadowHntr, YurikBot, Jtbandes, Locke411, Yyy, ALoopingIcon, Virek, RicReis, Qviri, Panscient, Zephalis, Mike92591, MaxDZ8, Wknight94, Delirium of disorder, Arthur Rubin, D'Agosta, E Wing, Red Jay, David Biddulph, Mikkow, Nekura, Veinor, FearTec, SmackBot, Colinstu, AFBorchert, Bigbluefish, Unyoyega, Jagged 85, Renku, KVDP, Jrockley, Eskimbot, Scott Paeth, Jpvinall, Gilliam, Bluebot, TimBentley, GoldDragon, QTCaptain, Thumperward, Jerome Charles Potts, Octahedron80, Anabus, Tsca.bot, Can't sleep, clown will eat me, Harumphy, Frap, JonHarder, Ruw1090, Easwarno1, Theonlyedge, Cybercobra, Melter, Nakon, PointyOintment, Trieste, HarisM, Nitro912gr~enwiki, Swaaye, Salamurai, HeroTsai, Soumya92, Disavian, Wibbble, Joffeloff, Codepro, Aleenf1, Vuurmeester, Phranq, Cxk271, Sjf, Hu12, Stargaming~enwiki, Agelu, ScottHolden, Stoakron97, Aeons, Tawkerbot2, Jafet, Braddodson, SkyWalker, Xcentaur, Zarex, Mattdj, Nczempin, Jsmaye, Jesse Viviano, Shandris, Lazulilasher, Sahrin, Pi Guy 31415, Phatom87, Danrok, JJC1138, Gogo Dodo, Scissorhands1203, Soetermans, Mr. XYZ, Tawkerbot4, Bitsmart, Thijs!bot, Wermlandsdata, Mentifisto, Eberhart, AntiVandalBot, Konman72, Gioto, SEG88, Flex Flint, Johan.Seland, Skarkkai, Serpent's Choice, JAnDbot, MER-C, Jdevesa, Arch dude, Kremerica, RubyQ, Vidsi, AndriusG, RBBrittain, Gbrose85, Michaelothomas, Nikevich, I JethroBT, Marmoulak, David Eppstein, Crazyideas21, Frampis, El Krem, UnfriendlyFire, Trusader, R'n'B, J.delanoy, Pharaoh of the Wizards, ChrisfromHouston, Maurice Carbonaro, Jesant13, Smite-Meister, Gzkn, Xbspiro, M-le-mot-dit, Urzadek, Jo7hs2, EconomistBR, Sugarbat, Spiesr, Canadianbob, Martial75, Lights, VolkovBot, MrRK, TXiKiBoT, Oshwah, Like.liberation, Tr-the-maniac, Tandral, Cody-7, Broadbot, Haseo9999, Squalk25, AlleborgoBot, Glitchrf, SieBot, 4wajzkd02, Yulu, Garde, Djayjp, Flyer22 Reborn, Nopetro, Jimthing, Oxymoron83, Lightmouse, Earthere, Twsl, Pinkadelica, Gillwill, WikipedianMarlith, Accessory, ClueBot, The Thing That Should Not Be, Placi1982, Rilak, Nnemo, Dpmuk, Jappalang, Hexmaster, Niceguyedc, Alexbot, Socrates2008, Technobadger, Arjayay, Jotterbot, Ark25, Muro Bot, Vapourmile, GlasGhost, Andy16666, Socks 01, Tigeron, 5900FX, GeoffMacartney, DumZiBoT, Rreagan007, Salam32, Frood, JeGX, Noctibus, Eleven even, Zodon, Veritysense, NonNobisSolum, Dsimic, Osarius, Addbot, Willking1979, Ronhjones, MrOllie, Download, LaaknorBot, Favonian, Aunva6, Peti610botH, Fiftyquid, Jarble, Xowets, Ben Ben, Legobot, Publicly Visible, Luckas-bot, Yobot, Ptbotgourou, Becky Sayles, GateKeeper, Sg227, 4thotaku, AnomieBOT, Götz, Masterofwiki666, Galoubet, Materialscientist, Clark89, LilHelpa, JanEnEm, PavelSolin, Xqbot, Holden15, Erud, Victorbabkov, CoolingGibbon, P99am, Braxtonw1, J04n, Winstonliang, =Josh.Harris, Robert SkyBot, FrescoBot, IvarTJ, Umawera, Math1337, Jusses2, Vincentfpgarcia, RedBot, Akkida, Rzęsor, Hitachi-Train, Yogi m, Ale And Quail, Ravenperch, Jesse V., John Buchan, DARTH SIDIOUS 2, Onel5969, Dewritech, Dcirovic, Serketan, Cogiati, Vitkovskiy Roman, Midas02, Handheldpenguin, Veikk0.ma,

Romdanen, Tomy9510, Topeil, Bomazi, Evan-Amos, Des3dhj, ClueBot NG, Matthiaspaul, Sand3r., Dholcombe, Widr, Tijok, Marcus-British, Helpful Pixie Bot, Largecrashman, Wbm1058, KLBot2, Aayush.nitb, Kangaroopower, Sqzx, MusikAnimal, Joydeep, Diculous, Alanau8605, Isenherz, Tagremover, Comatmebro, Dymatic, Stocbuster, Codename Lisa, SoledadKabocha, Webclient101, Makecat-bot, Ckoerner, Nonnompow, Andrei.gheorghe, Frosty, Calinou1, OSXiOSMacFan, EdwardJK, Jmankovecky, Reatlas, Mahbubur-r-aaman, Hallowin, Eyesnore, Nigma2k, Dannyniu, CrystalCanine, Comp.arch, Papagao, Sibekoe, Jdog147123, Acc12345acc, ScotXW, UltraFireFX, Kral Petr, Mansoor-siamak, ChamithN, RogerFareham, Sizeofint, IntriguingStar, Newwikieditor678, KasparBot, LiamMcS, JJMC89, BBQ, Pateljay43, Sing0512, Fmadd and Anonymous: 486

- **Network processor** *Source:* https://en.wikipedia.org/wiki/Network_processor?oldid=716179570 *Contributors:* SebastianHelm, Paul August, Dyl, Brholden, Woohookitty, Ruud Koot, Kglavin, BD2412, FlaBot, RobyWayne, Alphachimp, Spasemunki, Gaius Cornelius, Ksyrie, Falcon9x5, Scope creep, Museo8bits, David Biddulph, SmackBot, Henriok, Chris the speller, Bluebot, Frap, JonHarder, Ryan Roos, Pramod.s, ArglebargleIV, ManiacK, Bushsf, JHunterJ, Dicklyon, CmdrObot, Squater, Dougher, Raanoo, Magioladitis, Meredyth, NotACow, Gwern, Verdatum, Sweetness46, Oshwah, Erahgl, SieBot, Frappucino, Navanee2u, Freebullets, Skua~enwiki, Joel Saks, Addbot, Ramu50, MrOllie, Fraggle81, SabbaZ, Bquigman, Jignesh.4soni, GB fan, Omnipaedista, Madison Alex, Amarnathshanbhag, Super48paul, Tuo4004, AvicAWB, Sahimrobot, Ipsign, ClueBot NG, Frietjes, Rezabot, Helpful Pixie Bot, BG19bot, Semiexpert, KH-1, Hashken, KasparBot, Assain22 and Anonymous: 52

4.7.2 Images

- **File:6600GT_GPU.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/44/6600GT_GPU.jpg *License:* CC-BY-SA-3.0 *Contributors:* Own work *Original artist:* Berkut
- **File:80486dx2-large.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/02/80486dx2-large.jpg *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:A790GXH-128M-Motherboard.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/0c/A790GXH-128M-Motherboard.jpg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Evan-Amos
- **File:ABasicComputer.gif** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/d8/ABasicComputer.gif *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* Lambtron
- **File:ALU_block.gif** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/0f/ALU_block.gif *License:* CC BY-SA 4.0 *Contributors:* Own work *Original artist:* Lambtron
- **File:AMD_HD5470_GPU.JPG** *Source:* https://upload.wikimedia.org/wikipedia/en/8/88/AMD_HD5470_GPU.JPG *License:* CC0 *Contributors:*

  Self created

  Original artist:

  highwycombe (talk)
- **File:Binary_Forty.PNG** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/25/Binary_Forty.PNG *License:* CC0 *Contributors:* Own work *Original artist:* P Astbury
- **File:Board_with_SPARC64_VIIIfx_processors_on_display_in_Fujitsu_HQ.JPG** *Source:* https://upload.wikimedia.org/wikipedia/commons/c/c8/Board_with_SPARC64_VIIIfx_processors_on_display_in_Fujitsu_HQ.JPG *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Tonusamuel
- **File:Commons-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/4/4a/Commons-logo.svg *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:Computer-aj_aj_ashton_01.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/d7/Desktop_computer_clipart_-_Yellow_theme.svg *License:* CC0 *Contributors:* https://openclipart.org/detail/105871/computeraj-aj-ashton-01 *Original artist:* AJ from openclipart.org
- **File:DIAMONDSTEALTH3D2000-top.JPG** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f8/DIAMONDSTEALTH3D2000-top.JPG *License:* Public domain *Contributors:* Transferred from ar.wikipedia to Commons. *Original artist:* The original uploader was Salam32 at Arabic Wikipedia
- **File:DSP_block_diagram.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bc/DSP_block_diagram.svg *License:* CC-BY-SA-3.0 *Contributors:* This vector image was created with Inkscape. *Original artist:* en:User:Cburnett
- **File:Dsp_chip.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a9/Dsp_chip.jpg *License:* CC BY 3.0 *Contributors:* Own work *Original artist:* Matarese photos
- **File:Dstealth32.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/22/Dstealth32.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Swaaye at English Wikipedia
- **File:EBIntel_Corei5.JPG** *Source:* https://upload.wikimedia.org/wikipedia/commons/5/52/EBIntel_Corei5.JPG *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* highwycombe (talk)
- **File:Edvac.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/1/17/Edvac.jpg *License:* Public domain *Contributors:* Photo located at: http://ftp.arl.mil/ftp/historic-computers/ *Original artist:* ?
- **File:Fivestagespipeline.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/21/Fivestagespipeline.png *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:IBM_PPC604e_200.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/a/a7/IBM_PPC604e_200.jpg *License:* Public domain *Contributors:* Own work *Original artist:* Henrik Wannheden
- **File:Intel_80486DX2_bottom.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/e/e7/Intel_80486DX2_bottom.jpg *License:* CC BY-SA 2.0 *Contributors:* ? *Original artist:* ?
- **File:Intel_80486DX2_top.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/d/dc/Intel_80486DX2_top.jpg *License:* CC BY-SA 2.0 *Contributors:* ? *Original artist:* ?

- **File:Marktanteil_GPU-Hersteller.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/20/Marktanteil_GPU-Hersteller.png *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Mark H.
- **File:Motherboard_diagram.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/bd/Motherboard_diagram.svg *License:* CC-BY-SA-3.0 *Contributors:* This file was derived from: Diagramme carte mère.png
Original artist: **Original:** Gribeco at French Wikipedia
Derivative work: Moxfyre at English Wikipedia
- **File:Nopipeline.png** *Source:* https://upload.wikimedia.org/wikipedia/commons/2/2c/Nopipeline.png *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?
- **File:PDP-8i_cpu.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/0/03/PDP-8i_cpu.jpg *License:* Public domain *Contributors:* http://en.wikipedia.org/wiki/Image:PDP-8i_cpu.jpg *Original artist:* Robert Krten
- **File:Question_book-new.svg** *Source:* https://upload.wikimedia.org/wikipedia/en/9/99/Question_book-new.svg *License:* Cc-by-sa-3.0 *Contributors:*
Created from scratch in Adobe Illustrator. Based on Image:Question book.png created by User:Equazcion *Original artist:*
Tkgd2007
- **File:Sound-icon.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/47/Sound-icon.svg *License:* LGPL *Contributors:* Derivative work from Silsor's versio *Original artist:* Crystal SVG icon set
- **File:Superscalarpipeline.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/46/Superscalarpipeline.svg *License:* CC BY-SA 3.0 *Contributors:* Own work *Original artist:* Amit6, original version (File:Superscalarpipeline.png) by User:Poil
- **File:Telecom-icon.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/4/4e/Telecom-icon.svg *License:* Public domain *Contributors:* ? *Original artist:* ?
- **File:Voodoo3-2000AGP.jpg** *Source:* https://upload.wikimedia.org/wikipedia/commons/8/88/Voodoo3-2000AGP.jpg *License:* CC-BY-SA-3.0 *Contributors:* Transferred from en.wikipedia to Commons by JohnnyMrNinja using CommonsHelper. *Original artist:* Swaaye at English Wikipedia
- **File:Wikiversity-logo.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/9/91/Wikiversity-logo.svg *License:* CC BY-SA 3.0 *Contributors:* Snorky (optimized and cleaned up by verdy_p) *Original artist:* Snorky (optimized and cleaned up by verdy_p)

4.7.3 Content license

- Creative Commons Attribution-Share Alike 3.0