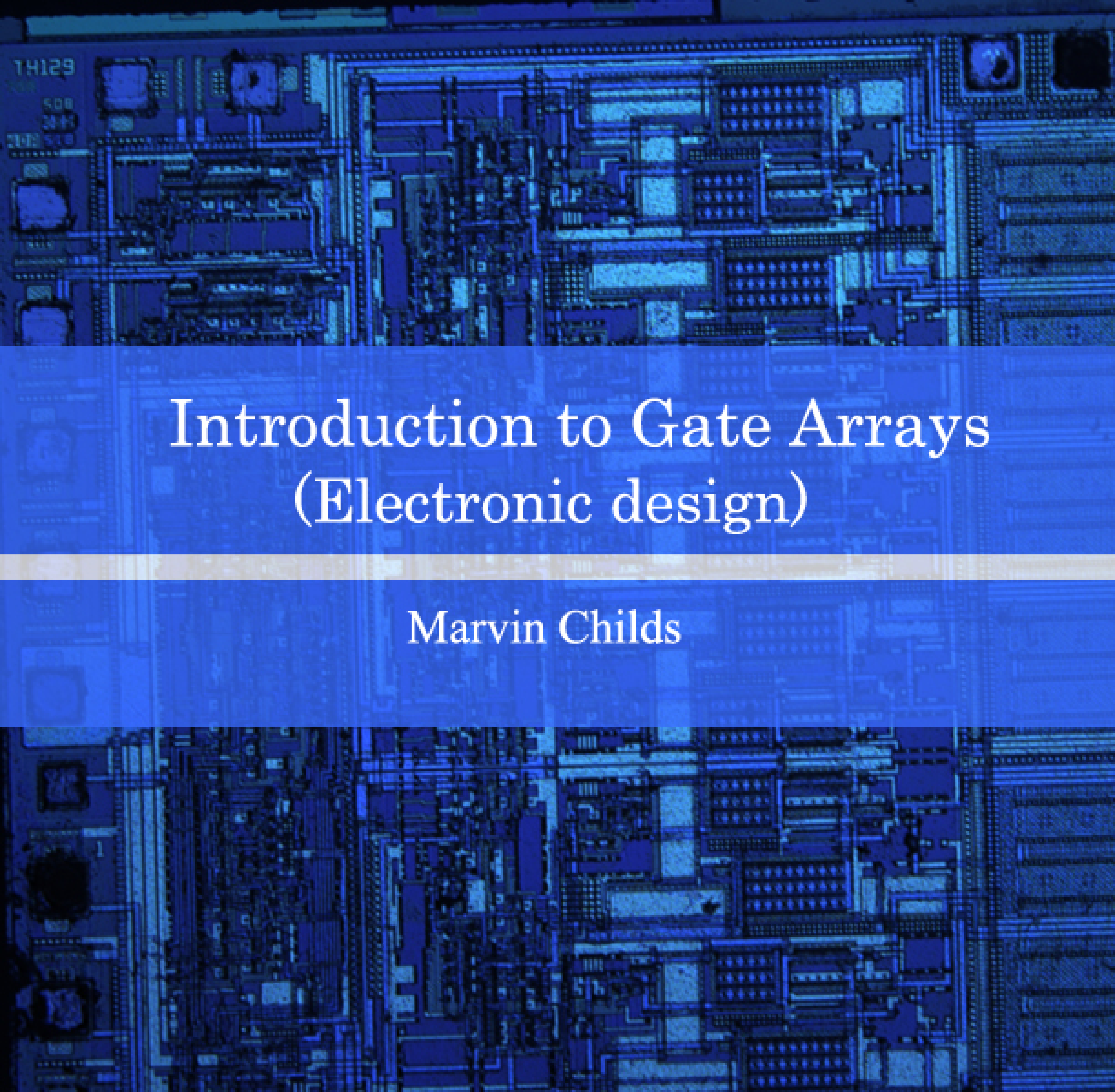


Introduction to Gate Arrays (Electronic design)

Marvin Childs

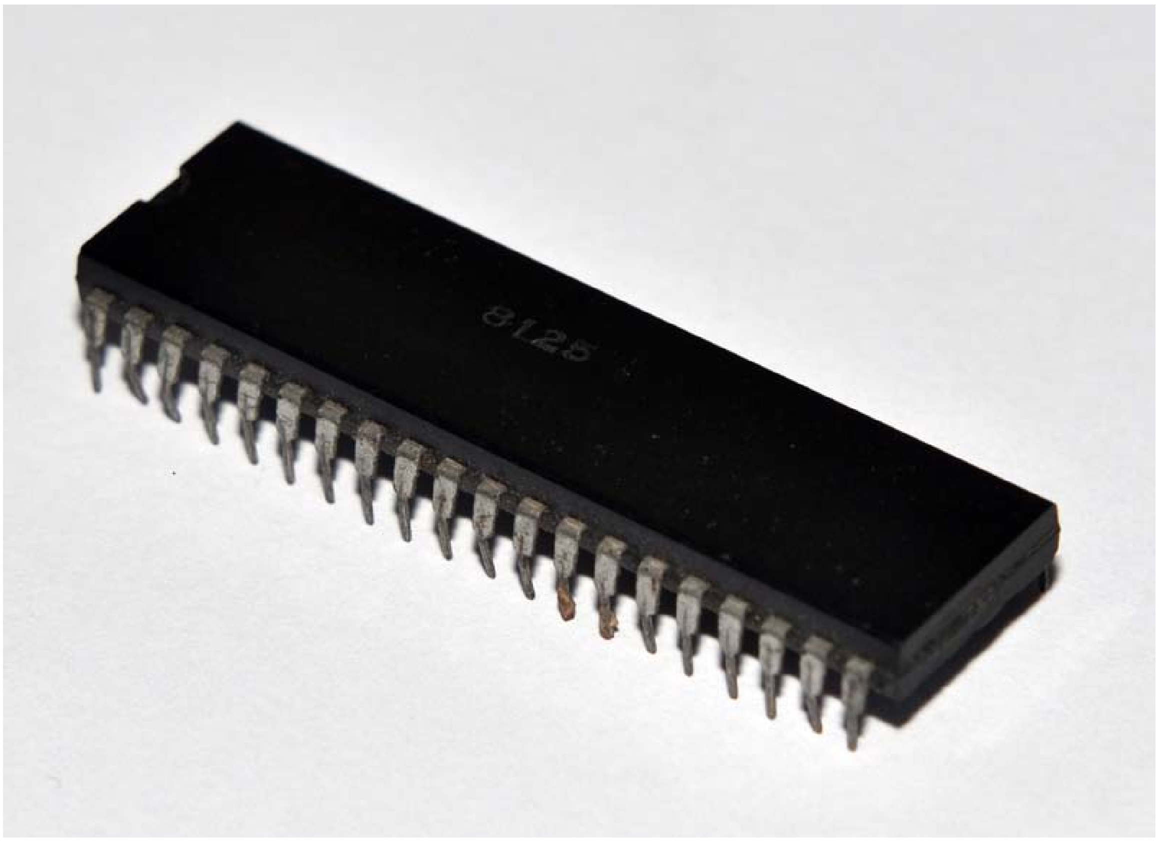
First Edition, 2012

ISBN 978-81-323-3076-9

Published by:
Research World
4735/22 Prakashdeep Bldg,
Ansari Road, Darya Ganj,
Delhi - 110002
Email: info@wtbooks.com

Table of Contents

Introduction



Sinclair ZX81 ULA

A **gate array** or **uncommitted logic array** (**ULA**) is an approach to the design and manufacture of application-specific integrated circuits (ASICs). A gate array circuit is a prefabricated silicon chip circuit with no particular function in which transistors, standard NAND or NOR logic gates, and other active devices are placed at regular predefined positions and manufactured on a wafer, usually called a *master slice*. Creation of a circuit with a specified function is accomplished by adding a final surface layer or layers of metal interconnects to the chips on the master slice late in the manufacturing process, joining these elements to allow the function of the chip to be customized as desired. This layer is analogous to the copper layer(s) of a printed circuit board (PCB).

Gate array master slices are usually prefabricated and stockpiled in large quantities regardless of customer orders. The design and fabrication according to the individual customer specifications may be finished in a shorter time compared with standard cell or full custom design. The gate array approach reduces the mask costs since fewer custom masks need to be produced. In addition manufacturing test tooling lead time and costs are reduced since the same test fixtures may be used for all gate array products manufactured on the same die size. Gate arrays were the predecessor of the more advanced structured ASIC; unlike gate arrays, structured ASICs tend to include predefined or configurable memories and/or analog blocks. Structured ASICs are still sold by companies such as ChipX, Inc.

An application circuit must be built on a gate array that has enough gates, wiring and I/O pins. Since requirements vary, gate arrays usually come in families, with larger members having more of all resources, but correspondingly more expensive. While the designer can fairly easily count how many gates and I/Os pins are needed, the amount of routing tracks needed may vary considerably even among designs with the same amount of logic. (For example, a crossbar switch requires much more routing than a systolic array with the same gate count.) Since unused routing tracks increase the cost (and decrease the performance) of the part without providing any benefit, gate array manufacturers try to provide just enough tracks so that most designs that will fit in terms of gates and I/O pins can be routed. This is determined by estimates such as those derived from Rent's rule or by experiments with existing designs.

The main drawbacks of gate arrays are their somewhat lower density and performance compared with other approaches to ASIC design. However this style is often a viable approach for low production volumes.

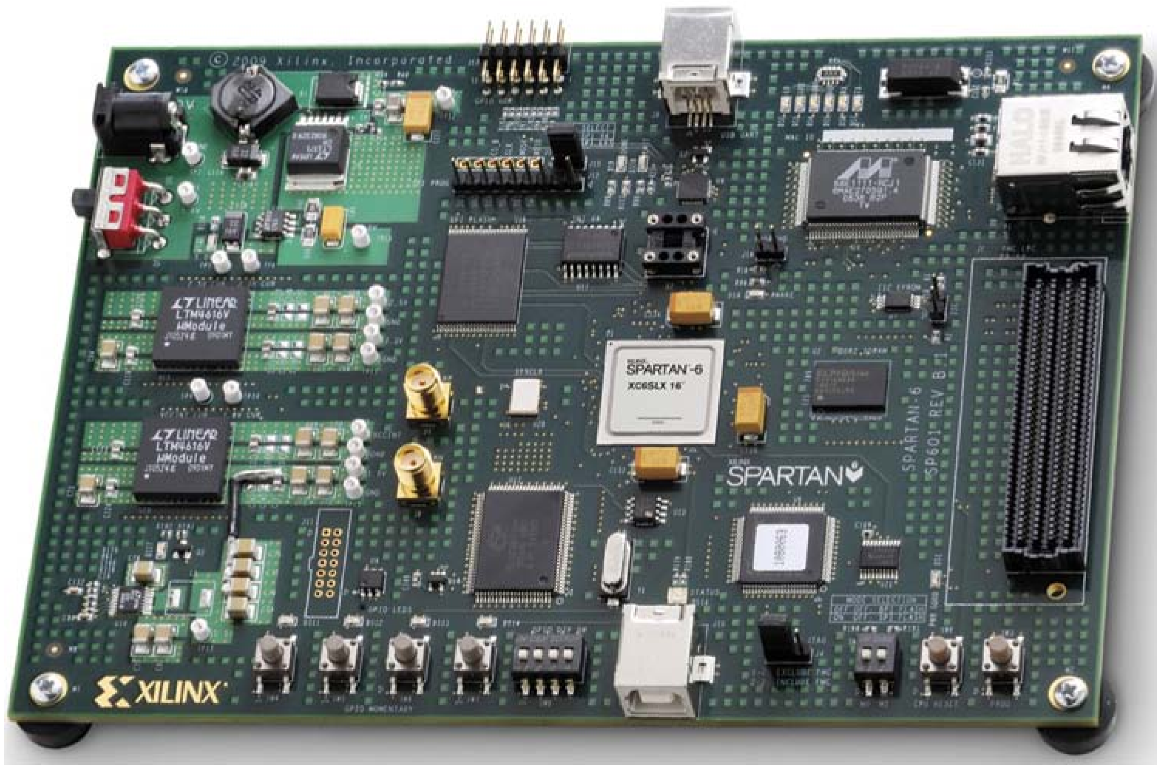
Sinclair Research ported an enhanced ZX80 design to a ULA chip for the ZX81, and later used a ULA in the ZX Spectrum. A compatible chip was made in Russia as T34VG1. Acorn Computers used several ULA chips in the BBC Micro, and later managed to compress almost all of that machine's logic into a single ULA for the Acorn Electron. Many other manufacturers from the time of the home computer boom period used ULAs in their machines. Ferranti in the UK pioneered ULA technology, then later abandoned this lead in semi-custom chips. The IBM PC took over much of the personal computer market, and the sales volumes made full-custom chips more economical.

Designers still wished for a way to create their own complex chips without the expense of full-custom design, and eventually this wish was granted with the arrival of the field-programmable gate array (FPGA), complex programmable logic device (CPLD), and structured ASIC. Whereas a ULA required a semiconductor wafer foundry to deposit and etch the interconnections, the FPGA and CPLD had programmable interconnections.

Chapter 1

Field-Programmable Gate Array

An Altera Stratix IV GX FPGA



An example of a Xilinx Spartan 6 FPGA programming/evaluation board

A **Field-programmable Gate Array** (**FPGA**) is an integrated circuit designed to be configured by the customer or designer after manufacturing—hence "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC) (circuit diagrams were previously used to specify the configuration, as they were for ASICs, but this is increasingly rare). FPGAs can be used to implement any logical function that an ASIC could perform. The ability to update the functionality after shipping, partial re-configuration of the portion of the design and the low non-recurring engineering costs relative to an ASIC design (notwithstanding the generally higher unit cost), offer advantages for many applications.

FPGAs contain programmable logic components called "logic blocks", and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together"—somewhat like many (changeable) logic gates that can be inter-wired in (many) different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, the logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory.

In addition to digital functions, some FPGAs have analog features. The most common analog feature is programmable slew rate and drive strength on each output pin, allowing the engineer to set slow rates on lightly loaded pins that would otherwise ring unacceptably, and to set stronger, faster rates on heavily loaded pins on high-speed channels that would otherwise run too slow. Another relatively common analog feature is differential comparators on input pins designed to be connected to differential signaling channels. A few "mixed signal FPGAs" have integrated peripheral Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs) with analog signal conditioning blocks allowing them to operate as a system-on-a-chip. Such devices blur the line between an FPGA, which carries digital ones and zeros on its internal programmable interconnect fabric, and field-programmable analog array (FPAA), which carries analog values on its internal programmable interconnect fabric.

History

The FPGA industry sprouted from programmable read-only memory (PROM) and programmable logic devices (PLDs). PROMs and PLDs both had the option of being programmed in batches in a factory or in the field (field programmable), however programmable logic was hard-wired between logic gates.

In the late 1980s the Naval Surface Warfare Department funded an experiment proposed by Steve Casselman to develop a computer that would implement 600,000 reprogrammable gates. Casselman was successful and a patent related to the system was issued in 1992.

Some of the industry’s foundational concepts and technologies for programmable logic arrays, gates, and logic blocks are founded in patents awarded to David W. Page and LuVerne R. Peterson in 1985.

Xilinx Co-Founders, Ross Freeman and Bernard Vonderschmitt, invented the first commercially viable field programmable gate array in 1985 – the XC2064. The XC2064 had programmable gates and programmable interconnects between gates, the beginnings of a new technology and market. The XC2064 boasted a mere 64 configurable logic blocks (CLBs), with two 3-input lookup tables (LUTs). More than 20 years later, Freeman was entered into the National Inventors Hall of Fame for his invention.

Xilinx continued unchallenged and quickly growing from 1985 to the mid-1990s, when competitors sprouted up, eroding significant market-share. By 1993, Actel was serving about 18 percent of the market.

The 1990s were an explosive period of time for FPGAs, both in sophistication and the volume of production. In the early 1990s, FPGAs were primarily used in telecommunications and networking. By the end of the decade, FPGAs found their way into consumer, automotive, and industrial applications.

FPGAs got a glimpse of fame in 1997, when Adrian Thompson, a researcher working at the University of Sussex, merged genetic algorithm technology and FPGAs to create a sound recognition device. Thomson’s algorithm configured an array of 10 x 10 cells in a Xilinx FPGA chip to discriminate between two tones, utilising analogue features of the digital chip. The application of genetic algorithms to the configuration of devices like FPGA's is now referred to as Evolvable hardware

Modern developments

A recent trend has been to take the coarse-grained architectural approach a step further by combining the logic blocks and interconnects of traditional FPGAs with embedded microprocessors and related peripherals to form a complete "system on a programmable chip". This work mirrors the architecture by Ron Perlof and Hana Potash of Burroughs Advanced Systems Group which combined a reconfigurable CPU architecture on a single chip called the SB24. That work was done in 1982. Examples of such hybrid technologies can be found in the Xilinx Virtex-II PRO and Virtex-4 devices, which include one or more PowerPC processors embedded within the FPGA's logic fabric. The Atmel FPSLIC is another such device, which uses an AVR processor in combination with Atmel's programmable logic architecture. The Actel SmartFusion devices incorporate an ARM architecture Cortex-M3 hard processor core (with up to 512kB of flash and 64kB of RAM) and analog peripherals such as a multi-channel ADC and DACs to their flash-based FPGA fabric.

An alternate approach to using hard-macro processors is to make use of soft processor cores that are implemented within the FPGA logic.

As previously mentioned, many modern FPGAs have the ability to be reprogrammed at "run time," and this is leading to the idea of reconfigurable computing or reconfigurable systems — CPUs that reconfigure themselves to suit the task at hand. The Mitrion Virtual Processor from Mitrionics is an example of a reconfigurable soft processor, implemented on FPGAs. However, it does not support dynamic reconfiguration at runtime, but instead adapts itself to a specific program.

Additionally, new, non-FPGA architectures are beginning to emerge. Software-configurable microprocessors such as the Stretch S5000 adopt a hybrid approach by providing an array of processor cores and FPGA-like programmable cores on the same chip.

Gates

- 1987: 9,000 gates, Xilinx
- 1992: 600,000, Naval Surface Warfare Department
- Early 2000s: Millions

Market size

- 1985: First commercial FPGA technology invented by Xilinx
- 1987: $14 million
- ~1993: >$385 million
- 2005: $1.9 billion

- 2010 estimates: $2.75 billion

FPGA design starts

- 10,000
- 2005: 80,000
- 2008: 90,000

FPGA comparisons

Historically, FPGAs have been slower, less energy efficient and generally achieved less functionality than their fixed ASIC counterparts. A study has shown that designs implemented on FPGAs need on average 18 times as much area, draw 7 times as much dynamic power, and are 3 times slower than the corresponding ASIC implementations.

An Altera Cyclone II FPGA, on an Altera teraSIC DE1 Prototyping board.

Advantages include the ability to re-program in the field to fix bugs, and may include a shorter time to market and lower non-recurring engineering costs. Vendors can also take a middle road by developing their hardware on ordinary FPGAs, but manufacture their final version so it can no longer be modified after the design has been committed.

Xilinx claims that several market and technology dynamics are changing the ASIC/FPGA paradigm:

- Integrated circuit costs are rising aggressively
- ASIC complexity has lengthened development time
- R&D resources and headcount are decreasing

- Revenue losses for slow time-to-market are increasing
- Financial constraints in a poor economy are driving low-cost technologies

These trends make FPGAs a better alternative than ASICs for a larger number of higher-volume applications than they have been historically used for, to which the company attributes the growing number of FPGA design starts.

Some FPGAs have the capability of partial re-configuration that lets one portion of the device be re-programmed while other portions continue running.

Versus complex programmable logic devices

The primary differences between CPLDs (Complex Programmable Logic Devices) and FPGAs are architectural. A CPLD has a somewhat restrictive structure consisting of one or more programmable sum-of-products logic arrays feeding a relatively small number of clocked registers. The result of this is less flexibility, with the advantage of more predictable timing delays and a higher logic-to-interconnect ratio. The FPGA architectures, on the other hand, are dominated by interconnect. This makes them far more flexible (in terms of the range of designs that are practical for implementation within them) but also far more complex to design for.

Another notable difference between CPLDs and FPGAs is the presence in most FPGAs of higher-level embedded functions (such as adders and multipliers) and embedded memories, as well as to have logic blocks implement decoders or mathematical functions.

Security considerations

With respect to security, FPGAs have both advantages and disadvantages as compared to ASICs or secure microprocessors. FPGAs' flexibility makes malicious modifications during fabrication a lower risk. For many FPGAs, the loaded design is exposed while it is loaded (typically on every power-on). To address this issue, some FPGAs support bitstream encryption.

Applications

Applications of FPGAs include digital signal processing, software-defined radio, aerospace and defense systems, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation, radio astronomy, metal detection and a growing range of other areas.

FPGAs originally began as competitors to CPLDs and competed in a similar space, that of glue logic for PCBs. As their size, capabilities, and speed increased, they began to take over larger and larger functions to the state where some are now marketed as full systems on chips (SoC). Particularly with the introduction of dedicated multipliers into FPGA architectures in the late 1990s, applications which had traditionally been the sole reserve of DSPs began to incorporate FPGAs instead.

FPGAs especially find applications in any area or algorithm that can make use of the massive parallelism offered by their architecture. One such area is code breaking, in particular brute-force attack, of cryptographic algorithms.

FPGAs are increasingly used in conventional high performance computing applications where computational kernels such as FFT or Convolution are performed on the FPGA instead of a microprocessor.

The inherent parallelism of the logic resources on an FPGA allows for considerable computational throughput even at a low MHz clock rates. The flexibility of the FPGA allows for even higher performance by trading off precision and range in the number format for an increased number of parallel arithmetic units. This has driven a new type of processing called reconfigurable computing, where time intensive tasks are offloaded from software to FPGAs.

The adoption of FPGAs in high performance computing is currently limited by the complexity of FPGA design compared to conventional software and the turn-around times of current design tools.

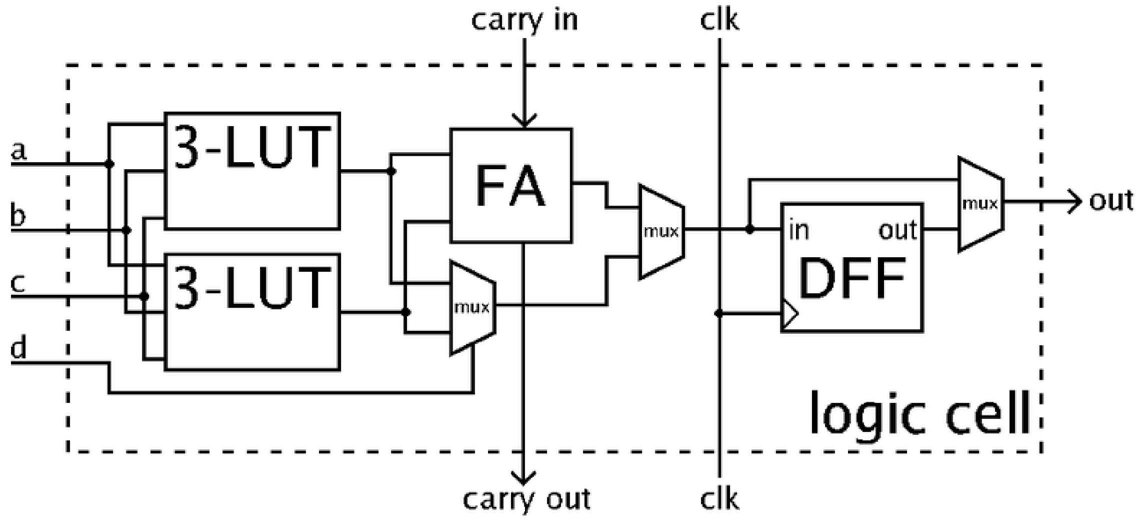
Traditionally, FPGAs have been reserved for specific vertical applications where the volume of production is small. For these low-volume applications, the premium that companies pay in hardware costs per unit for a programmable chip is more affordable than the development resources spent on creating an ASIC for a low-volume application. Today, new cost and performance dynamics have broadened the range of viable applications.

Architecture

The most common FPGA architecture consists of an array of logic blocks (called Configurable Logic Block, CLB, or Logic Array Block, LAB, depending on vendor), I/O pads, and routing channels. Generally, all the routing channels have the same width (number of wires). Multiple I/O pads may fit into the height of one row or the width of one column in the array.

An application circuit must be mapped into an FPGA with adequate resources. While the number of CLBs/LABs and I/Os required is easily determined from the design, the number of routing tracks needed may vary considerably even among designs with the same amount of logic. For example, a crossbar switch requires much more routing than a systolic array with the same gate count. Since unused routing tracks increase the cost (and decrease the performance) of the part without providing any benefit, FPGA manufacturers try to provide just enough tracks so that most designs that will fit in terms of LUTs and IOs can be routed. This is determined by estimates such as those derived from Rent's rule or by experiments with existing designs.

In general, a logic block (CLB or LAB) consists of a few logical cells (called ALM, LE, Slice etc). A typical cell consists of a 4-input Lookup table (LUT), a Full adder (FA) and a D-type flip-flop, as shown below. The LUTs are in this figure split into two 3-input LUTs. In *normal mode* those are combined into a 4-input LUT through the left mux. In *arithmetic* mode, their outputs are fed to the FA. The selection of mode is programmed into the middle mux. The output can be either synchronous or asynchronous, depending on the programming of the mux to the right, in the figure example. In practice, entire or parts of the FA are put as functions into the LUTs in order to save space.



Simplified example illustration of a logic cell

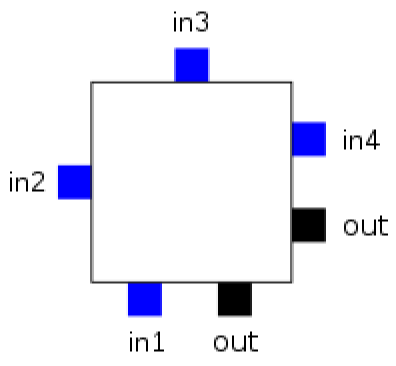
ALMs and Slices usually contains 2 or 4 structures similar to the example figure, with some shared signals.

CLBs/LABs typically contains a few ALMs/LEs/Slices.

In recent years, manufacturers have started moving to 6-input LUTs in their high performance parts, claiming increased performance.

Since clock signals (and often other high-fanout signals) are normally routed via special-purpose dedicated routing networks in commercial FPGAs, they and other signals are separately managed.

For this example architecture, the locations of the FPGA logic block pins are shown below.



Logic Block Pin Locations

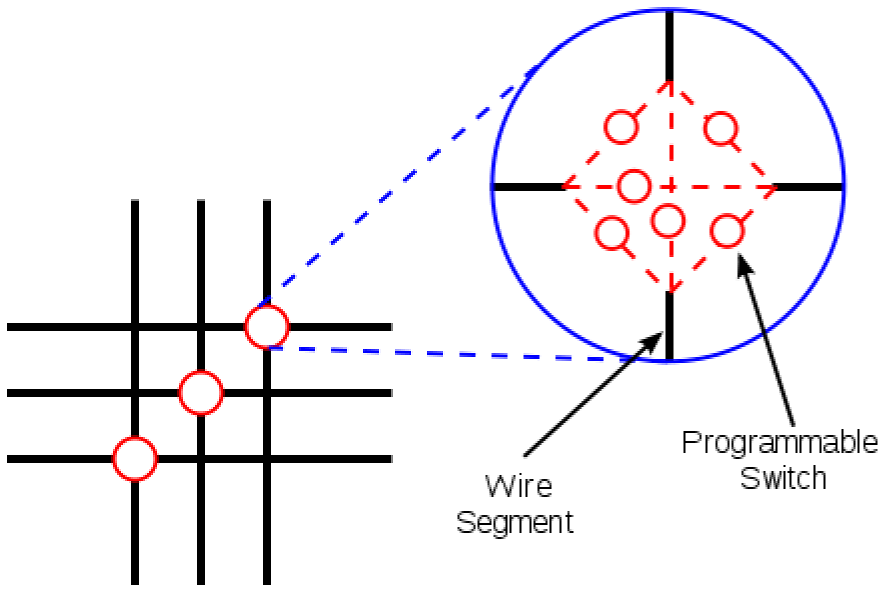
Each input is accessible from one side of the logic block, while the output pin can connect to routing wires in both the channel to the right and the channel below the logic block.

Each logic block output pin can connect to any of the wiring segments in the channels adjacent to it.

Similarly, an I/O pad can connect to any one of the wiring segments in the channel adjacent to it. For example, an I/O pad at the top of the chip can connect to any of the W wires (where W is the channel width) in the horizontal channel immediately below it.

Generally, the FPGA routing is unsegmented. That is, each wiring segment spans only one logic block before it terminates in a switch box. By turning on some of the programmable switches within a switch box, longer paths can be constructed. For higher speed interconnect, some FPGA architectures use longer routing lines that span multiple logic blocks.

Whenever a vertical and a horizontal channel intersect, there is a switch box. In this architecture, when a wire enters a switch box, there are three programmable switches that allow it to connect to three other wires in adjacent channel segments. The pattern, or topology, of switches used in this architecture is the planar or domain-based switch box topology. In this switch box topology, a wire in track number one connects only to wires in track number one in adjacent channel segments, wires in track number 2 connect only to other wires in track number 2 and so on. The figure below illustrates the connections in a switch box.



Switch box topology

Modern FPGA families expand upon the above capabilities to include higher level functionality fixed into the silicon. Having these common functions embedded into the silicon reduces the area required and gives those functions increased speed compared to building them from primitives. Examples of these include multipliers, generic DSP blocks, embedded processors, high speed IO logic and embedded memories.

FPGAs are also widely used for systems validation including pre-silicon validation, post-silicon validation, and firmware development. This allows chip companies to validate their design before the chip is produced in the factory, reducing the time-to-market.

FPGA design and programming

To define the behavior of the FPGA, the user provides a hardware description language (HDL) or a schematic design. The HDL form is more suited to work with large structures because it's possible to just specify them numerically rather than having to draw every piece by hand. However, schematic entry can allow for easier visualisation of a design.

Then, using an electronic design automation tool, a technology-mapped netlist is generated. The netlist can then be fitted to the actual FPGA architecture using a process called place-and-route, usually performed by the FPGA company's proprietary place-and-route software. The user will validate the map, place and route results via timing analysis, simulation, and other verification methodologies. Once the design and validation process is complete, the binary file generated (also using the FPGA company's proprietary software) is used to (re)configure the FPGA.

Going from schematic/HDL source files to actual configuration: The source files are fed to a software suite from the FPGA/CPLD vendor that through different steps will produce a file. This file is then transferred to the FPGA/CPLD via a serial interface (JTAG) or to an external memory device like an EEPROM.

The most common HDLs are VHDL and Verilog, although in an attempt to reduce the complexity of designing in HDLs, which have been compared to the equivalent of assembly languages, there are moves to raise the abstraction level through the introduction of alternative languages. National Instrument's LabVIEW graphical programming language (sometimes referred to as "G") has an FPGA add-in module available to target and program FPGA hardware.

To simplify the design of complex systems in FPGAs, there exist libraries of predefined complex functions and circuits that have been tested and optimized to speed up the design process. These predefined circuits are commonly called *IP cores*, and are available from FPGA vendors and third-party IP suppliers (rarely free, and typically released under proprietary licenses). Other predefined circuits are available from developer communities such as OpenCores (typically released under free and open source licenses such as the GPL, BSD or similar license), and other sources.

In a typical design flow, an FPGA application developer will simulate the design at multiple stages throughout the design process. Initially the RTL description in VHDL or Verilog is simulated by creating test benches to simulate the system and observe results. Then, after the synthesis engine has mapped the design to a netlist, the netlist is translated to a gate level description where simulation is repeated to confirm the synthesis proceeded without errors. Finally the design is laid out in the FPGA at which point propagation delays can be added and the simulation run again with these values back-annotated onto the netlist.

Basic process technology types

- SRAM - based on static memory technology. In-system programmable and re-programmable. Requires external boot devices. CMOS.
- Antifuse - One-time programmable. CMOS.
- PROM - Programmable Read-Only Memory technology. One-time programmable because of plastic packaging.
- EPROM - Erasable Programmable Read-Only Memory technology. One-time programmable but with window, can be erased with ultraviolet (UV) light. CMOS.
- EEPROM - Electrically Erasable Programmable Read-Only Memory technology. Can be erased, even in plastic packages. Some but not all EEPROM devices can be in-system programmed. CMOS.
- Flash - Flash-erase EPROM technology. Can be erased, even in plastic packages. Some but not all flash devices can be in-system programmed. Usually, a flash cell is smaller than an equivalent EEPROM cell and is therefore less expensive to manufacture. CMOS.
- Fuse - One-time programmable. Bipolar.

Major manufacturers

Xilinx and Altera are the current FPGA market leaders and long-time industry rivals. Together, they control over 80 percent of the market, with Xilinx alone representing over 50 percent.

Both Xilinx and Altera provide free Windows and Linux design software.

Other competitors include Lattice Semiconductor (SRAM based with integrated configuration Flash, instant-on, low power, live reconfiguration), Actel (antifuse, flash-based, mixed-signal), SiliconBlue Technologies (extremely low power SRAM-based FPGAs with option integrated nonvolatile configuration memory), Achronix (RAM based, 1.5 GHz fabric speed) who will be building their chips on Intels' state-of-the art 22 nm process, and QuickLogic (handheld focused CSSP, no general purpose FPGAs).

In March 2010, Tabula announced their new FPGA technology that uses time-multiplexed logic and interconnect for greater potential cost savings for high-density applications.

Chapter 2

Application-Specific Integrated Circuit

An **application-specific integrated circuit** (ASIC) is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. For example, a chip designed solely to run a cell phone is an ASIC. Application-specific standard products (ASSPs) are intermediate between ASICs and industry standard integrated circuits like the 7400 or the 4000 series.

As feature sizes have shrunk and design tools improved over the years, the maximum complexity (and hence functionality) possible in an ASIC has grown from 5,000 gates to over 100 million. Modern ASICs often include entire 32-bit processors, memory blocks including ROM, RAM, EEPROM, Flash and other large building blocks. Such an ASIC is often termed a SoC (system-on-a-chip). Designers of digital ASICs use a hardware description language (HDL), such as Verilog or VHDL, to describe the functionality of ASICs.

Field-programmable gate arrays (FPGA) are the modern-day technology for building a breadboard or prototype from standard parts; programmable logic blocks and programmable interconnects allow the same FPGA to be used in many different applications. For smaller designs and/or lower production volumes, FPGAs may be more cost effective than an ASIC design even in production. The non-recurring engineering cost of an ASIC can run into the millions of dollars.

History

The initial ASICs used gate array technology. Ferranti produced perhaps the first gate-array, the ULA (Uncommitted Logic Array), around 1980. An early successful commercial application was the ULA circuitry found in the 8-bit ZX81 and ZX Spectrum low-end personal computers, introduced in 1981 and 1982. These were used by Sinclair

Research (UK) essentially as a low-cost I/O solution aimed at handling the computer's graphics. Some versions of ZX81/Timex Sinclair 1000 used just four chips (ULA, 2Kx8 RAM, 8Kx8 ROM, Z80A CPU) to implement an entire mass-market personal computer with built-in BASIC interpreter.

Customization occurred by varying the metal interconnect mask. ULAs had complexities of up to a few thousand gates. Later versions became more generalized, with different base dies customised by both metal and polysilicon layers. Some base dies include RAM elements.

Standard cell design

In the mid 1980s, a designer would choose an ASIC manufacturer and implement their design using the design tools available from the manufacturer. While third-party design tools were available, there was not an effective link from the third-party design tools to the layout and actual semiconductor process performance characteristics of the various ASIC manufacturers. Most designers ended up using factory-specific tools to complete the implementation of their designs. A solution to this problem, which also yielded a much higher density device, was the implementation of Standard Cells. Every ASIC manufacturer could create functional blocks with known electrical characteristics, such as propagation delay, capacitance and inductance, that could also be represented in third-party tools. Standard Cell design is the utilization of these functional blocks to achieve very high gate density and good electrical performance. Standard cell design fits between Gate Array and Full Custom design in terms of both its NRE (Non-Recurring Engineering) and recurring component cost.

By the late 1990s, logic synthesis tools became available. Such tools could compile HDL descriptions into a gate-level netlist. Standard-cell Integrated Circuits (ICs) are designed in the following conceptual stages, although these stages overlap significantly in practice.

1. A team of design engineers starts with a non-formal understanding of the required functions for a new ASIC, usually derived from Requirements analysis.
2. The design team constructs a description of an ASIC to achieve these goals using an HDL. This process is analogous to writing a computer program in a high-level language. This is usually called the RTL (Register transfer level) design.
3. Suitability for purpose is verified by functional verification. This may include such techniques as logic simulation, formal verification, emulation, or creating an equivalent pure software model. Each technique has advantages and disadvantages, and often several methods are used.
4. Logic synthesis transforms the RTL design into a large collection of lower-level constructs called standard cells. These constructs are taken from a standard-cell library consisting of pre-characterized collections of gates (such as 2 input nor, 2 input nand, inverters, etc.). The standard cells are typically specific to the planned manufacturer of the ASIC. The resulting collection of standard cells, plus the needed electrical connections between them, is called a gate-level netlist.

5. The gate-level netlist is next processed by a placement tool which places the standard cells onto a region representing the final ASIC. It attempts to find a placement of the standard cells, subject to a variety of specified constraints.
6. The routing tool takes the physical placement of the standard cells and uses the netlist to create the electrical connections between them. Since the search space is large, this process will produce a “sufficient” rather than “globally-optimal” solution. The output is a file which can be used to create a set of photomasks enabling a semiconductor fabrication facility (commonly called a 'fab') to produce physical ICs.
7. Given the final layout, circuit extraction computes the parasitic resistances and capacitances. In the case of a digital circuit, this will then be further mapped into delay information, from which the circuit performance can be estimated, usually by static timing analysis. This, and other final tests such as design rule checking and power analysis (collectively called signoff) are intended to ensure that the device will function correctly over all extremes of the process, voltage and temperature. When this testing is complete the photomask information is released for chip fabrication.

These steps, implemented with a level of skill common in the industry, almost always produce a final device that correctly implements the original design, unless flaws are later introduced by the physical fabrication process.

The design steps (or flow) are also common to standard product design. The significant difference is that Standard Cell design uses the manufacturer's cell libraries that have been used in potentially hundreds of other design implementations and therefore are of much lower risk than full custom design. Standard Cells produce a design density that is cost effective, and they can also integrate IP cores and SRAM (Static Random Access Memory) effectively, unlike Gate Arrays.

Gate array design

Gate array design is a manufacturing method in which the diffused layers, i.e. transistors and other active devices, are predefined and wafers containing such devices are held in stock prior to metallization — in other words, unconnected. The physical design process then defines the interconnections of the final device. For most ASIC manufacturers, this consists of from two to as many as nine metal layers, each metal layer running perpendicular to the one below it. Non-recurring engineering costs are much lower, as photolithographic masks are required only for the metal layers, and production cycles are much shorter, as metallization is a comparatively quick process.

Gate array ASICs are always a compromise as mapping a given design onto what a manufacturer held as a stock wafer never gives 100% utilization. Often difficulties in routing the interconnect require migration onto a larger array device with consequent increase in the piece part price. These difficulties are often a result of the layout software used to develop the interconnect.

Pure, logic-only gate array design is rarely implemented by circuit designers today, having been replaced almost entirely by field-programmable devices, such as field-programmable gate arrays (FPGAs), which can be programmed by the user and thus offer minimal tooling charges (non-recurring engineering (NRE)), only marginally increased piece part cost, and comparable performance. Today, gate arrays are evolving into structured ASICs that consist of a large IP core like a CPU, DSP unit, peripherals, standard interfaces, integrated memories SRAM, and a block of reconfigurable, uncommited logic. This shift is largely because ASIC devices are capable of integrating such large blocks of system functionality and "system-on-a-chip" requires far more than just logic blocks.

In their frequent usages in the field, the terms "gate array" and "semi-custom" are synonymous. Process engineers more commonly use the term "semi-custom", while "gate-array" is more commonly used by logic (or gate-level) designers.

Full-custom design

By contrast, full-custom ASIC design defines all the photolithographic layers of the device. Full-custom design is used for both ASIC design and for standard product design.

The benefits of full-custom design usually include reduced area (and therefore recurring component cost), performance improvements, and also the ability to integrate analog components and other pre-designed — and thus fully verified — components, such as microprocessor cores that form a system-on-chip.

The disadvantages of full-custom design can include increased manufacturing and design time, increased non-recurring engineering costs, more complexity in the computer-aided design (CAD) system, and a much higher skill requirement on the part of the design team.

For digital-only designs, however, "standard-cell" cell libraries, together with modern CAD systems, can offer considerable performance/cost benefits with low risk. Automated layout tools are quick and easy to use and also offer the possibility to "hand-tweak" or manually optimize any performance-limiting aspect of the design.

Structured design

Structured ASIC design (also referred to as "platform ASIC design"), is a relatively new term in the industry, resulting in some variation in its definition. However, the basic premise of a structured ASIC is that both manufacturing cycle time and design cycle time are reduced compared to cell-based ASIC, by virtue of there being pre-defined metal layers (thus reducing manufacturing time) and pre-characterization of what is on the silicon (thus reducing design cycle time). One definition states that

> In a "structured ASIC" design, the logic mask-layers of a device are predefined by the ASIC vendor (or in some cases by a third party). Design differentiation and

> customization is achieved by creating custom metal layers that create custom connections between predefined lower-layer logic elements. "Structured ASIC" technology is seen as bridging the gap between field-programmable gate arrays and "standard-cell" ASIC designs. Because only a small number of chip layers must be custom-produced, "structured ASIC" designs have much smaller non-recurring expenditures (NRE) than "standard-cell" or "full-custom" chips, which require that a full mask set be produced for every design.

This is effectively the same definition as a gate array. What makes a structured ASIC different is that in a gate array, the predefined metal layers serve to make manufacturing turnaround faster. In a structured ASIC, the use of predefined metallization is primarily to reduce cost of the mask sets as well as making the design cycle time significantly shorter. For example, in a cell-based or gate-array design the user must often design power, clock, and test structures themselves; these are predefined in most structured ASICs and therefore can save time and expense for the designer compared to gate-array. Likewise, the design tools used for structured ASIC can be substantially lower cost and easier (faster) to use than cell-based tools, because they do not have to perform all the functions that cell-based tools do. In some cases, the structured ASIC vendor requires that customized tools for their device (e.g., custom physical synthesis) be used, also allowing for the design to be brought into manufacturing more quickly.

One other important aspect about structured ASIC is that it allows intellectual property (IP) that is common to certain applications or industry segments to be "built in", rather than "designed in". By building the IP directly into the architecture the designer can again save both time and money compared to designing IP into a cell-based ASIC.

Cell libraries, IP-based design, hard and soft macros

Cell libraries of logical primitives are usually provided by the device manufacturer as part of the service. Although they will incur no additional cost, their release will be covered by the terms of a non-disclosure agreement (NDA) and they will be regarded as intellectual property by the manufacturer. Usually their physical design will be pre-defined so they could be termed "hard macros".

What most engineers understand as "intellectual property" are IP cores, designs purchased from a third-party as sub-components of a larger ASIC. They may be provided as an HDL description (often termed a "soft macro"), or as a fully routed design that could be printed directly onto an ASIC's mask (often termed a hard macro). Many organizations now sell such pre-designed cores — CPUs, Ethernet, USB or telephone interfaces — and larger organizations may have an entire department or division to produce cores for the rest of the organization. Indeed, the wide range of functions now available is a significant factor in the phenomenal increase in electronics in the late 1990s and early 2000s; as a core takes a lot of time and investment to create, its re-use and further development cuts product cycle times dramatically and creates better products. Additionally, organizations such as OpenCores are collecting free IP cores paralleling the open source movement in software.

Soft macros are often process-independent, i.e., they can be fabricated on a wide range of manufacturing processes and different manufacturers. Hard macros are process-limited and usually further design effort must be invested to migrate (port) to a different process or manufacturer.

Multi-project wafers

Some manufacturers offer Multi-Project Wafers (MPW) as a method of obtaining low cost prototypes. Often called shuttles, these MPW, containing several designs, run at regular, scheduled intervals on a "cut and go" basis, usually with very little liability on the part of the manufacturer. The contract involves the assembly and packaging of a handful of devices. The service usually involves the supply of a physical design data base i.e. masking information or Pattern Generation (PG) tape. The manufacturer is often referred to as a "silicon foundry" due to the low involvement it has in the process. .

ASIC suppliers

There are two different types of ASIC suppliers, IDM and fabless. An IDM supplier's ASIC product is based in large part on proprietary technology such as design tools, IP, packaging, and usually although not necessarily the process technology. Fabless ASIC suppliers rely almost exclusively on outside suppliers for their technology. The classification can be confusing since several IDM's are also fabless semiconductor companies.

IDM ASIC suppliers

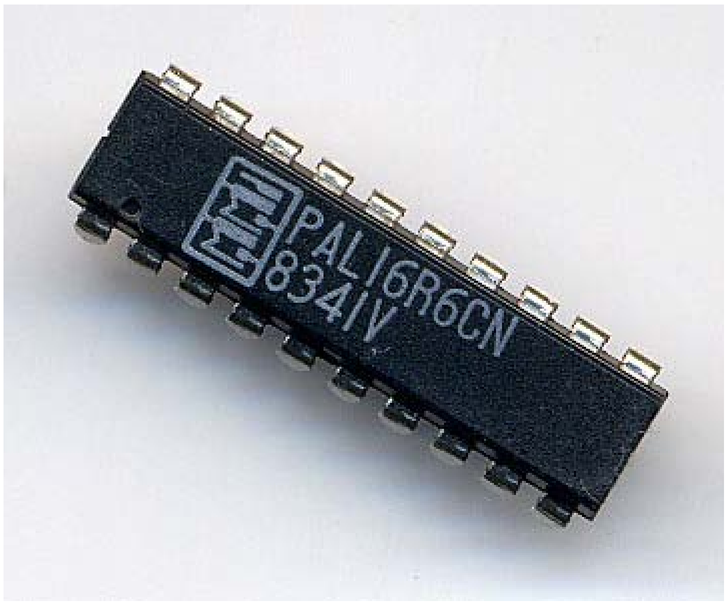
- Avago Technologies
- Elmos Semiconductor
- Cavium Networks
- Fujitsu
- Freescale
- HITACHI
- IBM
- Infineon Technologies
- LSI Corporation
- Marvell Semiconductor
- NEC
- NXP Semiconductors
- ON Semiconductor
- Renesas
- Samsung
- STMicroelectronics
- Texas Instruments
- Toshiba
- TSMC

Fabless ASIC suppliers

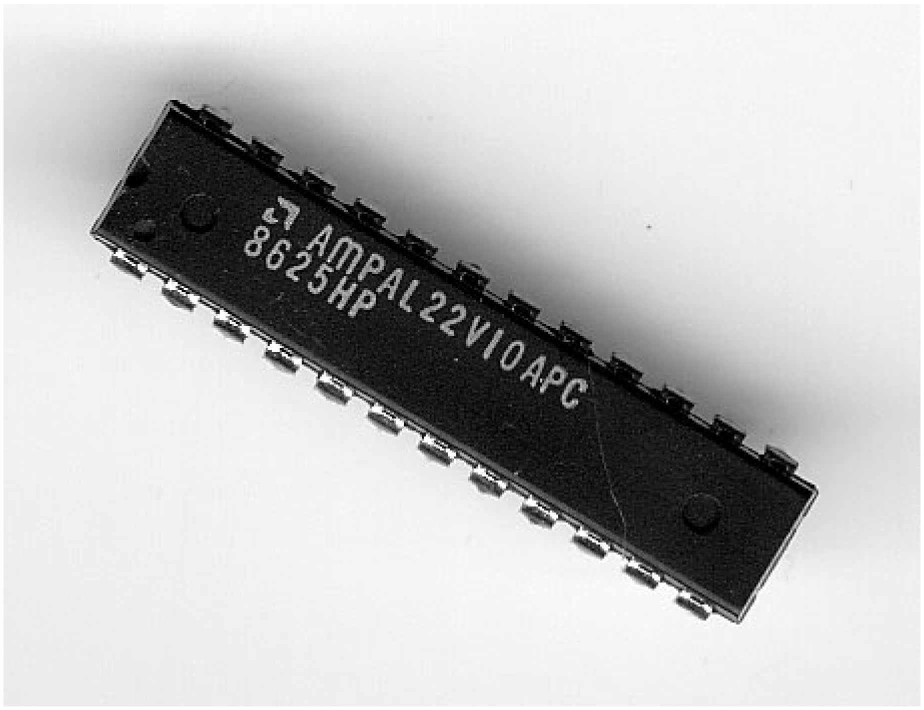
- Alchip
- Aeroflex Colorado Springs
- Brite Semiconductor
- Custom Silicon Solutions
- eASIC
- eSilicon
- Faraday Technology
- Hong Kong Science and Technology Parks Corporation
- JVD Inc.
- Marvell Semiconductor
- MOSIS
- Nvidia
- Open-Silicon
- PMC Sierra
- Qualcomm
- Socle
- System to ASIC

Chapter 3

Programmable Array Logic



MMI PAL 16R6 in 20-pin DIP



AMD 22V10 in 24-pin DIP

The term **Programmable Array Logic** (PAL) is used to describe a family of programmable logic device semiconductors used to implement logic functions in digital circuits introduced by Monolithic Memories, Inc. (MMI) in March 1978. MMI obtained a registered trademark on the term PAL for use in "Programmable Semiconductor Logic Circuits". The trademark is currently held by Lattice Semiconductor.

PAL devices consisted of a small PROM (programmable read-only memory) core and additional output logic used to implement particular desired logic functions with few components.

Using specialized machines, PAL devices were "field-programmable". Each PAL device was "one-time programmable" (OTP), meaning that it could not be updated and reused after its initial programming. (MMI also offered a similar family called HAL, or "hard array logic", which were like PAL devices except that they were mask-programmed at the factory.)

Early history

Before PALs were introduced, designers of digital logic circuits would use small-scale integration (SSI) components, such as those in the 7400 series TTL (transistor-transistor logic) family; the 7400 family included a variety of logic building blocks, such as gates (NOT, NAND, NOR, AND, OR), multiplexers (MUXes) and demultiplexers (DEMUXes), flip flops (D-type, JK, etc.) and others. One PAL device would typically replace dozens of such "discrete" logic packages, so the SSI business went into decline as the PAL business took off. PALs were used advantageously in many products, such as minicomputers, as documented in Tracy Kidder's best-selling book "The Soul of a New Machine."

PALs were not the first commercial programmable logic devices; Signetics had been selling its field programmable logic array (FPLA) since 1975. These devices were completely unfamiliar to most circuit designers and were perceived to be too difficult to use. The FPLA had a relatively slow maximum operating speed (due to having both programmable-AND and programmable-OR arrays), was expensive, and had a poor reputation for testability. Another factor limiting the acceptance of the FPLA was the large package, a 600-mil (0.6", or 15.24 mm) wide 28-pin dual in-line package (DIP).

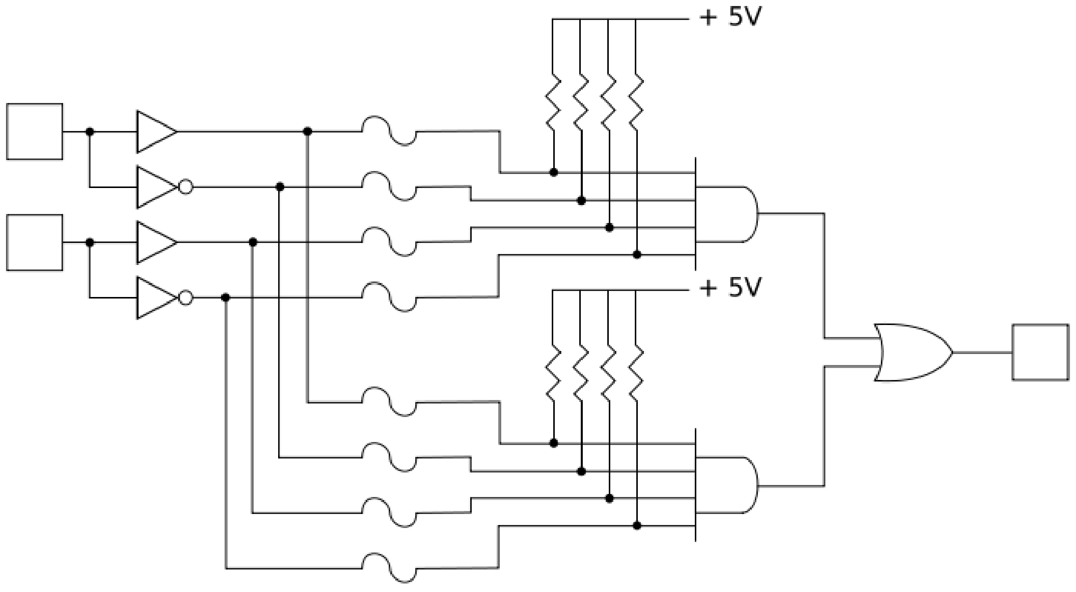
The project to create the PAL device was managed by John Birkner and the actual PAL circuit was designed by H. T. Chua. In a previous job, Birkner had developed a 16-bit processor using 80 standard logic devices. His experience with standard logic led him to believe that user programmable devices would be more attractive to users if the devices were designed to replace standard logic. This meant that the package sizes had to be more typical of the existing devices, and the speeds had to be improved. The PAL met these requirements and was a huge success and were "second sourced" by National Semiconductor, Texas Instruments, and Advanced Micro Devices.

Process technologies

Early PALs were 20-pin DIP components fabricated in silicon using bipolar transistor technology with one-time programmable (OTP) titanium-tungsten programming fuses. Later devices were manufactured by Lattice Semiconductor and Advanced Micro Devices using CMOS technology.

The original 20 and 24-pin PALs were described by MMI as medium-scale integration (MSI) devices.

PAL architecture



Simplified programmable logic device

The programmable elements (shown as a fuse) connect both the true and complemented inputs to the AND gates. These AND gates, also known as *product terms*, are ORed together to form a *sum-of-products* logic array.

The PAL architecture consists of two main components: a logic plane and output logic macrocells.

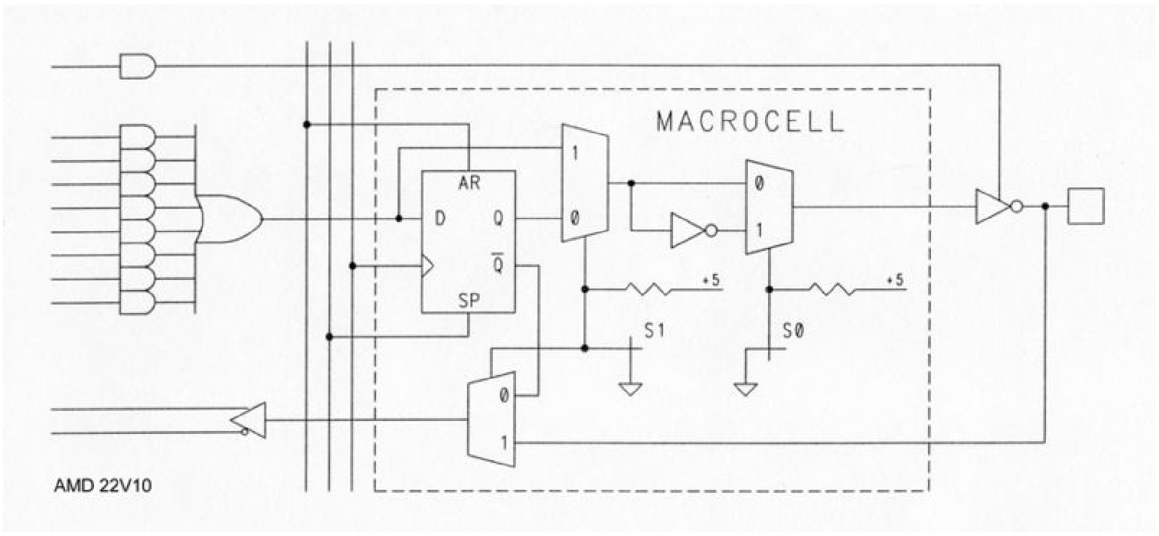
Programmable logic plane

The programmable logic plane is a programmable read-only memory (PROM) array that allows the signals present on the devices pins (or the logical complements of those signals) to be routed to an output logic macrocell.

PAL devices have arrays of transistor cells arranged in a "fixed-OR, programmable-AND" plane used to implement "sum-of-products" binary logic equations for each of the outputs in terms of the inputs and either synchronous or asynchronous feedback from the outputs.

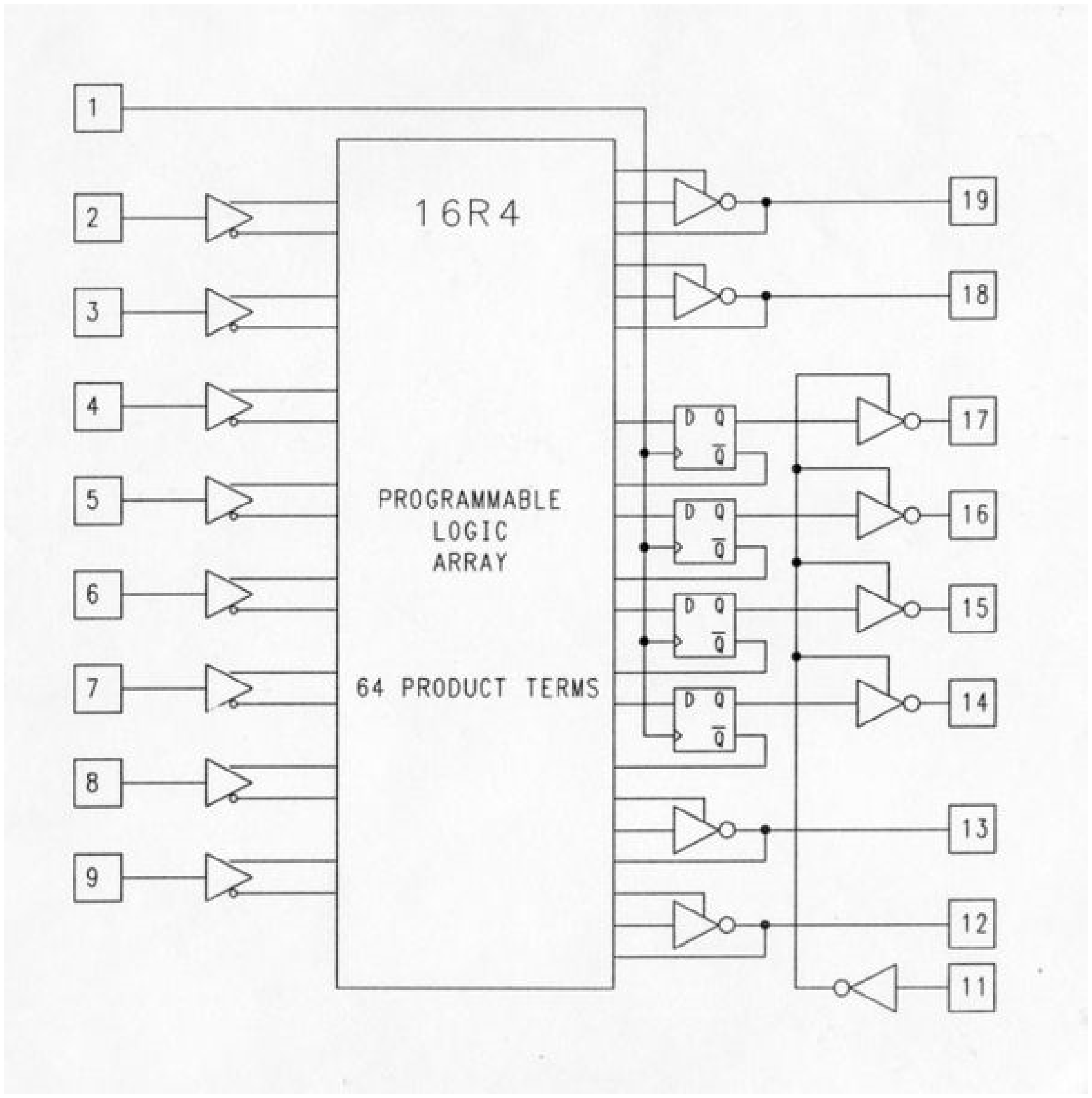
Output logic

The early 20-pin PALs had 10 inputs and 8 outputs. The outputs were active low and could be registered or combinational. Members of the PAL family were available with various output structures called "output logic macrocells" or OLMCs. Prior to the introduction of the "V" (for "variable") series, the types of OLMCs available in each PAL were fixed at the time of manufacture. (The PAL16L8 had 8 combinational outputs and the PAL16R8 had 8 registered outputs. The PAL16R6 had 6 registered and 2 combinational while the PAL16R4 had 4 of each.) Each output could have up to 8 product terms (effectively AND gates), however the combinational outputs used one of the terms to control a bidirectional output buffer. There were other combinations that had fewer outputs with more product term per output and were available with active high outputs. The 16X8 family or registered devices had an XOR gate before the register. There were also similar 24-pin versions of these PALs.

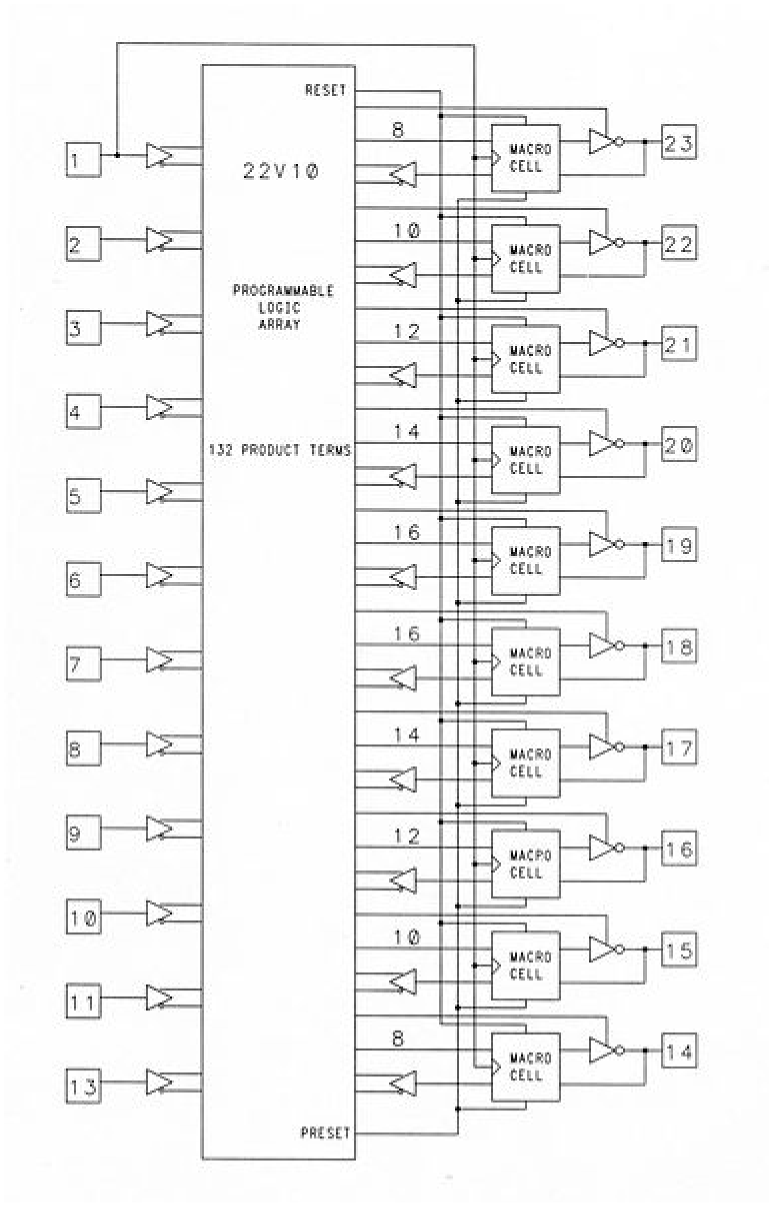


AMD 22V10 Output Macrocell

This fixed output structure often frustrated designers attempting to optimize the utility of PAL devices because output structures of different types were often required by their applications. (For example, one could not get 5 registered outputs with 3 active high combinational outputs.) So, in June 1983 AMD introduced the 22V10, a 24 pin device with 10 output logic macrocells. Each macrocell could be configured by the user to be combinational or registered, active high or active low. The number of product term allocated to an output varied from 8 to 16. This one device could replace all of the 24 pin fixed function PAL devices. Members of the PAL "V" ("variable") series included the PAL16V8, PAL20V8 and PAL22V10.



PAL 16R4 Block Diagram

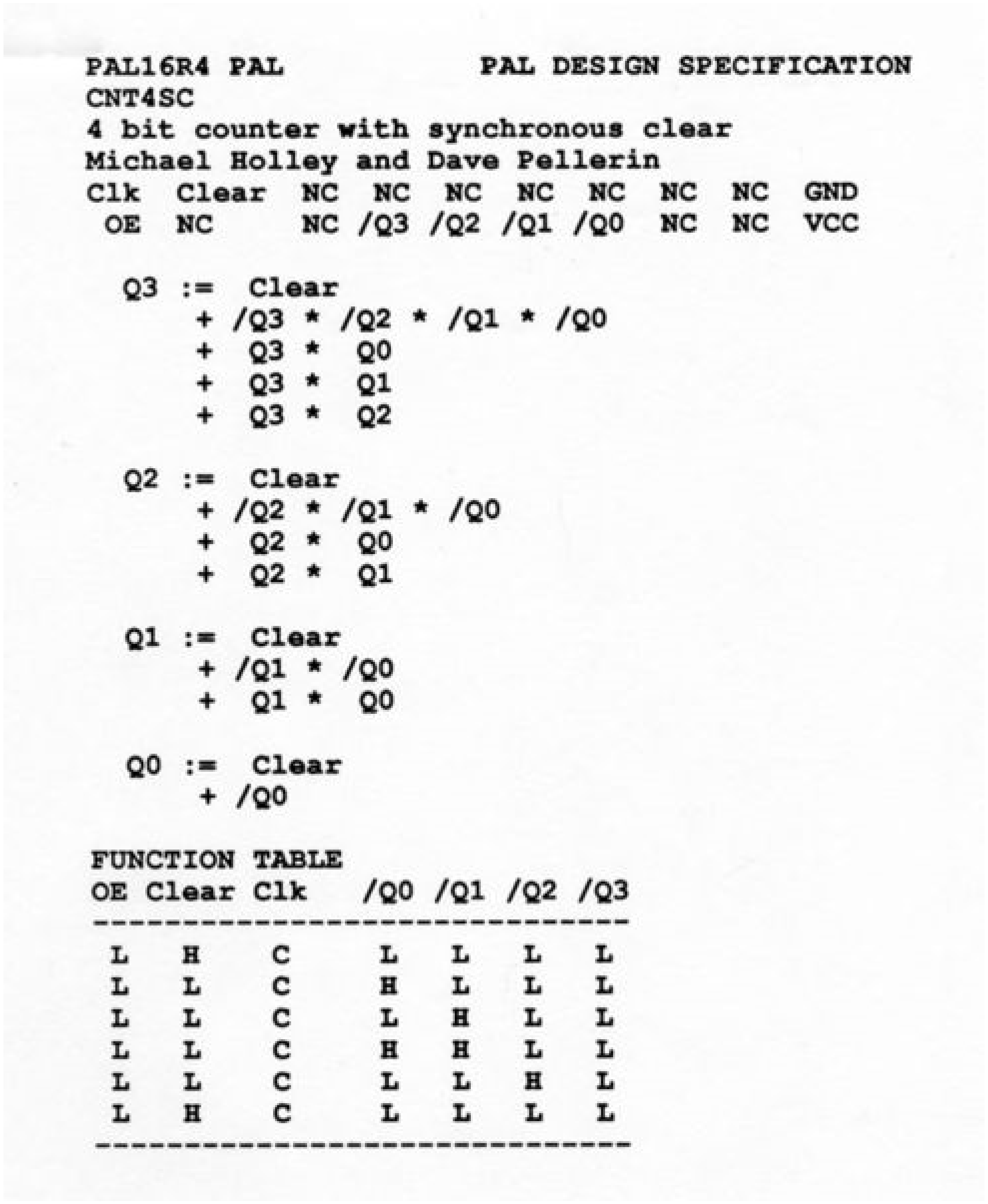


AMD 22V10 Block Diagram

Programming PALs

PALs were programmed electrically using binary patterns (as JEDEC ASCII/hexadecimal files) and a special electronic programming system available from either the manufacturer or a third-party, such as DATA/IO. In addition to single-unit device programmers, device feeders and gang programmers were often used when more than just a few PALs needed to be programmed. (For large volumes, electrical programming costs could be eliminated by having the manufacturer fabricate a custom metal mask used to program the customers' patterns at the time of manufacture; MMI used the term "hard array logic" (HAL) to refer to devices programmed in this way.)

Programming languages



```
PAL16R4 PAL          PAL DESIGN SPECIFICATION
CNT4SC
4 bit counter with synchronous clear
Michael Holley and Dave Pellerin
Clk  Clear  NC  NC  NC  NC  NC  NC  NC  GND
 OE  NC     NC /Q3 /Q2 /Q1 /Q0  NC  NC  VCC

  Q3 :=  Clear
       + /Q3 * /Q2 * /Q1 * /Q0
       +  Q3 *  Q0
       +  Q3 *  Q1
       +  Q3 *  Q2

  Q2 :=  Clear
       + /Q2 * /Q1 * /Q0
       +  Q2 *  Q0
       +  Q2 *  Q1

  Q1 :=  Clear
       + /Q1 * /Q0
       +  Q1 *  Q0

  Q0 :=  Clear
       + /Q0

FUNCTION TABLE
OE Clear Clk   /Q0 /Q1 /Q2 /Q3
------------------------------
 L   H    C     L   L   L   L
 L   L    C     H   L   L   L
 L   L    C     L   H   L   L
 L   L    C     H   H   L   L
 L   L    C     L   L   H   L
 L   H    C     L   L   L   L
------------------------------
```

PALASM design of a 4-bit counter.

Though some engineers programmed PAL devices by manually editing files containing the binary fuse pattern data, most opted to design their logic using a hardware description language (HDL) such as Data I/O's ABEL, Logical Devices' CUPL, or MMI's PALASM. These were computer-assisted design (CAD) (now referred to as "electronic design automation") programs which translated (or "compiled") the designers' logic equations into binary fuse map files used to program (and often test) each device.

PALASM

The PALASM (from "PAL assembler") language was used to express boolean equations for the outputs pins in a text file which was then converted to the 'fuse map' file for the programming system using a vendor-supplied program; later the option of translation from schematics became common, and later still, 'fuse maps' could be 'synthesized' from an HDL (hardware description language,) such as Verilog.

The PALASM compiler was written by MMI in FORTRAN IV on an IBM 370/168. MMI made the source code available to users at no cost. By 1983, MMI customers ran versions on the DEC PDP-11, Data General NOVA, Hewlett-Packard HP2100, MDS800 and others.

ABEL

Data I/O Corporation released ABEL.

CUPL

Logical Devices, Inc. released the Universal Compiler for Programmable Logic (CUPL), which ran under MSDOS on the IBM PC and is currently available as an integrated development package for Microsoft Windows.

Device programmers

Popular device programmers included Data I/O Corporation's Model 60A Logic Programmer and Model 2900.

One of the very first PAL Programmers was the Structured Design "SD-20". They had the PALASM software built-in and only required a CRT terminal to enter the equations and view the fuse plots. After fusing, the outputs of the PAL could be verified if test vectors were entered in the source file.

Successors

After MMI succeeded with the 20-pin PAL parts introduced circa 1978, AMD introduced the 24-pin 22V10 PAL with additional features. After buying out MMI (circa 1987), AMD spun off a consolidated operation as Vantis, and that business was acquired by Lattice Semiconductor in 1989.

Altera introduced the EP300 (first CMOS PAL) in 1983 and later moved into the FPGA business.

Lattice Semiconductor introduced the generic array logic (GAL) family in 1985, with functional equivalents of the "V" series PALs that used reprogrammable logic planes based on EEPROM (electrically eraseable programmable read-only memory) technology.

National Semiconductor was a "second source" of GAL parts. AMD introduced a similar family called PALCE. In general one GAL part is able to function as any of the similar family PAL devices. For example the 16V8 GAL is able to replace the 16L8, 16H8, 16H6, 16H4, 16H2 and 16R8 PALs (and many others besides).

ICT (International CMOS Technology) introduced the PEEL 18CV8 in 1986. The 20-pin CMOS EEPROM part could be used in place of any of the registered-output bipolar PALs and used much less power.

Larger-scale programmable logic devices were introduced by Atmel, Lattice Semiconductor, and others. These devices extended the PAL architecture by including multiple logic planes and/or burying logic macrocells within the logic plane(s). The term "complex programmable logic device" (CPLD) was introduced to differentiate these devices from their PAL and GAL predecessors, which were then sometimes referred to as "simple programmable logic devices" or SPLDs.

Another large programmable logic device is the "field-programmable gate array" or FPGA. This term is often used to describe devices currently made by Altera and Xilinx.

Chapter 4

Programmable Logic Device

A **programmable logic device** or PLD is an electronic component used to build reconfigurable digital circuits. Unlike a logic gate, which has a fixed function, a PLD has an undefined function at the time of manufacture. Before the PLD can be used in a circuit it must be programmed, that is, reconfigured.

Using a ROM as a PLD

Before PLDs were invented, read-only memory (ROM) chips were used to create arbitrary combinational logic functions of a number of inputs. Consider a ROM with m inputs (the address lines) and n outputs (the data lines). When used as a memory, the ROM contains 2^m words of n bits each. Now imagine that the inputs are driven not by an m-bit address, but by m independent logic signals. Theoretically, there are 2^m possible Boolean functions of these m signals, but the structure of the ROM allows just 2^n of these functions to be produced at the output pins. The ROM therefore becomes equivalent to n separate logic circuits, each of which generates a chosen function of the m inputs.

The advantage of using a ROM in this way is that any conceivable function of the m inputs can be made to appear at any of the n outputs, making this the most general-purpose combinational logic device available. Also, PROMs (programmable ROMs), EPROMs (ultraviolet-erasable PROMs) and EEPROMs (electrically erasable PROMs) are available that can be programmed using a standard PROM programmer without requiring specialised hardware or software. However, there are several disadvantages:

- they are usually much slower than dedicated logic circuits,
- they cannot necessarily provide safe "covers" for asynchronous logic transitions so the PROM's outputs may glitch as the inputs switch,
- they consume more power,
- they are often more expensive than programmable logic, especially if high speed is required.

Since most ROMs do not have input or output registers, they cannot be used stand-alone for sequential logic. An external TTL register was often used for sequential designs such as state machines. Common EPROMs, for example the 2716, are still sometimes used in this way by hobby circuit designers, who often have some lying around. This use is sometimes called a 'poor man's PAL'.

Early programmable logic

In 1969, Motorola offered the XC157, a mask-programmed gate array with 12 gates and 30 uncommitted input/output pins.

In 1970, Texas Instruments developed a mask-programmable IC based on the IBM read-only associative memory or ROAM. This device, the TMS2000, was programmed by altering the metal layer during the production of the IC. The TMS2000 had up to 17 inputs and 18 outputs with 8 JK flip flop for memory. TI coined the term Programmable Logic Array for this device.

In 1971, General Electric Company (GE) was developing a programmable logic device based on the new PROM technology. This experimental device improved on IBM's ROAM by allowing multilevel logic. Intel had just introduced the floating-gate UV erasable PROM so the researcher at GE incorporated that technology. The GE device was the first erasable PLD ever developed, predating the Altera EPLD by over a decade. GE obtained several early patents on programmable logic devices.

In 1973 National Semiconductor introduced a mask-programmable PLA device (DM7575) with 14 inputs and 8 outputs with no memory registers. This was more popular than the TI part but cost of making the metal mask limited its use. The device is significant because it was the basis for the field programmable logic array produced by Signetics in 1975, the 82S100. (Intersil actually beat Signetics to market but poor yield doomed their part.)

In 1974 GE entered into an agreement with Monolithic Memories to develop a mask-programmable logic device incorporating the GE innovations. The device was named the 'Programmable Associative Logic Array' or PALA. The MMI 5760 was completed in 1976 and could implement multilevel or sequential circuits of over 100 gates. The device was supported by a GE design environment where Boolean equations would be converted to mask patterns for configuring the device. The part was never brought to market.

PAL

MMI introduced a breakthrough device in 1978, the Programmable Array Logic or PAL. The architecture was simpler than that of Signetics FPLA because it omitted the programmable OR array. This made the parts faster, smaller and cheaper. They were available in 20 pin 300 mil DIP packages while the FPLAs came in 28 pin 600 mil packages. The PAL Handbook demystified the design process. The PALASM design software (PAL Assembler) converted the engineers' Boolean equations into the fuse pattern required to program the part. The PAL devices were soon second-sourced by National Semiconductor, Texas Instruments and AMD.

After MMI succeeded with the 20-pin PAL parts, AMD introduced the 24-pin 22V10 PAL with additional features. After buying out MMI (1987), AMD spun off a consolidated operation as Vantis, and that business was acquired by Lattice Semiconductor in 1999.

There are also PLA's : Programmable Logic Array.

GALs

Lattice GAL 16V8 and 20V8

An innovation of the PAL was the **generic array logic** device, or **GAL**, invented by Lattice Semiconductor in 1985. This device has the same logical properties as the PAL but can be erased and reprogrammed. The GAL is very useful in the prototyping stage of a design, when any bugs in the logic can be corrected by reprogramming. GALs are programmed and reprogrammed using a PAL programmer, or by using the in-circuit programming technique on supporting chips.

Lattice GALs combine CMOS and electrically erasable (E^2) floating gate technology for a high-speed, low-power logic device.

A similar device called a **PEEL** (**programmable electrically erasable logic**) was introduced by the International CMOS Technology (ICT) corporation.

CPLDs

PALs and GALs are available only in small sizes, equivalent to a few hundred logic gates. For bigger logic circuits, complex PLDs or CPLDs can be used. These contain the equivalent of several PALs linked by programmable interconnections, all in one integrated circuit. CPLDs can replace thousands, or even hundreds of thousands, of logic gates.

Some CPLDs are programmed using a PAL programmer, but this method becomes inconvenient for devices with hundreds of pins. A second method of programming is to solder the device to its printed circuit board, then feed it with a serial data stream from a personal computer. The CPLD contains a circuit that decodes the data stream and configures the CPLD to perform its specified logic function.

Each manufacturer has a proprietary name for this programming system. For example, Lattice Semiconductor calls it "in-system programming". However, these proprietary systems are beginning to give way to a standard from the Joint Test Action Group, JTAG.

FPGAs

While PALs were busy developing into GALs and CPLDs (all discussed above), a separate stream of development was happening. This type of device is based on gate array technology and is called the field-programmable gate array (FPGA). Early examples of FPGAs are the 82s100 array, and 82S105 sequencer, by Signetics, introduced in the late 1970s. The 82S100 was an array of AND terms. The 82S105 also had flip flop functions.

FPGAs use a grid of logic gates, and once stored, the data doesn't change, similar to that of an ordinary gate array. The term "field-programmable" means the device is programmed by the customer, not the manufacturer.

FPGAs are usually programmed after being soldered down to the circuit board, in a manner similar to that of larger CPLDs. In most larger FPGAs the configuration is volatile, and must be re-loaded into the device whenever power is applied or different functionality is required. Configuration is typically stored in a configuration PROM or EEPROM. EEPROM versions may be in-system programmable (typically via JTAG).

The difference between FPGAs and CPLDs is that FPGAs are internally based on Look-up tables (LUTs) whereas CPLDs form the logic functions with sea-of-gates (e.g. sum of products). CPLDs are meant for simpler designs while FPGAs are meant for more complex designs. In general, CPLDs are a good choice for wide combinational logic applications, whereas FPGAs are more suitable for large state machines (i.e. microprocessors).

Other variants

At present, much interest exists in reconfigurable systems. These are microprocessor circuits that contain some fixed functions and other functions that can be altered by code running on the processor. Designing self-altering systems requires engineers to learn new methods, and that new software tools be developed.

PLDs are being sold now that contain a microprocessor with a fixed function (the so-called *core*) surrounded by programmable logic. These devices let designers concentrate on adding new features to designs without having to worry about making the microprocessor work.

How PLDs retain their configuration

A PLD is a combination of a logic device and a memory device. The memory is used to store the pattern that was given to the chip during programming. Most of the methods for storing data in an integrated circuit have been adapted for use in PLDs. These include:

- Silicon antifuses
- SRAM
- EPROM or EEPROM cells
- Flash memory

Silicon antifuses are the storage elements used in the PAL, the first type of PLD. These are connections that are made by applying a voltage across a modified area of silicon inside the chip. They are called antifuses because they work in the opposite way to normal fuses, which begin life as connections until they are broken by an electric current.

SRAM, or static RAM, is a volatile type of memory, meaning that its contents are lost each time the power is switched off. SRAM-based PLDs therefore have to be programmed every time the circuit is switched on. This is usually done automatically by another part of the circuit.

An EPROM cell is a MOS (metal-oxide-semiconductor) transistor that can be switched on by trapping an electric charge permanently on its gate electrode. This is done by a PAL programmer. The charge remains for many years and can only be removed by exposing the chip to strong ultraviolet light in a device called an EPROM eraser.

Flash memory is non-volatile, retaining its contents even when the power is switched off. It can be erased and reprogrammed as required. This makes it useful for PLD memory.

As of 2005, most CPLDs are electrically programmable and erasable, and non-volatile. This is because they are too small to justify the inconvenience of programming internal SRAM cells every time they start up, and EPROM cells are more expensive due to their ceramic package with a quartz window.

PLD programming languages

Many PAL programming devices accept input in a standard file format, commonly referred to as 'JEDEC files'.They are analogous to software compilers. The languages used as source code for logic compilers are called hardware description languages, or HDLs.

PALASM and ABEL are frequently used for low-complexity devices, while Verilog and VHDL are popular higher-level description languages for more complex devices. The more limited ABEL is often used for historical reasons, but for new designs VHDL is more popular, even for low-complexity designs.

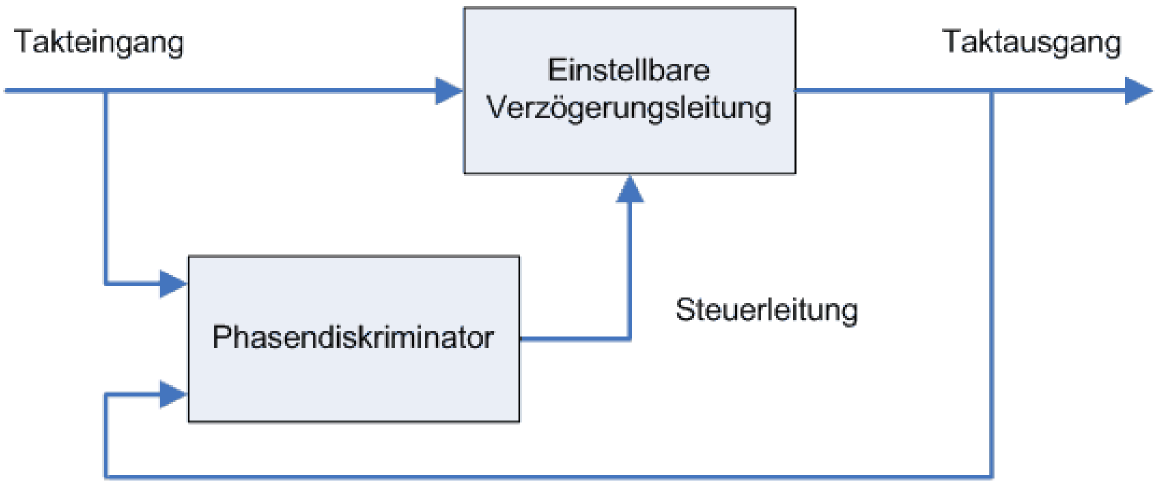
PLD programming devices

A device programmer is used to transfer the boolean logic pattern into the programmable device. In the early days of programmable logic, every PLD manufacturer also produced a specialized device programmer for its family of logic devices. Later, universal device programmers came onto the market that supported several logic device families from different manufacturers. Today's device programmers usually can program common PLDs (mostly PAL/GAL equivalents) from all existing manufacturers. Common file formats used to store the boolean logic pattern (fuses) are JEDEC, Altera POF (Programmable Object File), or Xilinx BITstream.

Chapter 5

Delay-Locked Loop & Complex Programmable Logic Device

Delay-Locked Loop



In electronics, a **delay-locked loop** (DLL) is a digital circuit similar to a phase-locked loop (PLL), with the main difference being the absence of an internal voltage-controlled oscillator. A DLL can be used to change the phase of a clock signal (a signal with a periodic waveform), usually to enhance the *clock rise*-to-*data output valid* timing characteristics of integrated circuits (such as DRAM devices). DLLs can also be used for clock recovery (CDR). From the outside, a DLL can be seen as a negative-delay gate placed in the clock path of a digital circuit.

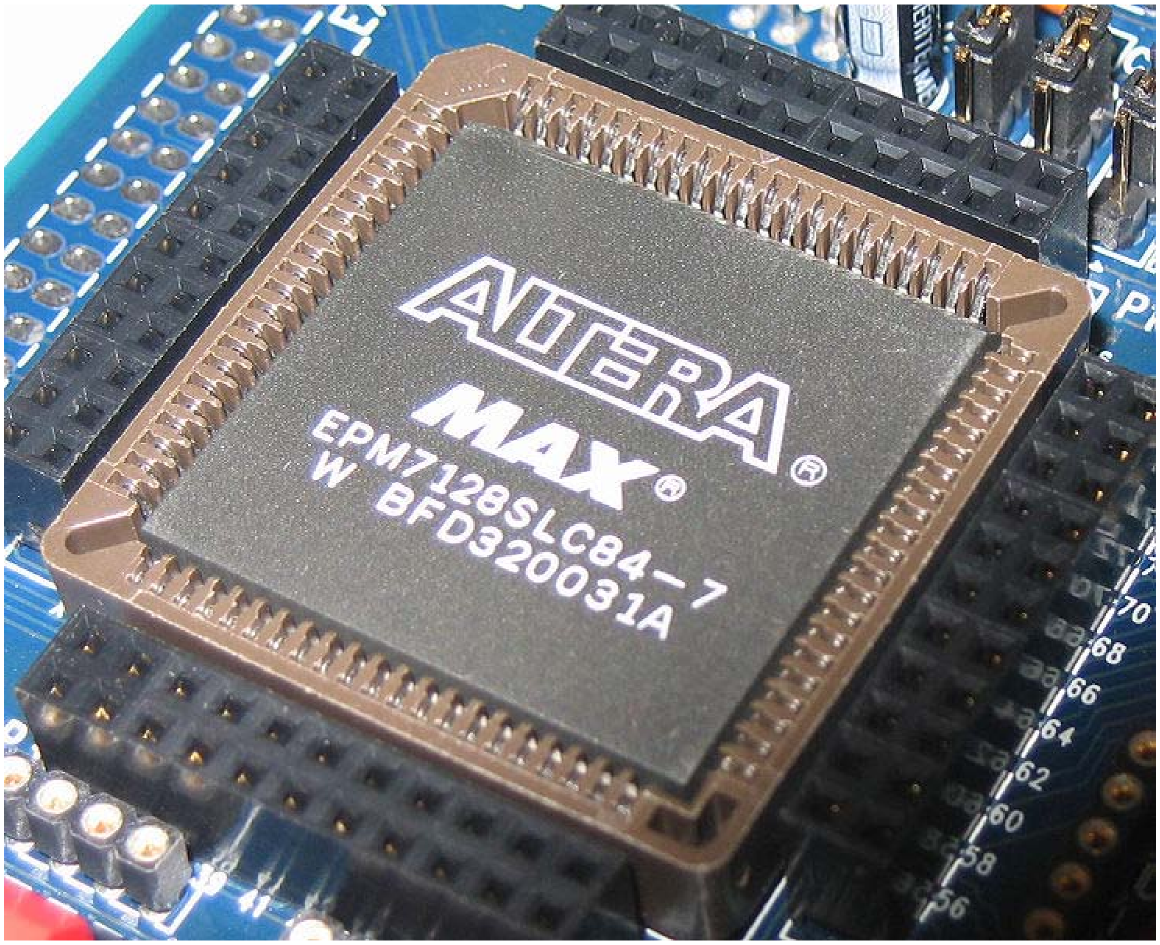
Another way to view the difference between a DLL and a PLL is that a DLL is a first order loop and a PLL is a second order loop. A DLL compares the phase of one of its outputs to the input clock to generate an error signal which is then integrated and fed back as the control to all of the delay elements. The integration allows the error to go to zero while keeping the control signal, and thus the delays, where they need to be for phase lock. Since the control signal directly impacts the phase this is all that is required. A PLL compares the phase of its oscillator with the incoming signal to generate an error signal which is then integrated to create a control signal for the voltage-controlled oscillator. The control signal impacts the frequency of the oscillator, and phase is the integral of frequency, so a second integration is unavoidably performed by the oscillator itself. A first order feedback system is significantly easier to stabilize than a second order feedback system, which is a major advantage of DLLs.

The main component of a DLL is a delay chain composed of many delay gates connected front-to-back. The input of the chain (and thus of the DLL) is connected to the clock that is to be negatively delayed. A multiplexer is connected to each stage of the delay chain; the selector of this multiplexer is automatically updated by a control circuit to produce the negative delay effect. The output of the DLL is the resulting, negatively delayed clock signal.

The phase shift can be specified either in absolute terms (in delay chain gate units), or as a proportion of the clock period, or both.

Compared to phase-locked loops, delay-locked loops are a relatively recent innovation, first found in Dr. Combes' work in the early 1990s, then popularized by Xilinx in their Virtex family of FPGA products.

Complex Programmable Logic Device



An Altera MAX 7000-series CPLD with 2500 gates.

A **complex programmable logic device (CPLD)** is a programmable logic device with complexity between that of PALs and FPGAs, and architectural features of both. The building block of a CPLD is the macrocell, which contains logic implementing disjunctive normal form expressions and more specialized logic operations.

Features in common with PALs:

- Non-volatile configuration memory. Unlike many FPGAs, an external configuration ROM isn't required, and the CPLD can function immediately on system start-up.
- For many legacy CPLD devices, routing constrains most logic blocks to have input and output signals connected to external pins, reducing opportunities for internal state storage and deeply layered logic. This is usually not a factor for larger CPLDs and newer CPLD product families.

Features in common with FPGAs:

- Large number of gates available. CPLDs typically have the equivalent of thousands to tens of thousands of logic gates, allowing implementation of moderately complicated data processing devices. PALs typically have a few hundred gate equivalents at most, while FPGAs typically range from tens of thousands to several million.
- Some provisions for logic more flexible than sum-of-product expressions, including complicated feedback paths between macro cells, and specialized logic for implementing various commonly-used functions, such as integer arithmetic.

The most noticeable difference between a large CPLD and a small FPGA is the presence of on-chip non-volatile memory in the CPLD. This distinction is rapidly becoming less relevant, as several of the latest FPGA products also offer models with embedded configuration memory.

The characteristic of non-volatility makes the CPLD the device of choice in modern digital designs to perform 'boot loader' functions before handing over control to other devices not having this capability. A good example is where a CPLD is used to load configuration data for an FPGA from non-volatile memory.

CPLDs were an evolutionary step from even smaller devices that preceded them, PLAs (first shipped by Signetics), and PALs. These in turn were preceded by standard logic products, that offered no programmability and were "programmed" by wiring several standard logic chips together.

The main distinction between FPGA and CPLD device architectures is that FPGAs are internally based on Look-up tables (LUTs) while CPLDs form the logic functions with sea-of-gates (e.g. sum of products).

Chapter 6

Partial Re-Configuration & Rent's Rule

Partial Re-Configuration

Partial Reconfiguration is the process of configuring a portion of a field programmable gate array while the other part is still running/operating.

Hardware, like software, can be designed modularly, by creating subcomponents and then higher-level components to instantiate them. In many cases it is useful to be able to swap out one or several of these subcomponents while the FPGA is still operating.

Normally, reconfiguring an FPGA requires it to be held in reset while an external controller reloads a design onto it. Partial reconfiguration allows for critical parts of the design to continue operating while a controller either on the FPGA or off of it loads a partial design into a reconfigurable module. Partial reconfiguration also can be used to save space for multiple designs by only storing the partial designs that change between designs.

A common example for when partial reconfiguration would be useful is the case of a communication device. If the device is controlling multiple connections, some of which require encryption, it would be useful to be able to load different encryption cores without bringing the whole controller down.

Partial reconfiguration is not supported on all FPGAs. A special software flow with emphasis on modular design is required. Typically the design modules are built along well defined boundaries inside the FPGA that require the design to be specially mapped to the internal hardware.

From the functionality of the design, partial reconfiguration can be divided into two groups:

- **dynamic partial reconfiguration**, also known as an **active** partial reconfiguration - permits to change the part of the device while the rest of an FPGA is still running;
- **static partial reconfiguration** - the device is not active during the reconfiguration process. While the partial data is sent into the FPGA, the rest of the device is stopped (in the shutdown mode) and brought up after the configuration is completed.

There are two styles of partial reconfiguration of FPGA devices from Xilinx: **module-based** and **difference-based**.

Module-based partial reconfiguration permits to reconfigure distinct modular parts of the design. To ensure the communication across the reconfigurable module boundaries, special bus macros ought to be prepared. It works as a fixed routing bridge that connects the reconfigurable module with the rest part of the design. Module-based partial reconfiguration requires to perform a set of specific guidelines during at the stage of design specification. Finally for each reconfigurable module of the design, separate bit-stream is created. Such a bit-stream is used to perform the partial reconfiguration of an FPGA.

Difference-based partial reconfiguration can be used when a small change is made to the design. It is especially useful in case of changing LUT equations or dedicated memory blocks content. The partial bit-stream contains only information about differences between the current design structure (that resides in the FPGA) and the new content of an FPGA. There are two ways of difference-based reconfiguration known as a front-end and back-end. The first one is based on the modification of the design in the hardware description languages (HDLs). It is clear that such a solution requires full repeating of the synthesis and implementation processes. The back-end difference-based partial reconfiguration permits to make changes at the implementation stage of the prototyping flow. Therefore there is no need for re-synthesis of the design. The usage of both methods (either front-end and back-end) leads to creation of a partial bit-stream that can be used for a partial reconfiguration of the FPGA.

Rent's Rule

Rent's rule pertains to the organization of computing logic, specifically the relationship between the number of external signal connections to a logic block (i.e., the number of "pins") with the number of logic gates in the logic block, and has been applied to circuits ranging from small digital circuits to mainframe computers.

E.F. Rent's discovery and first publications

In the 1960's, E.F. Rent, an IBM employee, found a remarkable trend between the number of pins (terminals T) at the boundaries of integrated circuit designs at IBM and the number of internal components (g), such as logic gates or standard cells. On a log − log plot, these datapoints were on a straight line, implying a power-law relation $T = tg^p$

where t and p are constants ($p < 1.0$, and generally $0.5 < p < 0.8$). Rent disclosed his findings in IBM-internal memoranda, but the relation was described in 1971 by Landman and Russo. They performed a hierarchical circuit partitioning in such a way that at each hierarchical level (top-down) the least number of interconnections had to be cut to partition the circuit (in more or less equal parts). At each partitioning step, they noted the number of terminals and the number of components in each partition and then partitioned the sub-partitions further. They found the power law rule applied to the resulting T versus g plot and named it "Rent's rule". It is crucial to recognise that Rent's rule is an empirical result based on observations of existing designs, and therefore it is less applicable to the analysis of non-traditional circuit architectures. Having said that, it does provide a useful framework with which to compare similar architectures.

Theoretical basis

Christie and Stroobandt later derived Rent's rule theoretically for homogeneous systems and pointed out that the amount of optimization achieved in placement is reflected by the parameter p, the "Rent exponent", which also depends on the circuit topology. In particular, values $p < 1$ correspond to a greater fraction of short interconnects. The constant t in Rent's rule can be viewed as the average number of terminals required by a single logic block since $T = t$ when $g = 1$.

Special cases and applications

Random arrangement of logic blocks typically have $p = 1$. Larger values are impossible since the maximum number of terminals for any region containing g logic components in a homogeneous system is given by $T = tg$. Lower bounds on p depend on the interconnection topology since it is generally impossible to make all wires short. This lower bound p * is often called the "intrinsic Rent exponent", a notion first introduced by Hagen et al. It can be used to characterize optimal placements and also measure the interconnection complexity of a circuit. Higher (intrinsic) Rent exponent values correspond to a higher topological complexity. One extreme example ($p = 0$) is a long chain of logic blocks, while a clique has $p = 1$. In realistic 2D circuits, p * ranges from 0.5 for highly-regular circuits (such as SRAM) to 0.75 for random logic.

System performance analysis tools such as BACPAC typically use Rent's rule to calculate expected wiring lengths and wiring demands.

Estimating Rent's exponent

To estimate Rent's exponent, one can use top-down partitioning, as used in min-cut placement. For every partition, count the number of terminals connected to the partition and compare it to the number of logic blocks in the partition. Rent's exponent can then be found by fitting these datapoints on a log-log plot, resulting in an exponent p'. For optimally partitioned circuits, $p' = p$ * but this is no longer the case for practical (heuristic) partitioning approaches. For partitioning-based placement algorithms $p^* \leq p' \leq p$.

Region II of Rent's rule

Landman and Russo found a deviation of Rent's rule near the "far end", i.e., for partitions with a large number of blocks, which is known as "Region II" of Rent's Rule . A similar deviation exists at for small partitions, and has been found by Stroobandt who called it Region III.

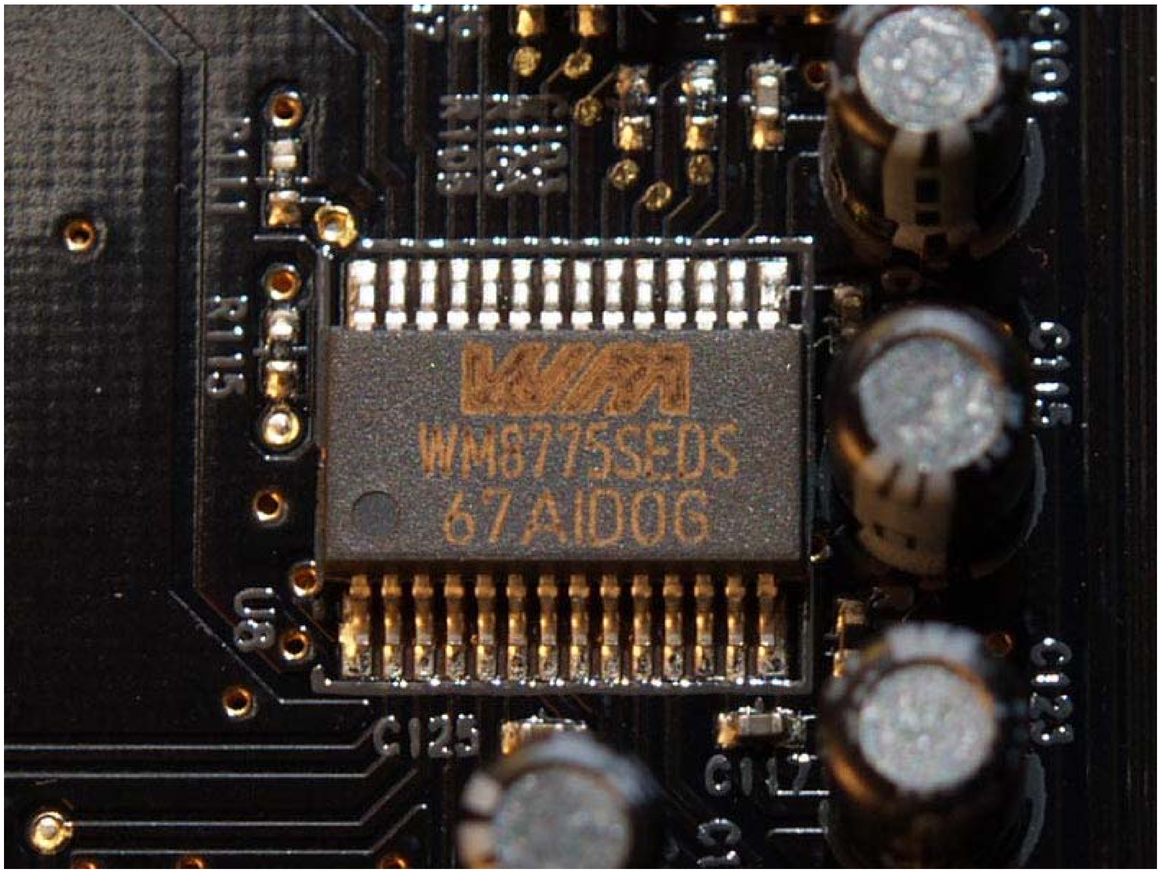
Rentian wirelength estimation

Another IBM employee, Donath, discovered that Rent's rule can be used to estimate the average wirelength and the wirelength distribution in VLSI chips. This motivated the System Level Interconnect Prediction workshop, founded in 1999, and an entire community working on wirelength prediction. The resulting wirelength estimates have been improved significantly since then and are now used for "technology exploration." The use of Rent's rule allows to perform such estimates *a priori* (i.e., before actual placement) and thus predict the properties of future technologies (clock frequencies, number of routing layers needed, area, power) based on limited information about future circuits and technologies.

A comprehensive overview of work based on Rent's rule has been published by Stroobandt.

Chapter 7

Analog-to-Digital Converter



4-channel stereo multiplexed analog-to-digital converter WM8775SEDS made by Wolfson Microelectronics placed on a X-Fi Fatal1ty Pro sound card.

An **analog-to-digital converter** (abbreviated **ADC**, **A/D** or **A to D**) is a device which converts a continuous quantity to a discrete time digital representation. An ADC may also provide an isolated measurement. The reverse operation is performed by a digital-to-analog converter (**DAC**).

Typically, an ADC is an electronic device that converts an input analog voltage or current to a digital number proportional to the magnitude of the voltage or current. However, some non-electronic or only partially electronic devices, such as rotary encoders, can also be considered ADCs.

The digital output may use different coding schemes. Typically the digital output will be a two's complement binary number that is proportional to the input, but there are other possibilities. An encoder, for example, might output a Gray code.

Concepts

Resolution

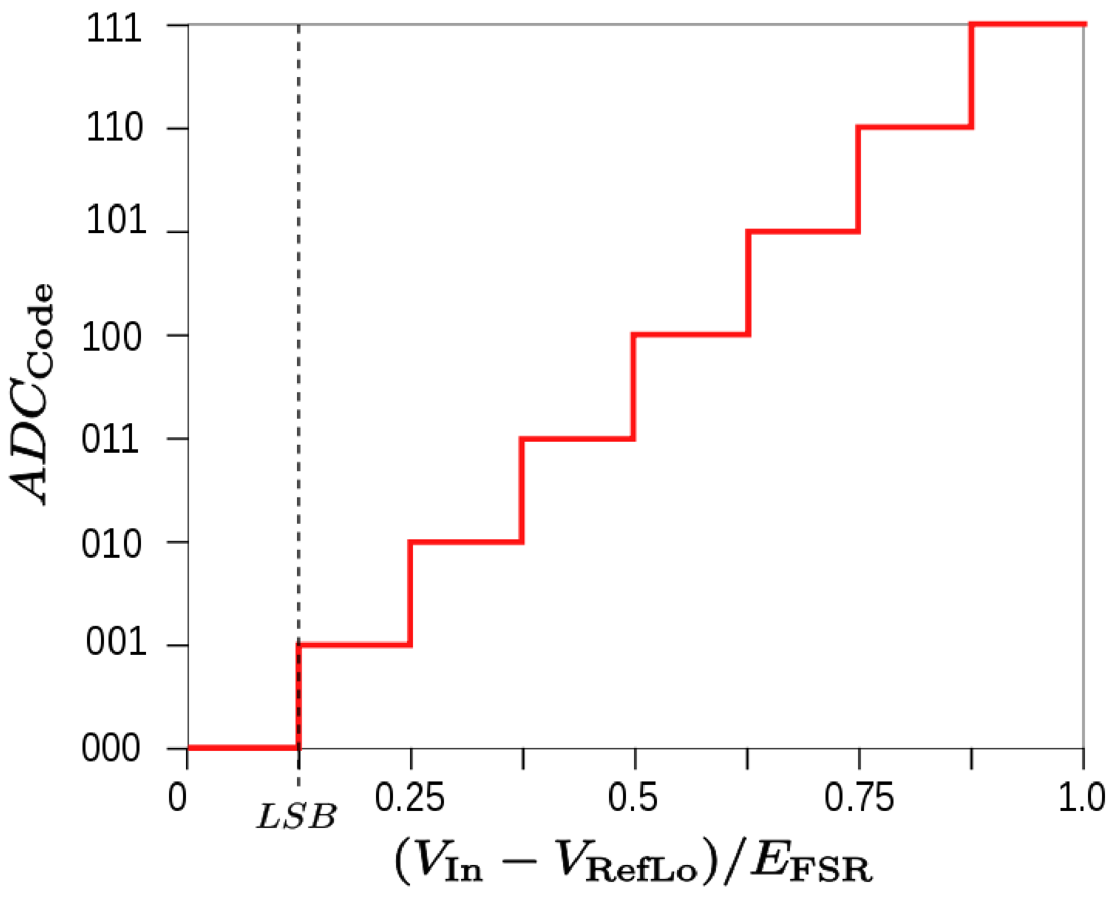


Fig. 1. An 8-level ADC coding scheme.

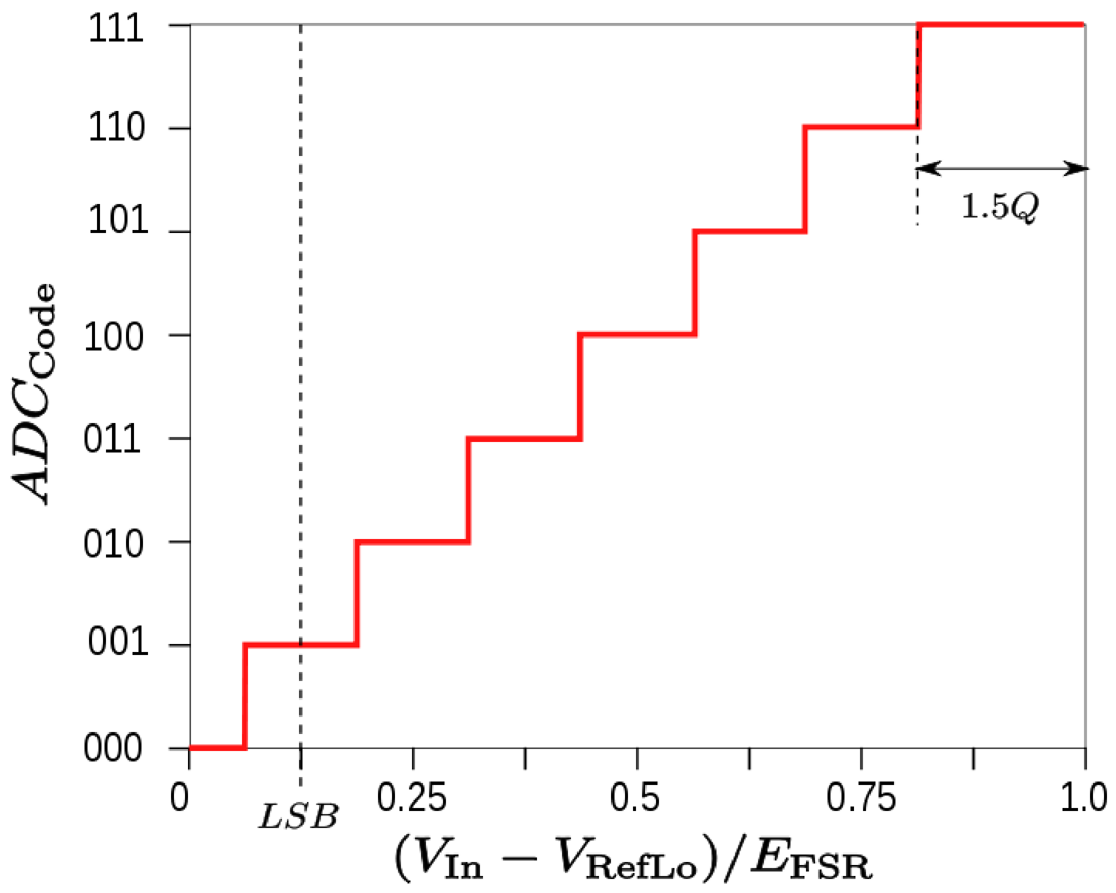


Fig. 2. An 8-level ADC coding scheme. As in figure 1 but with mid-tread coding.

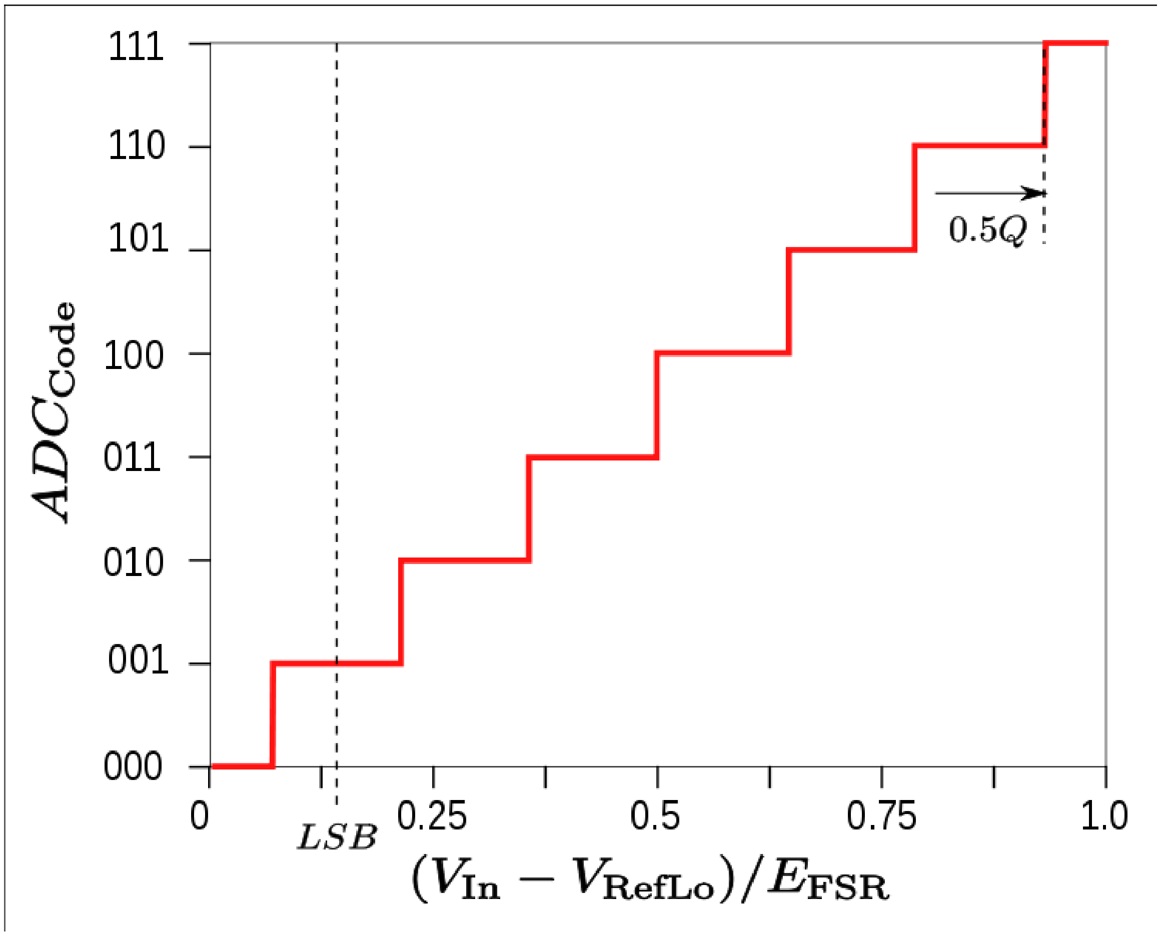


Fig. 3. An 8-level ADC mid-tread coding scheme. As in figure 2 but with equal half-*LSB* intervals at the highest and lowest codes. Note that *LSB* is now slightly larger than in figures 1 and 2.

The resolution of the converter indicates the number of discrete values it can produce over the range of analog values. The values are usually stored electronically in binary form, so the resolution is usually expressed in bits. In consequence, the number of discrete values available, or "levels", is usually a power of two. For example, an ADC with a resolution of 8 bits can encode an analog input to one in 256 different levels, since $2^8 = 256$. The values can represent the ranges from 0 to 255 (i.e. unsigned integer) or from −128 to 127 (i.e. signed integer), depending on the application.

Resolution can also be defined electrically, and expressed in volts. The minimum change in voltage required to guarantee a change in the output code level is called the *LSB* (least significant bit, since this is the voltage represented by a change in the LSB). The resolution Q of the ADC is equal to the *LSB* voltage. The voltage resolution of an ADC is equal to its overall voltage measurement range divided by the number of discrete voltage intervals:

$$Q = \frac{E_{\mathrm{FSR}}}{N},$$

where N is the number of voltage intervals and E_{FSR} is the full scale voltage range. E_{FSR} is given by

$$E_{\mathrm{FSR}} = V_{\mathrm{RefHi}} - V_{\mathrm{RefLow}},$$

where V_{RefHi} and V_{RefLow} are the upper and lower extremes, respectively, of the voltages that can be coded.

Normally, the number of voltage intervals is given by

$$N = 2^M,$$

where M is the ADC's resolution in bits.

That is, one voltage interval is assigned per code level. However, figure 3 shows a situation where

$$N = 2^M - 1$$

Some examples:

- Example 1
  - Coding scheme as in figure 1
  - Full scale measurement range = 0 to 10 volts
  - ADC resolution is 12 bits: $2^{12} = 4096$ quantization levels (codes)
  - ADC voltage resolution, $Q = (10\text{ V} - 0\text{ V}) / 4096 = 10\text{ V} / 4096 \approx 0.00244\text{ V} \approx 2.44\text{ mV}$.

- Example 2
  - Coding scheme as in figure 2
  - Full scale measurement range = -10 to +10 volts
  - ADC resolution is 14 bits: $2^{14} = 16384$ quantization levels (codes)
  - ADC voltage resolution is, $Q = (10\text{ V} - (-10\text{ V})) / 16384 = 20\text{ V} / 16384 \approx 0.00122\text{ V} \approx 1.22\text{ mV}$.

- Example 3
  - Coding scheme as in figure 3
  - Full scale measurement range = 0 to 7 volts
  - ADC resolution is 3 bits: $2^3 = 8$ quantization levels (codes)
  - ADC voltage resolution is, $Q = (7\text{ V} - 0\text{ V})/7 = 7\text{ V}/7 = 1\text{ V} = 1000\text{ mV}$

In most ADCs, the smallest output code ("0" in an unsigned system) represents a voltage range which is $0.5Q$, that is, half the ADC voltage resolution (Q). The largest code represents a range of 1.5Q as in figure 2 (if this were 0.5Q also, the result would be as figure 3). The other $N - 2$ codes are all equal in width and represent the ADC voltage resolution (Q) calculated above. Doing this centers the code on an input voltage that represents the M th division of the input voltage range. This practice is called "mid-tread" operation. This type of ADC can be modeled mathematically as:

$$ADC_{\text{Code}} = \text{round}\left(\left(\frac{2^M}{V_{\text{RefHi}} - V_{\text{RefLow}}}\right) \cdot (V_{\text{In}} - V_{\text{RefLow}})\right)$$

The exception to this convention seems to be the Microchip PIC processor, where all M steps are equal width, as shown in figure 1. This practice is called "Mid-Rise with Offset" operation.

$$ADC_{\text{Code}} = \text{floor}\left(\left(\frac{2^M}{V_{\text{RefHi}} - V_{\text{RefLow}}}\right) \cdot (V_{\text{In}} - V_{\text{RefLow}})\right)$$

In practice, the useful resolution of a converter is limited by the best signal-to-noise ratio (SNR) that can be achieved for a digitized signal. An ADC can resolve a signal to only a certain number of bits of resolution, called the effective number of bits (ENOB). One effective bit of resolution changes the signal-to-noise ratio of the digitized signal by 6 dB, if the resolution is limited by the ADC. If a preamplifier has been used prior to A/D conversion, the noise introduced by the amplifier can be an important contributing factor towards the overall SNR.

Response type

Linear ADCs

Most ADCs are of a type known as linear The term *linear* implies here the range of the input values that map to each output value has a linear relationship with the output value, i.e., that the output value k is used for the range of input values from

$m(k + b)$

to

$m(k + 1 + b)$,

where m and b are constants. Here b is typically 0 or −0.5. When $b = 0$, the ADC is referred to as *mid-rise*, and when $b = -0.5$ it is referred to as *mid-tread*.

Non-linear ADCs

If the probability density function of a signal being digitized is uniform, then the signal-to-noise ratio relative to the quantization noise is the best possible. Because this is often not the case, it is usual to pass the signal through its cumulative distribution function (CDF) before the quantization. This is good because the regions that are more important get quantized with a better resolution. In the dequantization process, the inverse CDF is needed.

This is the same principle behind the companders used in some tape-recorders and other communication systems, and is related to entropy maximization.

For example, a voice signal has a Laplacian distribution. This means that the region around the lowest levels, near 0, carries more information than the regions with higher amplitudes. Because of this, logarithmic ADCs are very common in voice communication systems to increase the dynamic range of the representable values while retaining fine-granular fidelity in the low-amplitude region.

An eight-bit A-law or the μ-law logarithmic ADC covers the wide dynamic range and has a high resolution in the critical low-amplitude region, that would otherwise require a 12-bit linear ADC.

Accuracy

An ADC has several sources of errors. Quantization error and (assuming the ADC is intended to be linear) non-linearity are intrinsic to any analog-to-digital conversion. There is also a so-called *aperture error* which is due to a clock jitter and is revealed when digitizing a time-variant signal (not a constant value).

These errors are measured in a unit called the *LSB*, which is an abbreviation for least significant bit. In the above example of an eight-bit ADC, an error of one LSB is 1/256 of the full signal range, or about 0.4%.

Quantization error

Quantization error (or quantization noise) is the difference between the original signal and the digitized signal. Hence, The magnitude of the quantization error at the sampling instant is between zero and half of one LSB. Quantization error is due to the finite resolution of the digital representation of the signal, and is an unavoidable imperfection in all types of ADCs.

Non-linearity

All ADCs suffer from non-linearity errors caused by their physical imperfections, causing their output to deviate from a linear function (or some other function, in the case of a deliberately non-linear ADC) of their input. These errors can sometimes be mitigated by calibration, or prevented by testing.

Important parameters for linearity are integral non-linearity (INL) and differential non-linearity (DNL). These non-linearities reduce the dynamic range of the signals that can be digitized by the ADC, also reducing the effective resolution of the ADC.

Aperture error

Imagine that we are digitizing a sine wave $x(t) = A\sin(2\pi f_0 t)$. Provided that the actual sampling time *uncertainty* due to the *clock jitter* is Δt, the error caused by this phenomenon can be estimated as $E_{ap} \leq |x'(t)\Delta t| \leq 2A\pi f_0 \Delta t$.

The error is zero for DC, small at low frequencies, but significant when high frequencies have high amplitudes. This effect can be ignored if it is drowned out by the *quantizing error*. Jitter requirements can be calculated using the following formula: $\Delta t < \frac{1}{2^q \pi f_0}$, where q is a number of ADC bits.

ADC resolution in bit	**input frequency**						
	1 Hz	**44.1 kHz**	**192 kHz**	**1 MHz**	**10 MHz**	**100 MHz**	**1 GHz**
8	1243 µs	28.2 ns	6.48 ns	1.24 ns	124 ps	12.4 ps	1.24 ps
10	311 µs	7.05 ns	1.62 ns	311 ps	31.1 ps	3.11 ps	0.31 ps
12	77.7 µs	1.76 ns	405 ps	77.7 ps	7.77 ps	0.78 ps	0.08 ps
14	19.4 µs	441 ps	101 ps	19.4 ps	1.94 ps	0.19 ps	0.02 ps
16	4.86 µs	110 ps	25.3 ps	4.86 ps	0.49 ps	0.05 ps	–
18	1.21 µs	27.5 ps	6.32 ps	1.21 ps	0.12 ps	–	–
20	304 ns	6.88 ps	1.58 ps	0.16 ps	–	–	–
24	19.0 ns	0.43 ps	0.10 ps	–	–	–	–
32	74.1 ps	–	–	–	–	–	–

This table shows, for example, that it is not worth using a precise 24-bit ADC for sound recording if there is not an *ultra low jitter* clock. One should consider taking this phenomenon into account before choosing an ADC.

Clock jitter is caused by phase noise. The resolution of ADCs with a digitization bandwidth between 1 MHz and 1 GHz is limited by jitter.

When sampling audio signals at 44.1 kHz, the anti-aliasing filter should have eliminated all frequencies above 22 kHz. The input frequency (in this case, 22 kHz), not the ADC clock frequency, is the determining factor with respect to jitter performance.

Sampling rate

The analog signal is continuous in time and it is necessary to convert this to a flow of digital values. It is therefore required to define the rate at which new digital values are sampled from the analog signal. The rate of new values is called the *sampling rate* or *sampling frequency* of the converter.

A continuously varying bandlimited signal can be sampled (that is, the signal values at intervals of time T, the sampling time, are measured and stored) and then the original signal can be *exactly* reproduced from the discrete-time values by an interpolation formula. The accuracy is limited by quantization error. However, this faithful reproduction is only possible if the sampling rate is higher than twice the highest frequency of the signal. This is essentially what is embodied in the Shannon-Nyquist sampling theorem.

Since a practical ADC cannot make an instantaneous conversion, the input value must necessarily be held constant during the time that the converter performs a conversion (called the *conversion time*). An input circuit called a sample and hold performs this task—in most cases by using a capacitor to store the analog voltage at the input, and using an electronic switch or gate to disconnect the capacitor from the input. Many ADC integrated circuits include the sample and hold subsystem internally.

Aliasing

All ADCs work by sampling their input at discrete intervals of time. Their output is therefore an incomplete picture of the behaviour of the input. There is no way of knowing, by looking at the output, what the input was doing between one sampling instant and the next. If the input is known to be changing slowly compared to the sampling rate, then it can be assumed that the value of the signal between two sample instants was somewhere between the two sampled values. If, however, the input signal is changing rapidly compared to the sample rate, then this assumption is not valid.

If the digital values produced by the ADC are, at some later stage in the system, converted back to analog values by a digital to analog converter or DAC, it is desirable that the output of the DAC be a faithful representation of the original signal. If the input signal is changing much faster than the sample rate, then this will not be the case, and spurious signals called *aliases* will be produced at the output of the DAC. The frequency of the aliased signal is the difference between the signal frequency and the sampling rate. For example, a 2 kHz sine wave being sampled at 1.5 kHz would be reconstructed as a 500 Hz sine wave. This problem is called *aliasing*.

To avoid aliasing, the input to an ADC must be low-pass filtered to remove frequencies above half the sampling rate. This filter is called an *anti-aliasing* filter, and is essential for a practical ADC system that is applied to analog signals with higher frequency content.

Although aliasing in most systems is unwanted, it should also be noted that it can be exploited to provide simultaneous down-mixing of a band-limited high frequency signal.

Dither

In A-to-D converters, performance can usually be improved using dither. This is a very small amount of random noise (white noise) which is added to the input before conversion. Its amplitude is set to be twice the value of the least significant bit. Its effect is to cause the state of the LSB to randomly oscillate between 0 and 1 in the presence of very low levels of input, rather than sticking at a fixed value. Rather than the signal simply getting cut off altogether at this low level (which is only being quantized to a resolution of 1 bit), it extends the effective range of signals that the A-to-D converter can convert, at the expense of a slight increase in noise - effectively the quantization error is diffused across a series of noise values which is far less objectionable than a hard cutoff. The result is an accurate representation of the signal over time. A suitable filter at the output of the system can thus recover this small signal variation.

An audio signal of very low level (with respect to the bit depth of the ADC) sampled without dither sounds extremely distorted and unpleasant. Without dither the low level may cause the least significant bit to "stick" at 0 or 1. With dithering, the true level of the audio may be calculated by averaging the actual quantized sample with a series of other samples [the dither] that are recorded over time.

A virtually identical process, also called dither or dithering, is often used when quantizing photographic images to a fewer number of bits per pixel—the image becomes noisier but to the eye looks far more realistic than the quantized image, which otherwise becomes banded. This analogous process may help to visualize the effect of dither on an analogue audio signal that is converted to digital.

Dithering is also used in integrating systems such as electricity meters. Since the values are added together, the dithering produces results that are more exact than the LSB of the analog-to-digital converter.

Note that dither can only increase the resolution of a sampler, it cannot improve the linearity, and thus accuracy does not necessarily improve.

Oversampling

Usually, signals are sampled at the minimum rate required, for economy, with the result that the quantization noise introduced is white noise spread over the whole pass band of the converter. If a signal is sampled at a rate much higher than the Nyquist frequency and then digitally filtered to limit it to the signal bandwidth there are the following advantages:

- digital filters can have better properties (sharper rolloff, phase) than analogue filters, so a sharper anti-aliasing filter can be realised and then the signal can be downsampled giving a better result
- a 20-bit ADC can be made to act as a 24-bit ADC with 256× oversampling
- the signal-to-noise ratio due to quantization noise will be higher than if the whole available band had been used. With this technique, it is possible to obtain an effective resolution larger than that provided by the converter alone
- The improvement in SNR is 3 dB (equivalent to 0.5 bits) per octave of oversampling which is not sufficient for many applications. Therefore, oversampling is usually coupled with noise shaping. With noise shaping, the improvement is 6L+3 dB per octave where L is the order of loop filter used for noise shaping. e.g. - a 2nd order loop filter will provide an improvement of 15 dB/octave.

Relative speed and precision

The speed of an ADC varies by type. The Wilkinson ADC is limited by the clock rate which is processable by current digital circuits. Currently, frequencies up to 300 MHz are possible. The conversion time is directly proportional to the number of channels. For a successive approximation ADC, the conversion time scales with the logarithm of the number of channels. Thus for a large number of channels, it is possible that the successive approximation ADC is faster than the Wilkinson. However, the time consuming steps in the Wilkinson are digital, while those in the successive approximation are analog. Since analog is inherently slower than digital, as the number of channels increases, the time required also increases. Thus there are competing processes at work. Flash ADCs are certainly the fastest type of the three. The conversion is basically performed in a single parallel step. For an 8-bit unit, conversion takes place in a few tens of nanoseconds.

There is, as expected, somewhat of a trade off between speed and precision. Flash ADCs have drifts and uncertainties associated with the comparator levels, which lead to poor uniformity in channel width. Flash ADCs have a resulting poor linearity. For successive approximation ADCs, poor linearity is also apparent, but less so than for flash ADCs. Here, non-linearity arises from accumulating errors from the subtraction processes. Wilkinson ADCs are the best of the three. These have the best differential non-linearity. The other types require channel smoothing in order to achieve the level of the Wilkinson.

The sliding scale principle

The sliding scale or randomizing method can be employed to greatly improve the channel width uniformity and differential linearity of any type of ADC, but especially flash and successive approximation ADCs. Under normal conditions, a pulse of a particular amplitude is always converted to a certain channel number. The problem lies in that channels are not always of uniform width, and the differential linearity decreases proportionally with the divergence from the average width. The sliding scale principle uses an averaging effect to overcome this phenomenon. A random, but known analog voltage is added to the input pulse. It is then converted to digital form, and the equivalent digital version is subtracted, thus restoring it to its original value. The advantage is that the conversion has taken place at a random point. The statistical distribution of the final channel numbers is decided by a weighted average over a region of the range of the ADC. This in turn desensitizes it to the width of any given channel.

ADC structures

These are the most common ways of implementing an electronic ADC:

- A **direct conversion ADC** or **flash ADC** has a bank of comparators sampling the input signal in parallel, each firing for their decoded voltage range. The comparator bank feeds a logic circuit that generates a code for each voltage range. Direct conversion is very fast, capable of gigahertz sampling rates, but usually has only 8 bits of resolution or fewer, since the number of comparators needed, $2^N - 1$, doubles with each additional bit, requiring a large expensive circuit. ADCs of this type have a large die size, a high input capacitance, high power dissipation, and are prone to produce glitches on the output (by outputting an out-of-sequence code). Scaling to newer submicrometre technologies does not help as the device mismatch is the dominant design limitation. They are often used for video, wideband communications or other fast signals in optical storage.

- A **successive-approximation ADC** uses a comparator to reject ranges of voltages, eventually settling on a final voltage range. Successive approximation works by constantly comparing the input voltage to the output of an internal digital to analog converter (DAC, fed by the current value of the approximation) until the best approximation is achieved. At each step in this process, a binary value of the approximation is stored in a successive approximation register (SAR). The SAR uses a reference voltage (which is the largest signal the ADC is to convert) for comparisons. For example if the input voltage is 60 V and the reference voltage is 100 V, in the 1st clock cycle, 60 V is compared to 50 V (the reference, divided by two. This is the voltage at the output of the internal DAC when the input is a '1' followed by zeros), and the voltage from the comparator is positive (or '1') (because 60 V is greater than 50 V). At this point the first binary digit (MSB) is set to a '1'. In the 2nd clock cycle the input voltage is compared to 75 V (being halfway between 100 and 50 V: This is the output of the internal DAC when its input is '11' followed by zeros) because 60 V is less than 75 V, the comparator output is now negative (or '0'). The second binary digit is therefore set to a '0'. In the 3rd clock cycle, the input voltage is compared with 62.5 V (halfway between 50 V and 75 V: This is the output of the internal DAC when its input is '101' followed by zeros). The output of the comparator is negative or '0' (because 60 V is less than 62.5 V) so the third binary digit is set to a 0. The fourth clock cycle similarly results in the fourth digit being a '1' (60 V is greater than 56.25 V, the DAC output for '1001' followed by zeros). The result of this would be in the binary form 1001. This is also called *bit-weighting conversion*, and is similar to a binary search. The analogue value is rounded to the nearest binary value below, meaning this converter type is mid-rise. Because the approximations are successive (not simultaneous), the conversion takes one clock-cycle for each bit of resolution desired. The clock frequency must be equal to the sampling frequency multiplied by the number of bits of resolution desired. For example, to sample audio at 44.1 kHz with 32 bit resolution, a clock frequency of over 1.4 MHz would be required. ADCs of this type have good resolutions and quite wide ranges. They are more complex than some other designs.

- A **ramp-compare ADC** produces a saw-tooth signal that ramps up or down then quickly returns to zero. When the ramp starts, a timer starts counting. When the ramp voltage matches the input, a comparator fires, and the timer's value is recorded. Timed ramp converters require the least number of transistors. The ramp time is sensitive to temperature because the circuit generating the ramp is often just some simple oscillator. There are two solutions: use a clocked counter driving a DAC and then use the comparator to preserve the counter's value, or calibrate the timed ramp. A special advantage of the ramp-compare system is that comparing a second signal just requires another comparator, and another register to store the voltage value. A very simple (non-linear) ramp-converter can be implemented with a microcontroller and one resistor and capacitor. Vice versa, a filled capacitor can be taken from an integrator, time-to-amplitude converter, phase detector, sample and hold circuit, or peak and hold circuit and discharged. This has the advantage that a slow comparator cannot be disturbed by fast input changes.

- The **Wilkinson ADC** was designed by D. H. Wilkinson in 1950. The Wilkinson ADC is based on the comparison of an input voltage with that produced by a charging capacitor. The capacitor is allowed to charge until its voltage is equal to the amplitude of the input pulse. (A comparator determines when this condition has been reached.) Then, the capacitor is allowed to discharge linearly, which produces a ramp voltage. At the point when the capacitor begins to discharge, a gate pulse is initiated. The gate pulse remains on until the capacitor is completely discharged. Thus the duration of the gate pulse is directly proportional to the amplitude of the input pulse. This gate pulse operates a linear gate which receives pulses from a high-frequency oscillator clock. While the gate is open, a discrete number of clock pulses pass through the linear gate and are counted by the address register. The time the linear gate is open is proportional to the amplitude of the input pulse, thus the number of clock pulses recorded in the address register is proportional also. Alternatively, the charging of the capacitor could be monitored, rather than the discharge.

- An **integrating ADC** (also **dual-slope** or **multi-slope** ADC) applies the unknown input voltage to the input of an integrator and allows the voltage to ramp for a fixed time period (the run-up period). Then a known reference voltage of opposite polarity is applied to the integrator and is allowed to ramp until the integrator output returns to zero (the run-down period). The input voltage is computed as a function of the reference voltage, the constant run-up time period, and the measured run-down time period. The run-down time measurement is usually made in units of the converter's clock, so longer integration times allow for higher resolutions. Likewise, the speed of the converter can be improved by sacrificing resolution. Converters of this type (or variations on the concept) are used in most digital voltmeters for their linearity and flexibility.

- A **delta-encoded ADC** or Counter-ramp has an up-down counter that feeds a digital to analog converter (DAC). The input signal and the DAC both go to a comparator. The comparator controls the counter. The circuit uses negative feedback from the comparator to adjust the counter until the DAC's output is close enough to the input signal. The number is read from the counter. Delta converters have very wide ranges, and high resolution, but the conversion time is dependent on the input signal level, though it will always have a guaranteed worst-case. Delta converters are often very good choices to read real-world signals. Most signals from physical systems do not change abruptly. Some converters combine the delta and successive approximation approaches; this works especially well when high frequencies are known to be small in magnitude.

- A **pipeline ADC** (also called **subranging quantizer**) uses two or more steps of subranging. First, a coarse conversion is done. In a second step, the difference to the input signal is determined with a digital to analog converter (DAC). This difference is then converted finer, and the results are combined in a last step. This can be considered a refinement of the successive approximation ADC wherein the feedback reference signal consists of the interim conversion of a whole range of bits (for example, four bits) rather than just the next-most-significant bit. By combining the merits of the successive approximation and flash ADCs this type is fast, has a high resolution, and only requires a small die size.

- A **Sigma-Delta ADC** (also known as a Delta-Sigma ADC) oversamples the desired signal by a large factor and filters the desired signal band. Generally, a smaller number of bits than required are converted using a Flash ADC after the filter. The resulting signal, along with the error generated by the discrete levels of the Flash, is fed back and subtracted from the input to the filter. This negative feedback has the effect of noise shaping the error due to the Flash so that it does not appear in the desired signal frequencies. A digital filter (decimation filter) follows the ADC which reduces the sampling rate, filters off unwanted noise signal and increases the resolution of the output (sigma-delta modulation, also called delta-sigma modulation).

- A **Time-interleaved ADC** uses M parallel ADCs where each ADC sample data every M:th cycle of the effective sample clock. The result is that the sample rate is increased M times compared to what each individual ADC can manage. In practice, the individual differences between the M ADCs degrade the overall performance reducing the SFDR. However, technologies exist to correct for these time-interleaving mismatch errors.

- An **ADC with intermediate FM stage** first uses a voltage-to-frequency converter to converts the desired signal into an oscillating signal with a frequency proportional to the voltage of the desired signal, and then uses a frequency counter to convert that frequency into a digital count proportional to the desired signal voltage. Longer integration times allow for higher resolutions. Likewise, the speed of the converter can be improved by sacrificing resolution. The two parts of the ADC may be widely separated, with the frequency signal passed through a opto-isolator or transmitted wirelessly. Some such ADCs use sine wave or square wave frequency modulation; others use pulse-frequency modulation. Such ADCs were once the most popular way to show a digital display of the status of a remote analog sensor.

There can be other ADCs that use a combination of electronics and other technologies:

- A **Time-stretch analog-to-digital converter (TS-ADC)** digitizes a very wide bandwidth analog signal, that cannot be digitized by a conventional electronic ADC, by time-stretching the signal prior to digitization. It commonly uses a photonic preprocessor frontend to time-stretch the signal, which effectively slows the signal down in time and compresses its bandwidth. As a result, an electronic backend ADC, that would have been too slow to capture the original signal, can now capture this slowed down signal. For continuous capture of the signal, the frontend also divides the signal into multiple segments in addition to time-stretching. Each segment is individually digitized by a separate electronic ADC. Finally, a digital signal processor rearranges the samples and removes any distortions added by the frontend to yield the binary data that is the digital representation of the original analog signal.

Commercial analog-to-digital converters

These are usually integrated circuits.

Most converters sample with 6 to 24 bits of resolution, and produce fewer than 1 megasample per second. Thermal noise generated by passive components such as resistors masks the measurement when higher resolution is desired. For audio applications and in room temperatures, such noise is usually a little less than 1 μV (microvolt) of white noise. If the Most Significant Bit corresponds to a standard 2 volts of output signal, this translates to a noise-limited performance that is less than 20~21 bits, and obviates the need for any dithering. Mega- and gigasample per second converters are available, though (Feb 2002). Megasample converters are required in digital video cameras, video capture cards, and TV tuner cards to convert full-speed analog video to digital video files. Commercial converters usually have ±0.5 to ±1.5 LSB error in their output.

In many cases the most expensive part of an integrated circuit is the pins, because they make the package larger, and each pin has to be connected to the integrated circuit's silicon. To save pins, it is common for slow ADCs to send their data one bit at a time over a serial interface to the computer, with the next bit coming out when a clock signal changes state, say from zero to 5 V. This saves quite a few pins on the ADC package, and in many cases, does not make the overall design any more complex (even microprocessors which use memory-mapped I/O only need a few bits of a port to implement a serial bus to an ADC).

Commercial ADCs often have several inputs that feed the same converter, usually through an analog multiplexer. Different models of ADC may include sample and hold circuits, instrumentation amplifiers or differential inputs, where the quantity measured is the difference between two voltages.

Applications

Application to music recording

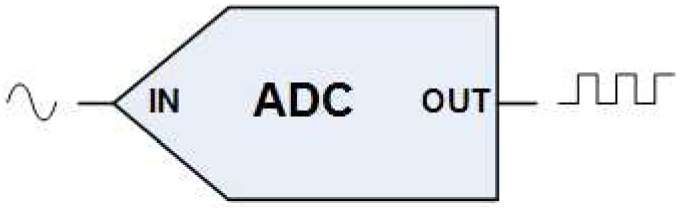
ADCs are integral to current music reproduction technology. Since much music production is done on computers, when an analog recording is used, an ADC is needed to create the PCM data stream that goes onto a compact disc or digital music file.

The current crop of AD converters utilized in music can sample at rates up to 192 kilohertz. High bandwidth headroom allows the use of cheaper or faster anti-aliasing filters of less severe filtering slopes. The proponents of oversampling assert that such shallower anti-aliasing filters produce less deleterious effects on sound quality, exactly because of their gentler slopes. Others prefer entirely filterless AD conversion, arguing that aliasing is less detrimental to sound perception than pre-conversion brickwall filtering. Considerable literature exists on these matters, but commercial considerations often play a significant role. Most high-profile recording studios record in 24-bit/192-176.4 kHz PCM or in DSD formats, and then downsample or decimate the signal for Red-Book CD production (44.1 kHz or at 48 kHz for commonly used for radio/TV broadcast applications).

Digital Signal Processing

AD converters are used virtually everywhere where an analog signal has to be processed, stored, or transported in digital form. Fast video ADCs are used, for example, in TV tuner cards. Slow on-chip 8, 10, 12, or 16 bit ADCs are common in microcontrollers. Very fast ADCs are needed in digital oscilloscopes, and are crucial for new applications like software defined radio.

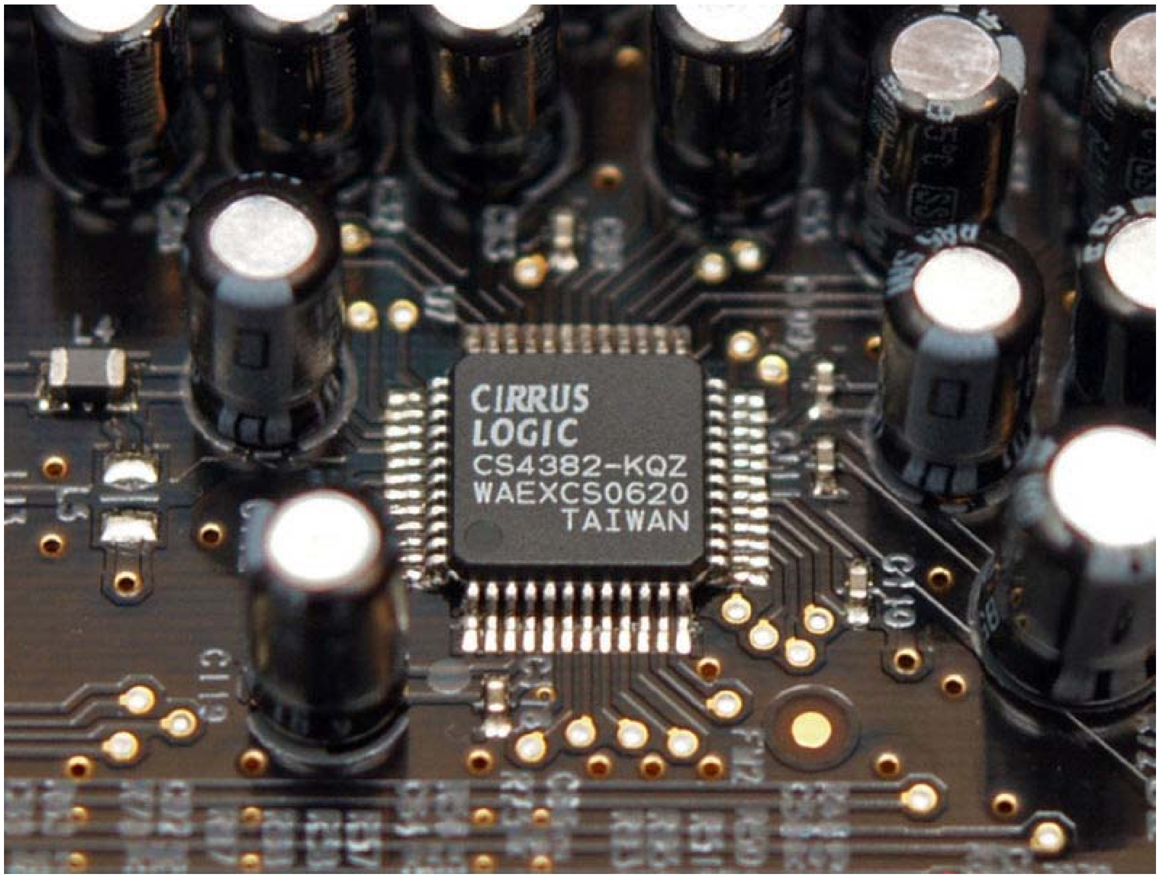
Electrical Symbol



ELECTRICAL SYMBOL FOR ANALOG TO DIGITAL CONVERTER (ADC)

Chapter 8

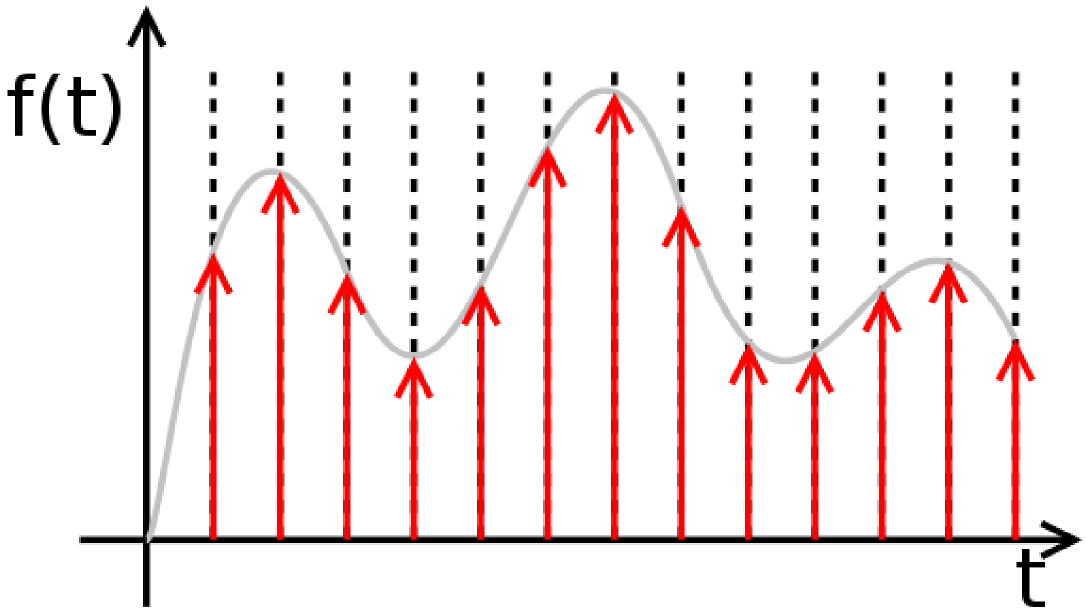
Digital-to-Analog Converter



8-channel digital-to-analog converter Cirrus Logic CS4382 as used in a soundcard.

In electronics, a **digital-to-analog converter** (**DAC** or **D-to-A**) is a device that converts a digital (usually binary) code to an analog signal (current, voltage, or electric charge). An analog-to-digital converter (ADC) performs the reverse operation.

Basic ideal operation



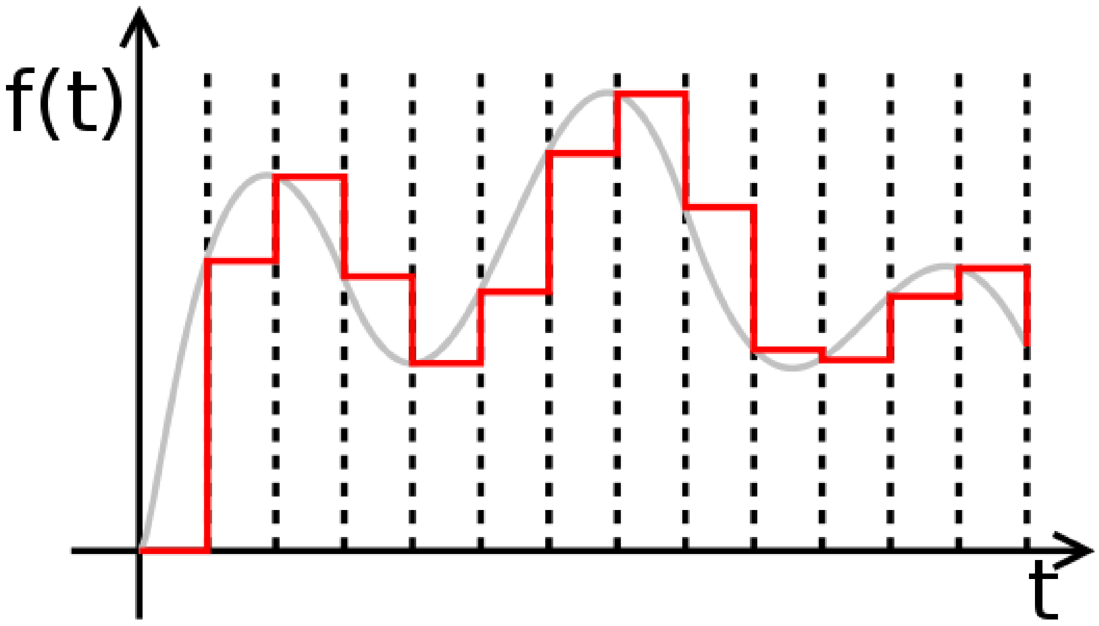
Ideally sampled signal.

A DAC converts an abstract finite-precision number (usually a fixed-point binary number) into a concrete physical quantity (e.g., a voltage or a pressure). In particular, DACs are often used to convert finite-precision time series data to a continually varying physical signal.

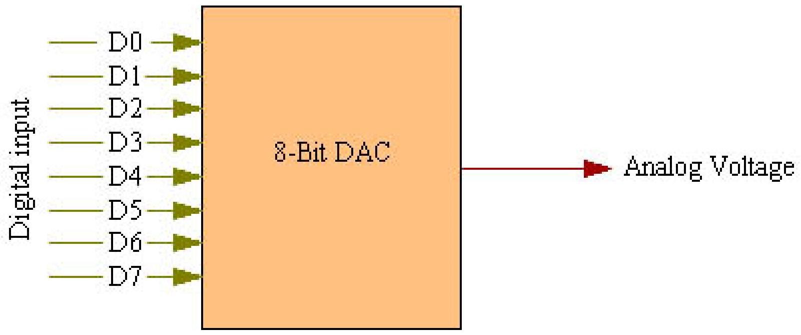
A typical DAC converts the abstract numbers into a concrete sequence of impulses that are then processed by a reconstruction filter using some form of interpolation to fill in data between the impulses. Other DAC methods (e.g., methods based on Delta-sigma modulation) produce a pulse-density modulated signal that can then be filtered in a similar way to produce a smoothly varying signal.

By the Nyquist–Shannon sampling theorem, sampled data can be reconstructed perfectly provided that its bandwidth meets certain requirements (e.g., a baseband signal with bandwidth less than the Nyquist frequency; BUT requires an infinite number of samples. The finite number used in real life cause other problems especially with the D/A reconstruction of the original signal. However, even with an ideal reconstruction filter, digital sampling introduces quantization error that makes perfect reconstruction practically impossible. Increasing the digital resolution (i.e., increasing the number of bits used in each sample) or introducing sampling dither can reduce this error.

Practical operation



Piecewise constant output of a conventional practical DAC.



A simplified functional diagram of an 8-bit DAC

Instead of impulses, usually the sequence of numbers update the analogue voltage at uniform sampling intervals.

These numbers are written to the DAC, typically with a clock signal that causes each number to be latched in sequence, at which time the DAC output voltage changes rapidly from the previous value to the value represented by the currently latched number. The effect of this is that the output voltage is *held* in time at the current value until the next input number is latched resulting in a piecewise constant or 'staircase' shaped output. This is equivalent to a zero-order hold operation and has an effect on the frequency response of the reconstructed signal.

The fact that DACs output a sequence of piecewise constant values (known as zero-order hold in sample data textbooks) or rectangular pulses causes multiple harmonics above the Nyquist frequency. Usually, these are removed with a low pass filter acting as a reconstruction filter in applications that require it.

Applications

Audio

Top-loading CD player and external digital-to-analog converter.

Most modern audio signals are stored in digital form (for example MP3s and CDs) and in order to be heard through speakers they must be converted into an analog signal. DACs are therefore found in CD players, digital music players, and PC sound cards.

Specialist standalone DACs can also be found in high-end hi-fi systems. These normally take the digital output of a compatible CD player or dedicated transport and convert the signal into an analog line-level output that can then be fed into an amplifier to drive speakers.

Similar digital-to-analog converters can be found in digital speakers such as USB speakers, and in sound cards.

VOIP (Voice over IP) Phone, Data transmission over the Internet is done digitally so in order for voice to be transmitted it must be converted to digital using an Analog-to-Digital Converter and be converted into analog again using a DAC so the voice it can be heard on the other end.

Video

Video signals from a digital source, such as a computer, must be converted to analog form if they are to be displayed on an analog monitor. As of 2007, analog inputs are more commonly used than digital, but this may change as flat panel displays with DVI and/or HDMI connections become more widespread. A video DAC is, however, incorporated in any digital video player with analog outputs. The DAC is usually integrated with some memory (RAM), which contains conversion tables for gamma correction, contrast and brightness, to make a device called a RAMDAC.

A device that is distantly related to the DAC is the digitally controlled potentiometer, used to control an analog signal digitally.

Mechanical

An unusual application of digital-to-analog conversion was the whiffletree electromechanical digital-to-analog convertor linkage in the IBM Selectric typewriter.

DAC types

The most common types of electronic DACs are:

- The pulse-width modulator, the simplest DAC type. A stable current or voltage is switched into a low-pass analog filter with a duration determined by the digital input code. This technique is often used for electric motor speed control, but has many other applications as well.
- Oversampling DACs or interpolating DACs such as the delta-sigma DAC, use a pulse density conversion technique. The oversampling technique allows for the use of a lower resolution DAC internally. A simple 1-bit DAC is often chosen because the oversampled result is inherently linear. The DAC is driven with a pulse-density modulated signal, created with the use of a low-pass filter, step nonlinearity (the actual 1-bit DAC), and negative feedback loop, in a technique called delta-sigma modulation. This results in an effective high-pass filter acting on the quantization (signal processing) noise, thus steering this noise out of the low frequencies of interest into the megahertz frequencies of little interest, which is called noise shaping. The quantization noise at these high frequencies is removed or greatly attenuated by use of an analog low-pass filter at the output (sometimes a simple RC low-pass circuit is sufficient). Most very high resolution

DACs (greater than 16 bits) are of this type due to its high linearity and low cost. Higher oversampling rates can relax the specifications of the output low-pass filter and enable further suppression of quantization noise. Speeds of greater than 100 thousand samples per second (for example, 192 kHz) and resolutions of 24 bits are attainable with delta-sigma DACs. A short comparison with pulse-width modulation shows that a 1-bit DAC with a simple first-order integrator would have to run at 3 THz (which is physically unrealizable) to achieve 24 meaningful bits of resolution, requiring a higher-order low-pass filter in the noise-shaping loop. A single integrator is a low-pass filter with a frequency response inversely proportional to frequency and using one such integrator in the noise-shaping loop is a first order delta-sigma modulator. Multiple higher order topologies (such as MASH) are used to achieve higher degrees of noise-shaping with a stable topology.

- The binary-weighted DAC, which contains one resistor or current source for each bit of the DAC connected to a summing point. These precise voltages or currents sum to the correct output value. This is one of the fastest conversion methods but suffers from poor accuracy because of the high precision required for each individual voltage or current. Such high-precision resistors and current sources are expensive, so this type of converter is usually limited to 8-bit resolution or less.
- The R-2R ladder DAC which is a binary-weighted DAC that uses a repeating cascaded structure of resistor values R and 2R. This improves the precision due to the relative ease of producing equal valued-matched resistors (or current sources). However, wide converters perform slowly due to increasingly large RC-constants for each added R-2R link.
- The thermometer-coded DAC, which contains an equal resistor or current-source segment for each possible value of DAC output. An 8-bit thermometer DAC would have 255 segments, and a 16-bit thermometer DAC would have 65,535 segments. This is perhaps the fastest and highest precision DAC architecture but at the expense of high cost. Conversion speeds of >1 billion samples per second have been reached with this type of DAC.
- Hybrid DACs, which use a combination of the above techniques in a single converter. Most DAC integrated circuits are of this type due to the difficulty of getting low cost, high speed and high precision in one device.
    - The segmented DAC, which combines the thermometer-coded principle for the most significant bits and the binary-weighted principle for the least significant bits. In this way, a compromise is obtained between precision (by the use of the thermometer-coded principle) and number of resistors or current sources (by the use of the binary-weighted principle). The full binary-weighted design means 0% segmentation, the full thermometer-coded design means 100% segmentation.

DAC performance

DACs are very important to system performance. The most important characteristics of these devices are:

- **Resolution**: This is the number of possible output levels the DAC is designed to reproduce. This is usually stated as the number of bits it uses, which is the base two logarithm of the number of levels. For instance a 1 bit DAC is designed to reproduce 2 (2^1) levels while an 8 bit DAC is designed for 256 (2^8) levels. Resolution is related to the **effective number of bits** (ENOB) which is a measurement of the actual resolution attained by the DAC.
- **Maximum sampling frequency**: This is a measurement of the maximum speed at which the DACs circuitry can operate and still produce the correct output. As stated in the Nyquist–Shannon sampling theorem, a signal must be sampled at over twice the frequency of the desired signal. For instance, to reproduce signals in all the audible spectrum, which includes frequencies of up to 20 kHz, it is necessary to use DACs that operate at over 40 kHz. The CD standard samples audio at 44.1 kHz, thus DACs of this frequency are often used. A common frequency in cheap computer sound cards is 48 kHz — many work at only this frequency, offering the use of other sample rates only through (often poor) internal resampling.
- **Monotonicity**: This refers to the ability of a DAC's analog output to move only in the direction that the digital input moves (i.e., if the input increases, the output doesn't dip before asserting the correct output.) This characteristic is very important for DACs used as a low frequency signal source or as a digitally programmable trim element.
- **THD+N**: This is a measurement of the distortion and noise introduced to the signal by the DAC. It is expressed as a percentage of the total power of unwanted harmonic distortion and noise that accompany the desired signal. This is a very important DAC characteristic for dynamic and small signal DAC applications.
- **Dynamic range**: This is a measurement of the difference between the largest and smallest signals the DAC can reproduce expressed in decibels. This is usually related to DAC resolution and noise floor.

Other measurements, such as phase distortion and jitter, can also be very important for some applications.

Bits	**Color limit**	**Frequency**	**Examples**
10	1.024 colors	54 MHz	
12		54 MHz	Sony NS-575p
12	4.096 colors	108 MHz	
12		150 MHz	NeoDigits Helios X5000
12		216 MHz	Philips BDP9000 (Blu-ray)
12		297 MHz	Toshiba HD-XE1
12		216 MHz	Samsung BD-P1200 (Blu-ray)
14	16.384 colors	108 MHz	Pioneer Elite, Black Finish, DV79AVI
14		216 MHz	Marantz DV9600, Sony DVPNS9100ES
16		149 MHz	NeuNeo HVD108

DAC figures of merit

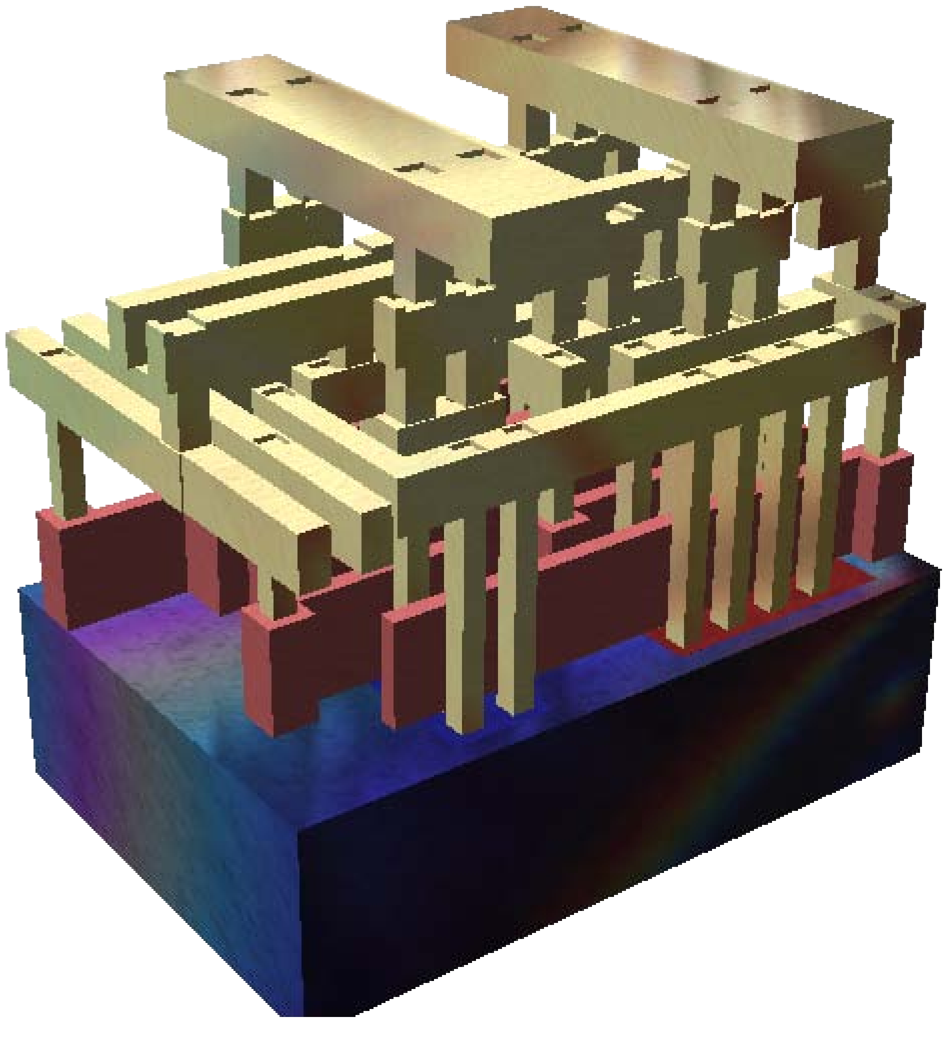
- Static performance:
    - Differential nonlinearity (DNL) shows how much two adjacent code analog values deviate from the ideal 1LSB step
    - Integral nonlinearity (INL) shows how much the DAC transfer characteristic deviates from an ideal one. That is, the ideal characteristic is usually a straight line; INL shows how much the actual voltage at a given code value differs from that line, in LSBs (1LSB steps).
    - Gain
    - Offset
    - Noise is ultimately limited by the thermal noise generated by passive components such as resistors. For audio applications and in room temperatures, such noise is usually a little less than 1 μV (microvolt) of white noise. This limits performance to less than 20~21 bits even in 24-bit DACs.
- Frequency domain performance
    - Spurious-free dynamic range (SFDR) indicates in dB the ratio between the powers of the converted main signal and the greatest undesired spur
    - Signal to noise and distortion ratio (SNDR) indicates in dB the ratio between the powers of the converted main signal and the sum of the noise and the generated harmonic spurs
    - i-th harmonic distortion (HDi) indicates the power of the i-th harmonic of the converted main signal
    - Total harmonic distortion (THD) is the sum of the powers of all HDi
    - If the maximum DNL error is less than 1 LSB, then D/A converter is guaranteed to be monotonic.

However, many monotonic converters may have a maximum DNL greater than 1 LSB.

- Time domain performance:
    - Glitch energy
    - Response uncertainty
    - Time nonlinearity (TNL)

Chapter 9

Standard Cell



A rendering of a small standard cell with three metal layers (dielectric has been removed). The sand-colored structures are metal interconnect, with the vertical pillars being contacts, typically plugs of tungsten. The reddish structures are polysilicon gates, and the solid at the bottom is the crystalline silicon bulk.

In semiconductor design, **standard cell** methodology is a method of designing application-specific integrated circuits (ASICs) with mostly digital-logic features. Standard cell methodology is an example of design abstraction, whereby a low-level very-large-scale integration (VLSI) layout is encapsulated into an abstract logic representation (such as a NAND gate). Cell-based methodology (the general class to which standard cells belong) makes it possible for one designer to focus on the high-level (logical function) aspect of digital design, while another designer focuses on the implementation (physical) aspect. Along with semiconductor manufacturing advances, standard cell methodology has helped designers scale ASICs from comparatively simple single-function ICs (of several thousand gates), to complex multi-million gate system-on-a-chip (SoC) devices.

Construction of a standard cell

A standard cell is a group of transistor and interconnect structures that provides a boolean logic function (e.g., AND, OR, XOR, XNOR, inverters) or a storage function (flipflop or latch). The simplest cells are direct representations of the elemental NAND, NOR, and XOR boolean function, although cells of much greater complexity are commonly used (such as a 2-bit full-adder, or muxed D-input flipflop.) The cell's boolean logic function is called its *logical view*: functional behavior is captured in the form of a truth table or Boolean algebra equation (for combinational logic), or a state transition table (for sequential logic).

Usually, the initial design of a standard cell is developed at the transistor level, in the form of a *transistor netlist* or *schematic* view. The netlist is a nodal description of transistors, of their connections to each other, and of their terminals (ports) to the external environment. A schematic view may be generated with a number of different Computer Aided Design (CAD) or Electronic Design Automation(EDA) programs that provide a Graphical User Interface (GUI) for this netlist generation process. Designers use additional CAD programs such as SPICE or Spectre to simulate the electronic behavior of the netlist, by declaring input stimulus (voltage or current waveforms) and then calculating the circuit's time domain (analogue) response. The simulations verify whether the netlist implements the desired function and predict other pertinent parameters, such as power consumption or signal propagation delay.

Since the logical and netlist views are only useful for abstract (algebraic) simulation, and not device fabrication, the physical representation of the standard cell must be designed too. Also called the *layout view*, this is the lowest level of design abstraction in common design practice. From a manufacturing perspective, the standard cell's VLSI layout is the most important view, as it is closest to an actual "manufacturing blueprint" of the standard cell. The layout is organized into *base layers*, which correspond to the different structures of the transistor devices, and *interconnect wiring layess* and *via layers*, which join together the terminals of the transistor formations. The *interconnect wiring layers* are usually numbered and have specific *via' layers representing specific connections between each sequential layer. Non-manufacturing layers may be also be present in a layout for purposes of Design Automation, but many layers used explicitly for Place and route (PNR) CAD programs are often included in a separate but similar* abstract *view. The abstract view often contains much less information than the layout and may be recognizable as a Layout Extraction Format (LEF) file or an equivalent.*

After a layout is created, additional CAD tools are often used to perform a number of common validations. A Design Rule Check (DRC) is done to verify that the design meets foundry and other layout requirements. A Parasitic EXtractction (PEX)then is performed to generate a PEX-netlist with parasitic properties from the layout. The nodal connections of that netlist are then compared to those of the schematic netlist with a *Layout Vs Schematic* (LVS) procedure to verify that the connectivity models are equivalent.

The PEX-netlist may then be simulated again (since it contains parasitic properties) to achieve more accurate timing, power, and noise models. These models are often *characterized* (contained) in a Synopsys Liberty format, but other Verilog formats may be used as well.

Finally, powerful Place and Route (PNR) tools may be used to pull everything together and *synthesize* (generate) Very Large Scale Integration (VLSI) layouts, in an automated fashion, from higher level design netlists and floor-plans.

Additionally, a number of other CAD tools may be used to validate other aspects of the cell views and models. And other files may be created to support various tools that utilize the standard cells for a plethora of other reasons. All of these files that are created to support the use of all of the standard cell variations are collectively known as a standard cell library.

For a typical Boolean function, there are many different functionally equivalent transistor netlists. Likewise, for a typical netlist, there are many different layouts that fit the netlist's performance parameters. The designer's challenge is to minimize the manufacturing cost of the standard cell's layout (generally by minimizing the circuit's die area), while still meeting the cell's speed and power performance requirements. Consequently, integrated circuit layout is a highly labor intensive job, despite the existence of design tools to aid this process.

Library

A standard cell library is a collection of low-level logic functions such as AND, OR, INVERT, flip-flops, latches, and buffers. These cells are realized as fixed-height, variable-width full-custom cells. The key aspect with these libraries is that they are of a fixed height, which enables them to be placed in rows, easing the process of automated digital layout. The cells are typically optimized full-custom layouts, which minimize delays and area.

A typical standard-cell library contains two main components:

1. Library Database - Consists of a number of views often including layout, schematic, symbol, abstract, and other logical or simulation views. From this, various information may be captured in a number of formats including the Cadence LEF format, and the Synopsys Milkyway format, which contain reduced information about the cell layouts, sufficient for automated "Place and Route" tools.
2. Timing Abstract - This is generally in the Synopsys Liberty format, and provides functional definitions, timing, power, and noise information for each cell.

A standard-cell library may also contain the following additional components:

- A full layout of the cells
- Spice models of the cells
- Verilog models or VHDL Vital models
- Parasitic Extraction models
- DRC rule decks

An example is a simple XOR logic gate, which can be formed from OR, INVERT and AND gates.

Application of standard cell

Strictly speaking, a 2-input NAND or NOR function is sufficient to form any arbitrary Boolean function set. But in modern ASIC design, standard-cell methodology is practiced with a sizable library (or libraries) of cells. The library usually contains multiple implementations of the same logic function, differing in area and speed. This variety enhances the efficiency of automated synthesis, place, and route (SPR) tools. Indirectly, it also gives the designer greater freedom to perform implementation trade-offs (area vs. speed vs. power consumption). A complete group of standard-cell descriptions is commonly called a *technology library*.

Commercially available Electronic Design Automation (EDA) tools use the technology libraries to automate synthesis, placement, and routing of a digital ASIC. The technology library is developed and distributed by the foundry operator. The library (along with a design netlist format) is the basis for exchanging design information between different phases of the SPR process.

Synthesis

Using the technology library's cell logical view, the *Logic Synthesis tool* performs the process of mathematically transforming the ASIC's register-transfer level (RTL) description into a technology-dependent netlist. This process is analogous to a software compiler converting a high-level C-program listing into a processor-dependent assembly-language listing.

The netlist is the standard-cell representation of the ASIC design, at the logical view level. It consists of instances of the standard-cell library gates, and port connectivity between gates. Proper synthesis techniques ensure mathematical equivalency between the synthesized netlist and original RTL description. The netlist contains no unmapped RTL statements and declarations.

The high-level synthesis tool performs the process of transforming the C-level models (SystemC, ANSI C/C++) description into a technology-dependent netlist.

Placement

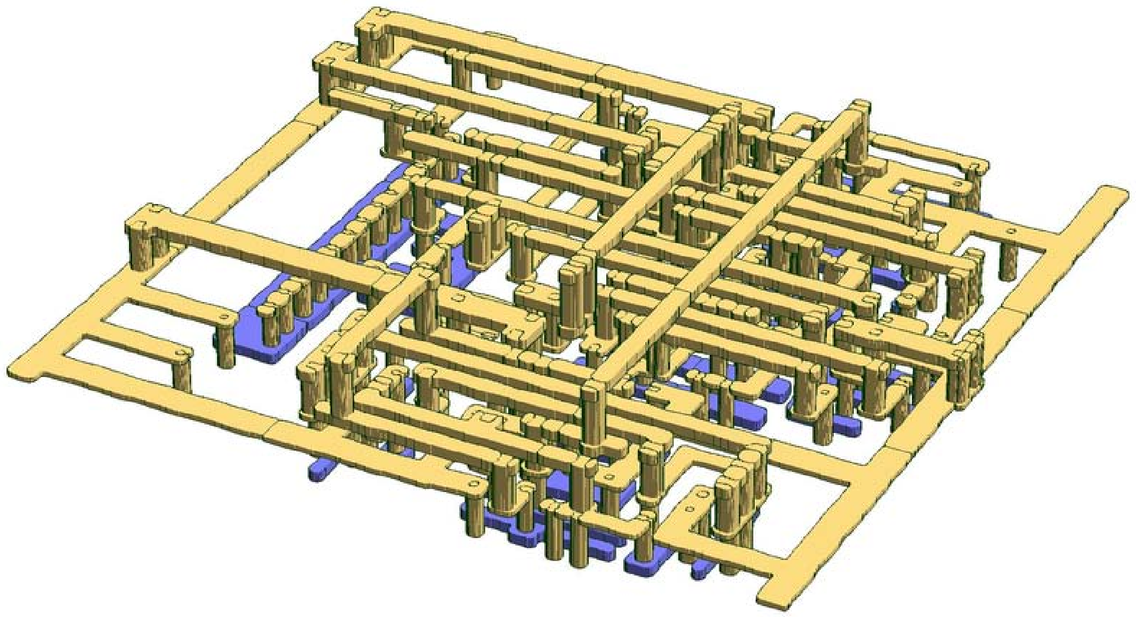
The placement tool starts the physical implementation of the ASIC. With a 2-D floorplan provided by the ASIC designer, the placer tool assigns locations for each gate in the netlist. The resulting *placed gates* netlist contains the physical location of each of the netlist's standard-cells, but retains an abstract description of how the gates' terminals are wired to each other.

Typically the standard cells have a constant size in at least one dimension that allows them to be lined up in rows on the integrated circuit. The chip will consist of a huge number of rows (with power and ground running next to each row) with each row filled with the various cells making up the actual design. Placers obey certain rules: Each gate is assigned a unique (exclusive) location on the die map. A given gate is placed once, and may not occupy or overlap the location of any other gate.

Routing

Using the placed-gates netlist and the layout view of the library, the router adds both signal connect lines and power supply lines. The fully routed physical netlist contains the listing of gates from synthesis, the placement of each gate from placement, and the drawn interconnects from routing.

DRC/LVS


Simulated lithographic and other fabrication defects visible in a small standard cell.

Design Rule Check (DRC) and Layout Versus Schematic (LVS) are verification processes. Reliable device fabrication at modern deep-submicrometer (0.13 µm and below) requires strict observance of transistor spacing, metal layer thickness, and power density rules. DRC exhaustively compares the physical netlist against a set of "foundry design rules" (from the foundry operator), then flags any observed violations.

The LVS process confirms that the layout has the same structure as the associated schematic; this is typically the final step in the layout process. The LVS tool takes as an input a schematic diagram and the extracted view from a layout. It then generates a netlist from each one and compares them. Nodes, ports, and device sizing are all compared. If they are the same, LVS passes and the designer can continue. LVS tends to consider transistor fingers to be the same as an extra-wide transistor. Thus, 4 transistors in parallel (each 1 µm wide), a 4-finger 1 µm transistor, and a 4 µm transistor are viewed as the same by the LVS tool. Functionality of .lib files will be taken from SPICE models and added as an attribute to the .lib file.

Other cell-based methodologies

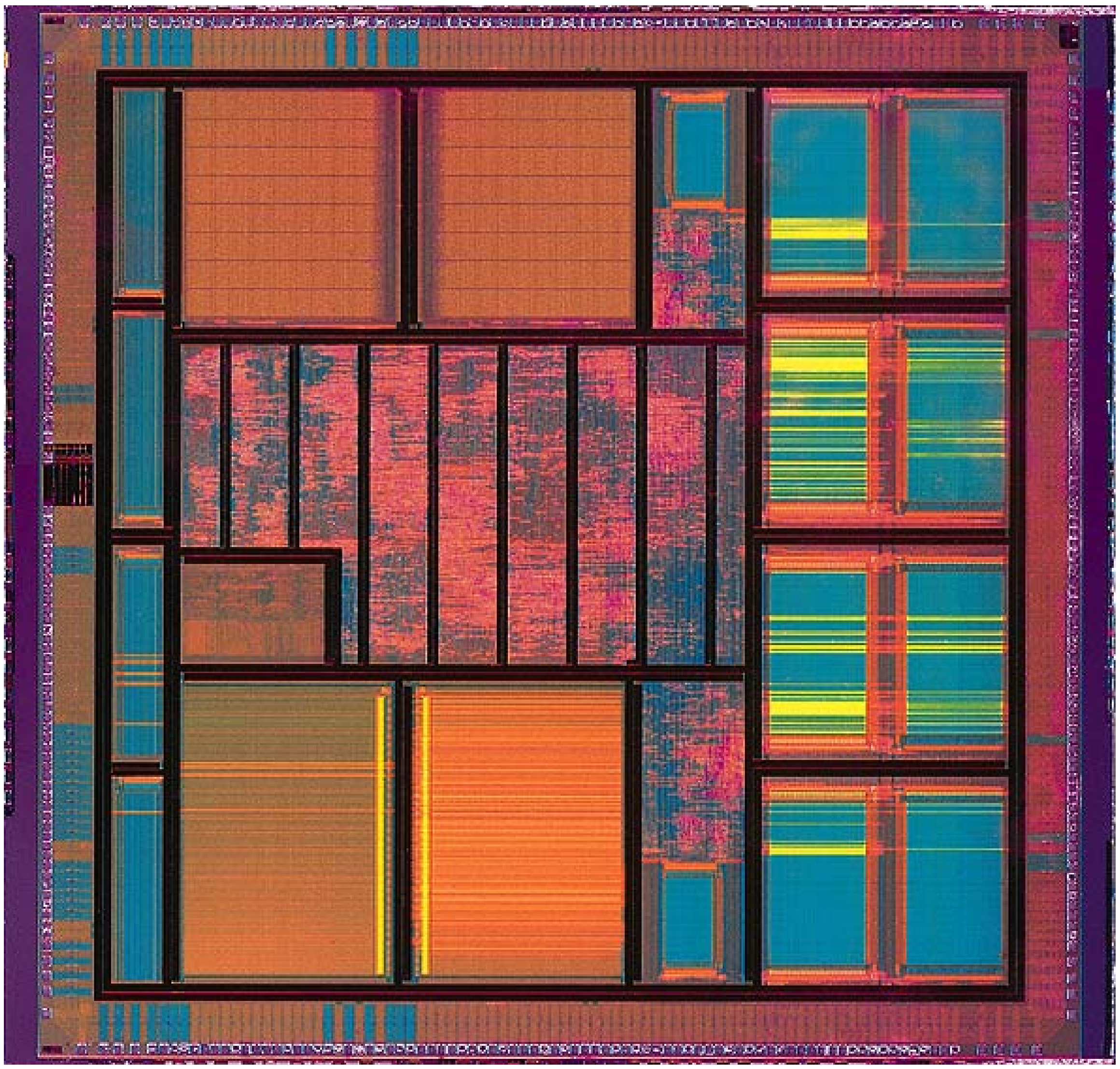
"Standard cell" falls into a more general class of design automation flows called cell-based design. Structured ASICs, FPGAs, and CPLDs are variations on cell-based design. From the designer's standpoint, all share the same input front end: an RTL description of the design. The three techniques, however, differ substantially in the details of the SPR flow and physical implementation.

Complexity measure

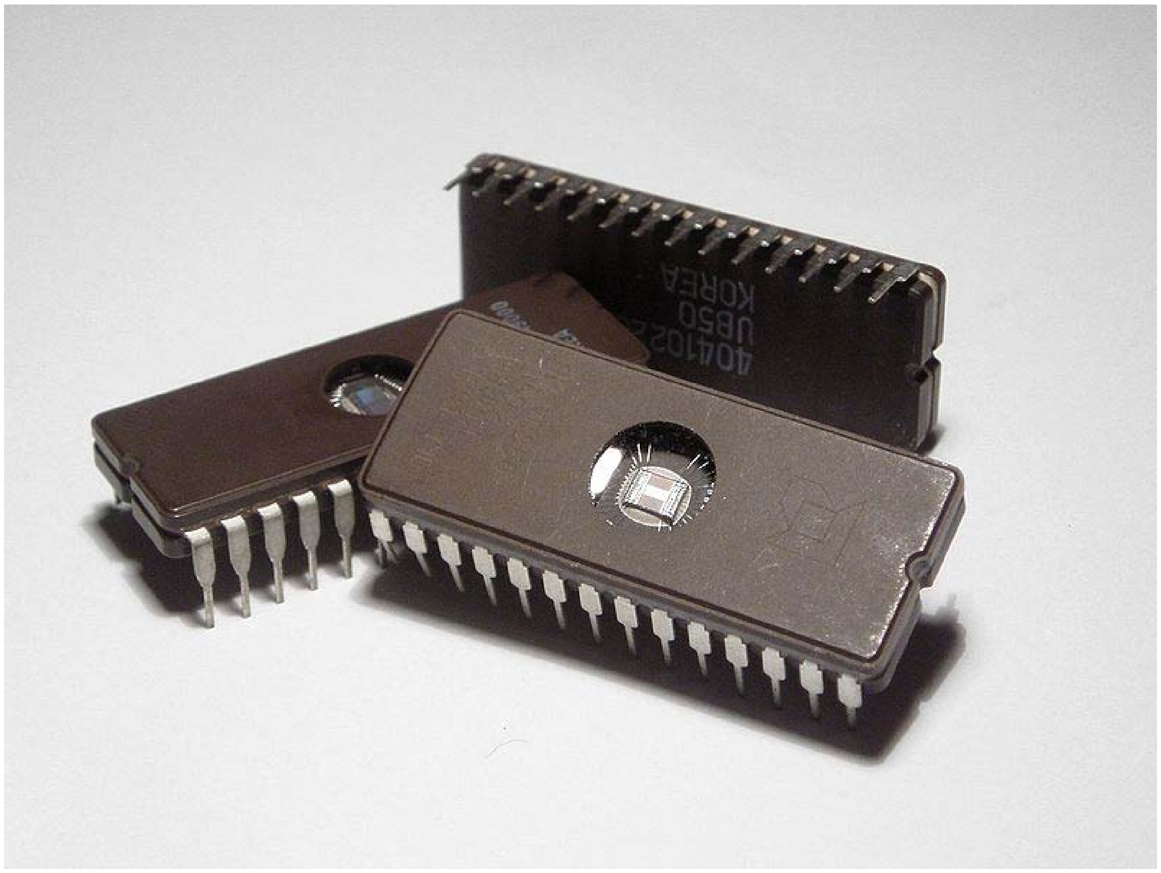
For digital standard cell designs, for instance in CMOS, a common technology-independent metric for complexity measure is gate equivalents (GE).

Chapter 10

Integrated Circuit



Integrated circuit of Atmel Diopsis 740 System on Chip showing memory blocks, logic and input/output pads around the periphery

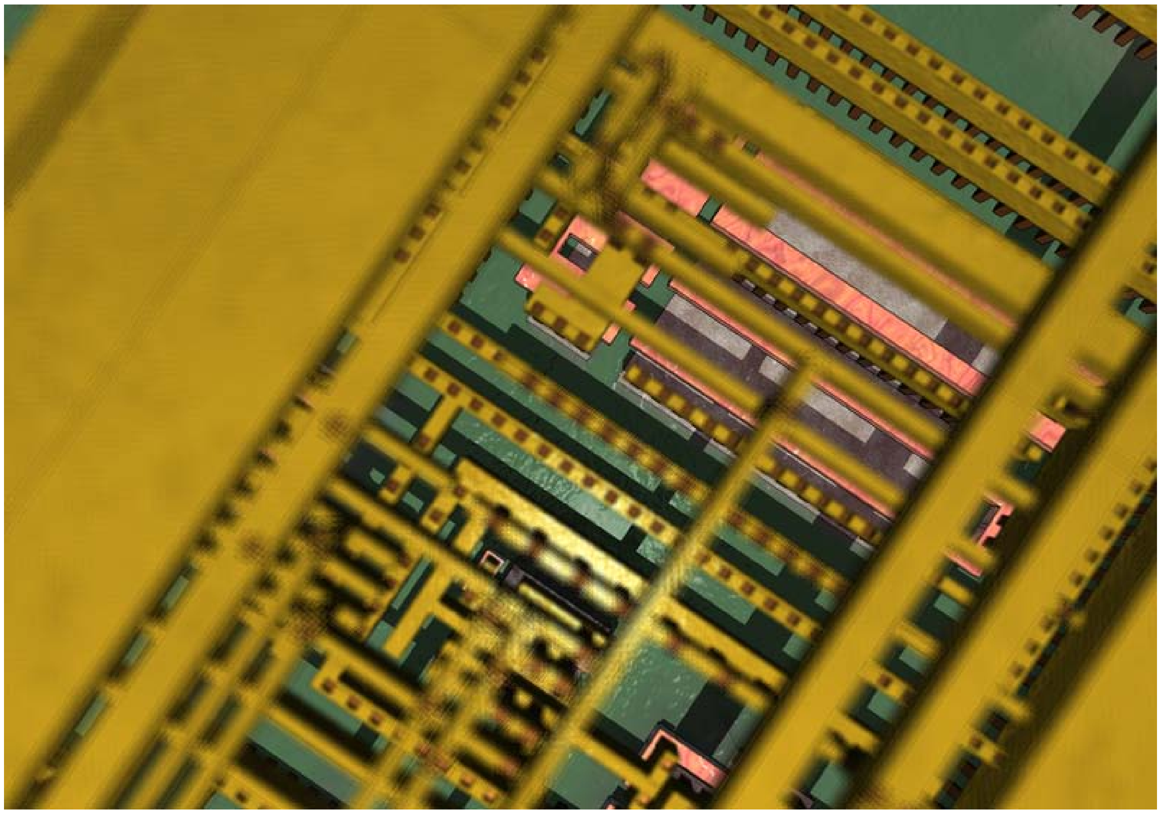


Microchips (EPROM memory) with a transparent window, showing the integrated circuit inside. Note the fine silver-colored wires that connect the integrated circuit to the pins of the package. The window allows the memory contents of the chip to be erased, by exposure to strong ultraviolet light in an eraser device.

An **integrated circuit** or **monolithic integrated circuit** (also referred to as **IC**, **chip**, and **microchip**) is an electronic circuit manufactured by diffusion of trace elements into the surface of a thin substrate of semiconductor material.

Integrated circuits are used in almost all electronic equipment in use today and have revolutionized the world of electronics. Computers, cellular phones, and other digital appliances are now inextricable parts of the structure of modern societies, made possible by the low cost of production of integrated circuits.

Introduction



Synthetic detail of an integrated circuit through four layers of planarized copper interconnect, down to the polysilicon (pink), wells (greyish), and substrate (green).

Integrated circuits were made possible by experimental discoveries which showed that semiconductor devices could perform the functions of vacuum tubes and by mid-20th-century technology advancements in semiconductor device fabrication. The integration of large numbers of tiny transistors into a small chip was an enormous improvement over the manual assembly of circuits using electronic components. The integrated circuit's mass production capability, reliability, and building-block approach to circuit design ensured the rapid adoption of standardized ICs in place of designs using discrete transistors.

There are two main advantages of ICs over discrete circuits: cost and performance. Cost is low because the chips, with all their components, are printed as a unit by photolithography rather than being constructed one transistor at a time. Furthermore, much less material is used to construct a packaged IC die than a discrete circuit. Performance is high since the components switch quickly and consume little power (compared to their discrete counterparts) because the components are small and positioned close together. As of 2006, chip areas range from a few square millimeters to around 350 mm^2, with up to 1 million transistors per mm^2.

Terminology

Integrated circuit originally referred to a miniaturized electronic circuit consisting of semiconductor devices, as well as passive components bonded to a substrate or circuit board. This configuration is now commonly referred to as a hybrid integrated circuit. *Integrated circuit* has since come to refer to the single-piece circuit construction originally known as a *monolithic integrated circuit*.

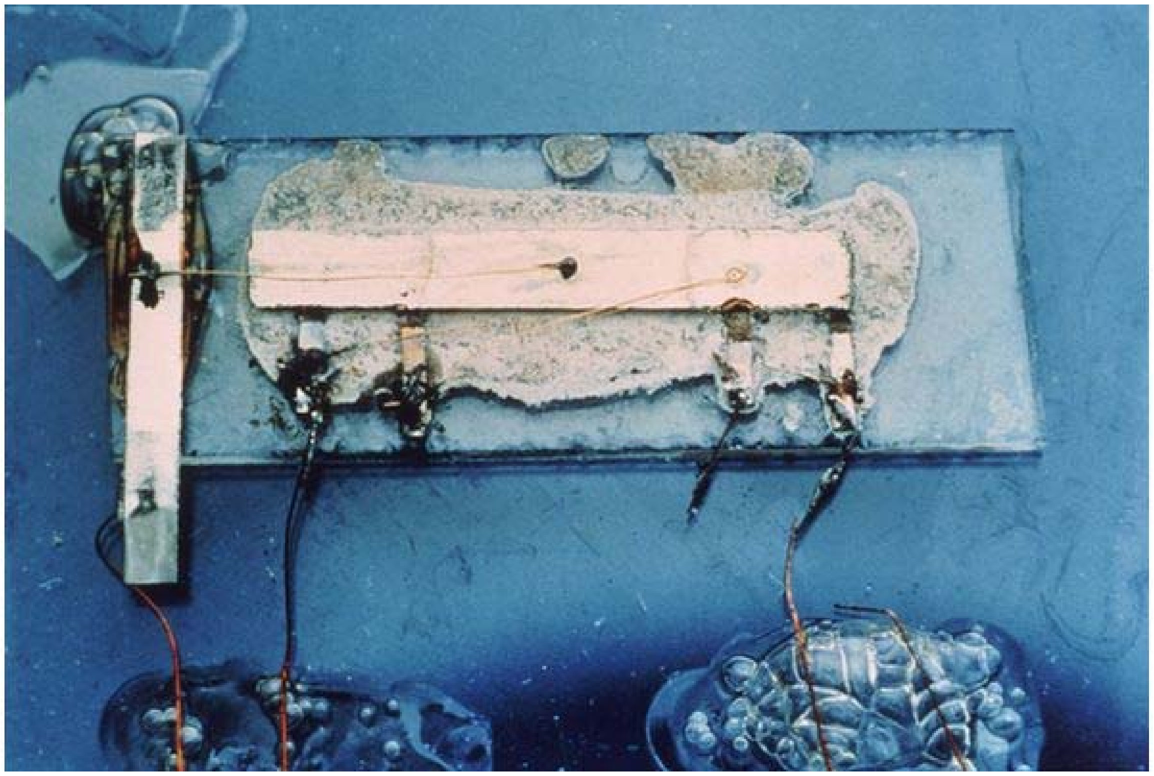
Invention

Early developments of the integrated circuit go back to 1949, when the German engineer Werner Jacobi (Siemens AG) filed a patent for an integrated-circuit-like semiconductor amplifying device showing five transistors on a common substrate arranged in a 2-stage amplifier arrangement. Jacobi discloses small and cheap hearing aids as typical industrial applications of his patent. A commercial use of his patent has not been reported.

The idea of the integrated circuit was conceived by a radar scientist working for the Royal Radar Establishment of the British Ministry of Defence, Geoffrey W.A. Dummer (1909–2002), who published it at the Symposium on Progress in Quality Electronic Components in Washington, D.C. on May 7, 1952. He gave many symposia publicly to propagate his ideas. Dummer unsuccessfully attempted to build such a circuit in 1956.

A precursor idea to the IC was to create small ceramic squares (wafers), each one containing a single miniaturized component. Components could then be integrated and wired into a bidimensional or tridimensional compact grid. This idea, which looked very promising in 1957, was proposed to the US Army by Jack Kilby, and led to the short-lived Micromodule Program (similar to 1951's Project Tinkertoy). However, as the project was gaining momentum, Kilby came up with a new, revolutionary design: the IC.

Robert Noyce credited Kurt Lehovec of Sprague Electric for the *principle of p-n junction isolation* caused by the action of a biased p-n junction (the diode) as a key concept behind the IC.



Jack Kilby's original integrated circuit

Jack Kilby recorded his initial ideas concerning the integrated circuit in July 1958 and successfully demonstrated the first working integrated circuit on September 12, 1958. In his patent application of February 6, 1959, Kilby described his new device as “a body of semiconductor material ... wherein all the components of the electronic circuit are completely integrated.” Kilby won the 2000 Nobel Prize in Physics for his part of the invention of the integrated circuit.

Robert Noyce also came up with his own idea of an integrated circuit half a year later than Kilby. Noyce's chip solved many practical problems that Kilby's had not. Noyce's chip, made at Fairchild Semiconductor, was made of silicon, whereas Kilby's chip was made of germanium.

Generations

In the early days of integrated circuits, only a few transistors could be placed on a chip, as the scale used was large because of the contemporary technology. As the degree of integration was small, the design was done easily. Later on, millions, and today billions, of transistors could be placed on one chip, and to make a good design became a task to be planned thoroughly. This gave rise to new design methods.

SSI, MSI and LSI

The first integrated circuits contained only a few transistors. Called "**Small-Scale Integration**" (**SSI**), digital circuits containing transistors numbering in the tens provided a few logic gates for example, while early linear ICs such as the Plessey SL201 or the Philips TAA320 had as few as two transistors. The term Large Scale Integration was first used by IBM scientist Rolf Landauer when describing the theoretical concept, from there came the terms for SSI, MSI, VLSI, and ULSI.

SSI circuits were crucial to early aerospace projects, and vice-versa. Both the Minuteman missile and Apollo program needed lightweight digital computers for their inertial guidance systems; the Apollo guidance computer led and motivated the integrated-circuit technology, while the Minuteman missile forced it into mass-production. The Minuteman missile program and various other Navy programs accounted for the total $4 million integrated circuit market in 1962, and by 1968, U.S. Government space and defense spending still accounted for 37% of the $312 million total production. The demand by the U.S. Government supported the nascent integrated circuit market until costs fell enough to allow firms to penetrate the industrial and eventually the consumer markets. The average price per integrated circuit dropped from $50.00 in 1962 to $2.33 in 1968. Integrated Circuits began to appear in consumer products by the turn of the decade, a typical application being FM inter-carrier sound processing in television receivers.

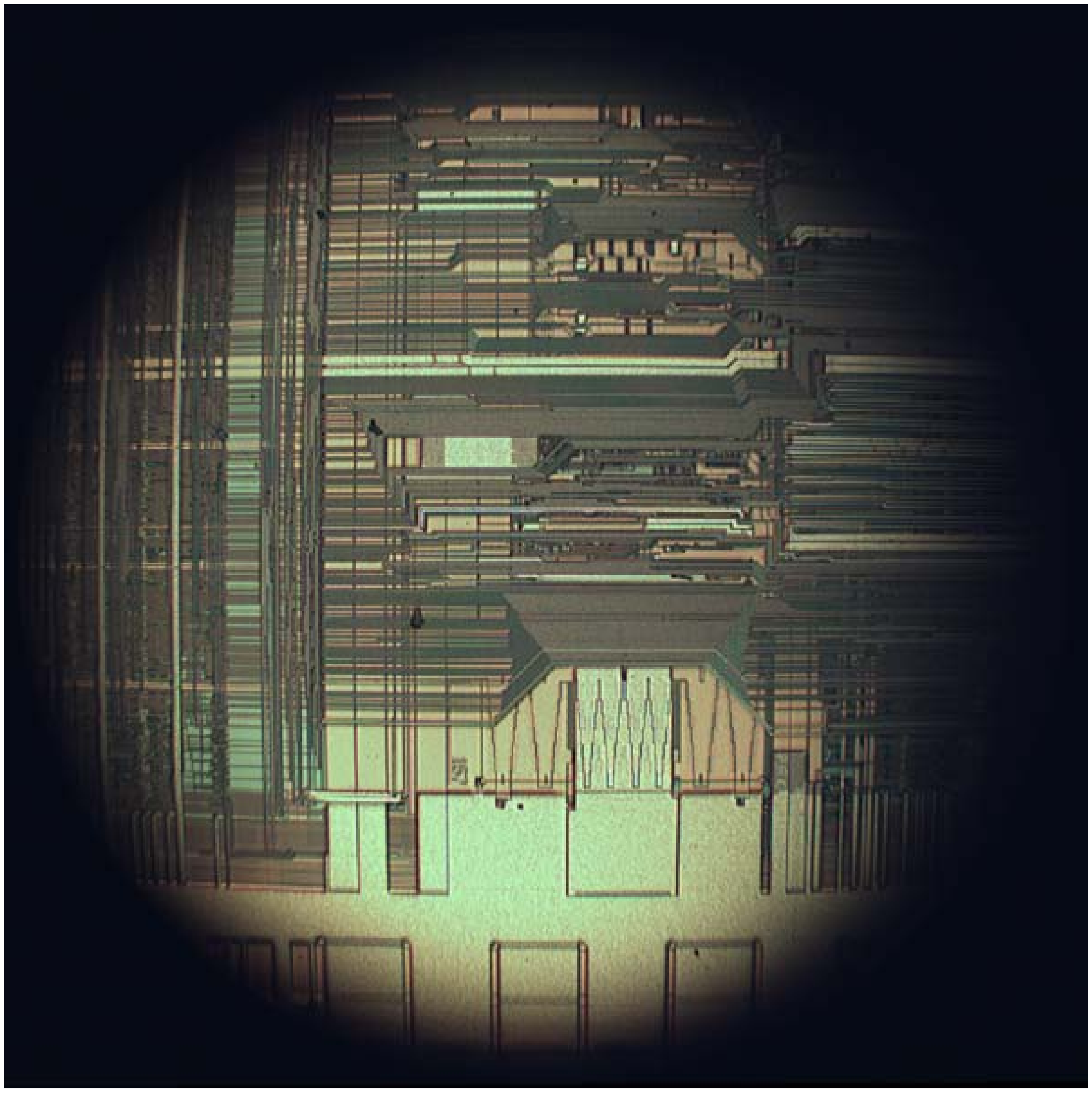
The next step in the development of integrated circuits, taken in the late 1960s, introduced devices which contained hundreds of transistors on each chip, called "**Medium-Scale Integration**" (**MSI**).

They were attractive economically because while they cost little more to produce than SSI devices, they allowed more complex systems to be produced using smaller circuit boards, less assembly work (because of fewer separate components), and a number of other advantages.

Further development, driven by the same economic factors, led to "**Large-Scale Integration**" (**LSI**) in the mid 1970s, with tens of thousands of transistors per chip.

Integrated circuits such as 1K-bit RAMs, calculator chips, and the first microprocessors, that began to be manufactured in moderate quantities in the early 1970s, had under 4000 transistors. True LSI circuits, approaching 10000 transistors, began to be produced around 1974, for computer main memories and second-generation microprocessors.

VLSI



Upper interconnect layers on an Intel 80486DX2 microprocessor die.

The final step in the development process, starting in the 1980s and continuing through the present, was "very large-scale integration" (VLSI). The development started with hundreds of thousands of transistors in the early 1980s, and continues beyond several billion transistors as of 2009.

Multiple developments were required to achieve this increased density. Manufacturers moved to smaller rules and cleaner fabs, so that they could make chips with more transistors and maintain adequate yield. The path of process improvements was summarized by the International Technology Roadmap for Semiconductors (ITRS). Design tools improved enough to make it practical to finish these designs in a reasonable time. The more energy efficient CMOS replaced NMOS and PMOS, avoiding a prohibitive increase in power consumption. Better texts such as the landmark textbook by Mead and Conway helped schools educate more designers, among other factors.

In 1986 the first one megabit RAM chips were introduced, which contained more than one million transistors. Microprocessor chips passed the million transistor mark in 1989 and the billion transistor mark in 2005. The trend continues largely unabated, with chips introduced in 2007 containing tens of billions of memory transistors.

ULSI, WSI, SOC and 3D-IC

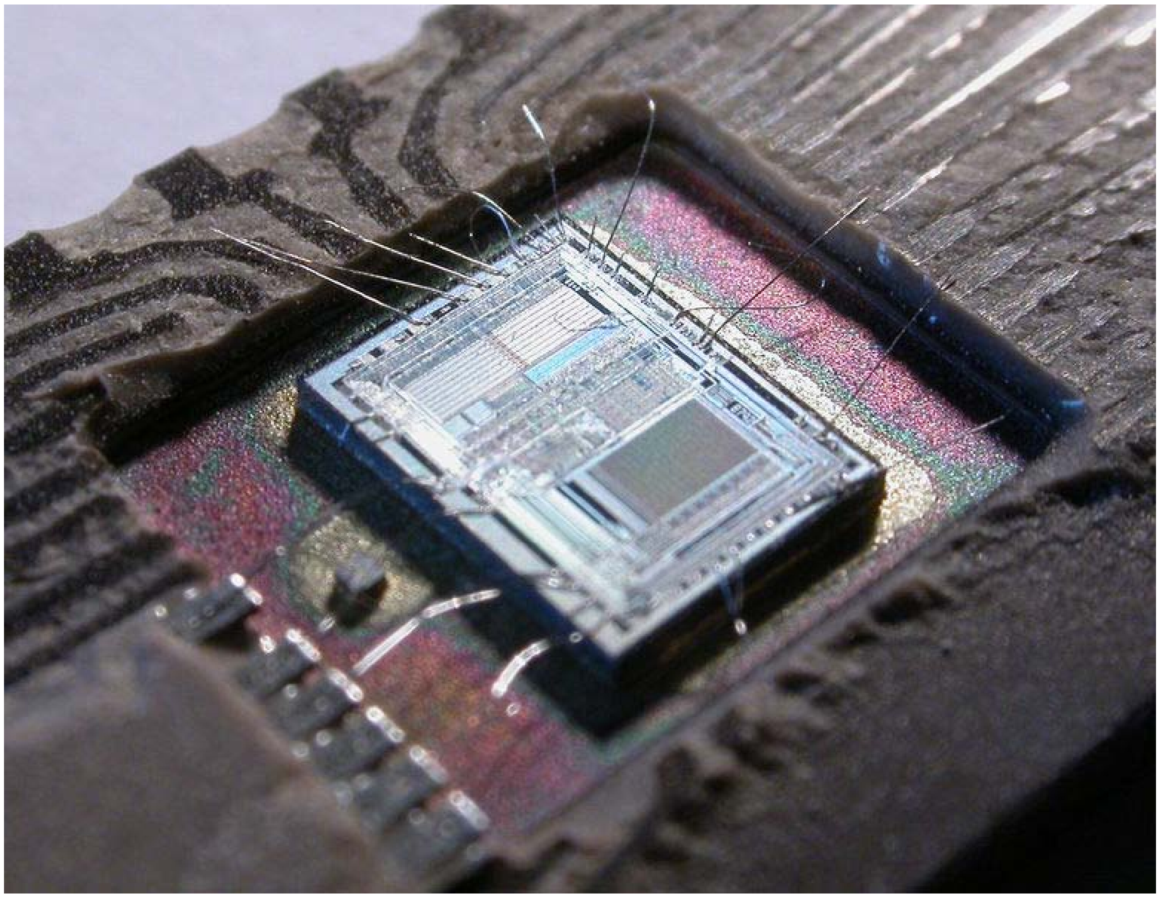
To reflect further growth of the complexity, the term *ULSI* that stands for "ultra-large-scale integration" was proposed for chips of complexity of more than 1 million transistors.

Wafer-scale integration (WSI) is a system of building very-large integrated circuits that uses an entire silicon wafer to produce a single "super-chip". Through a combination of large size and reduced packaging, WSI could lead to dramatically reduced costs for some systems, notably massively parallel supercomputers. The name is taken from the term Very-Large-Scale Integration, the current state of the art when WSI was being developed.

A system-on-a-chip (SoC or SOC) is an integrated circuit in which all the components needed for a computer or other system are included on a single chip. The design of such a device can be complex and costly, and building disparate components on a single piece of silicon may compromise the efficiency of some elements. However, these drawbacks are offset by lower manufacturing and assembly costs and by a greatly reduced power budget: because signals among the components are kept on-die, much less power is required.

A three-dimensional integrated circuit (3D-IC) has two or more layers of active electronic components that are integrated both vertically and horizontally into a single circuit. Communication between layers uses on-die signaling, so power consumption is much lower than in equivalent separate circuits. Judicious use of short vertical wires can substantially reduce overall wire length for faster operation.

Advances in integrated circuits



The die from an Intel 8742, an 8-bit microcontroller that includes a CPU running at 12 MHz, 128 bytes of RAM, 2048 bytes of EPROM, and I/O in the same chip.

Among the most advanced integrated circuits are the microprocessors or "**cores**", which control everything from computers and cellular phones to digital microwave ovens. Digital memory chips and ASICs are examples of other families of integrated circuits that are important to the modern information society. While the cost of designing and developing a complex integrated circuit is quite high, when spread across typically millions of production units the individual IC cost is minimized. The performance of ICs is high because the small size allows short traces which in turn allows low power logic (such as CMOS) to be used at fast switching speeds.

ICs have consistently migrated to smaller feature sizes over the years, allowing more circuitry to be packed on each chip. This increased capacity per unit area can be used to decrease cost and/or increase functionality—Moore's law which, in its modern interpretation, states that the number of transistors in an integrated circuit doubles every two years. In general, as the feature size shrinks, almost everything improves—the cost per unit and the switching power consumption go down, and the speed goes up. However, ICs with nanometer-scale devices are not without their problems, principal among which is leakage current, although these problems are not insurmountable and will likely be solved or at least ameliorated by the introduction of high-k dielectrics. Since these speed and power consumption gains are apparent to the end user, there is fierce competition among the manufacturers to use finer geometries. This process, and the expected progress over the next few years, is well described by the International Technology Roadmap for Semiconductors (ITRS).

In current research projects, integrated circuits are also developed for sensoric applications in medical implants or other bioelectronic devices. Particular sealing strategies have to be taken in such biogenic environments to avoid corrosion or biodegradation of the exposed semiconductor materials. As one of the few materials well established in CMOS technology, titanium nitride TiN turned out as exceptionally stable and well suited for electrode applications in medical implants.

Classification

A CMOS 4000 IC in a DIP

Integrated circuits can be classified into analog, digital and mixed signal (both analog and digital on the same chip).

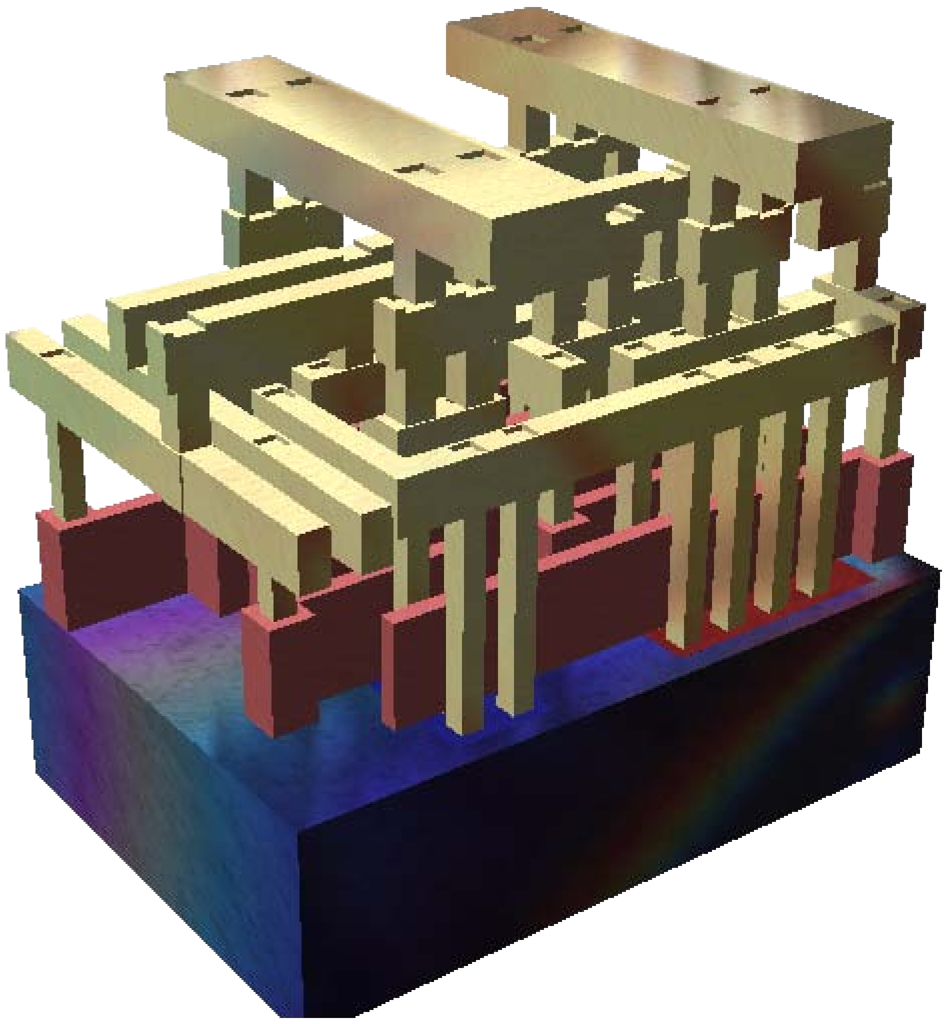
Digital integrated circuits can contain anything from one to millions of logic gates, flip-flops, multiplexers, and other circuits in a few square millimeters. The small size of these circuits allows high speed, low power dissipation, and reduced manufacturing cost compared with board-level integration. These digital ICs, typically microprocessors, DSPs, and micro controllers work using binary mathematics to process "one" and "zero" signals.

Analog ICs, such as sensors, power management circuits, and operational amplifiers, work by processing continuous signals. They perform functions like amplification, active filtering, demodulation, mixing, etc. Analog ICs ease the burden on circuit designers by having expertly designed analog circuits available instead of designing a difficult analog circuit from scratch.

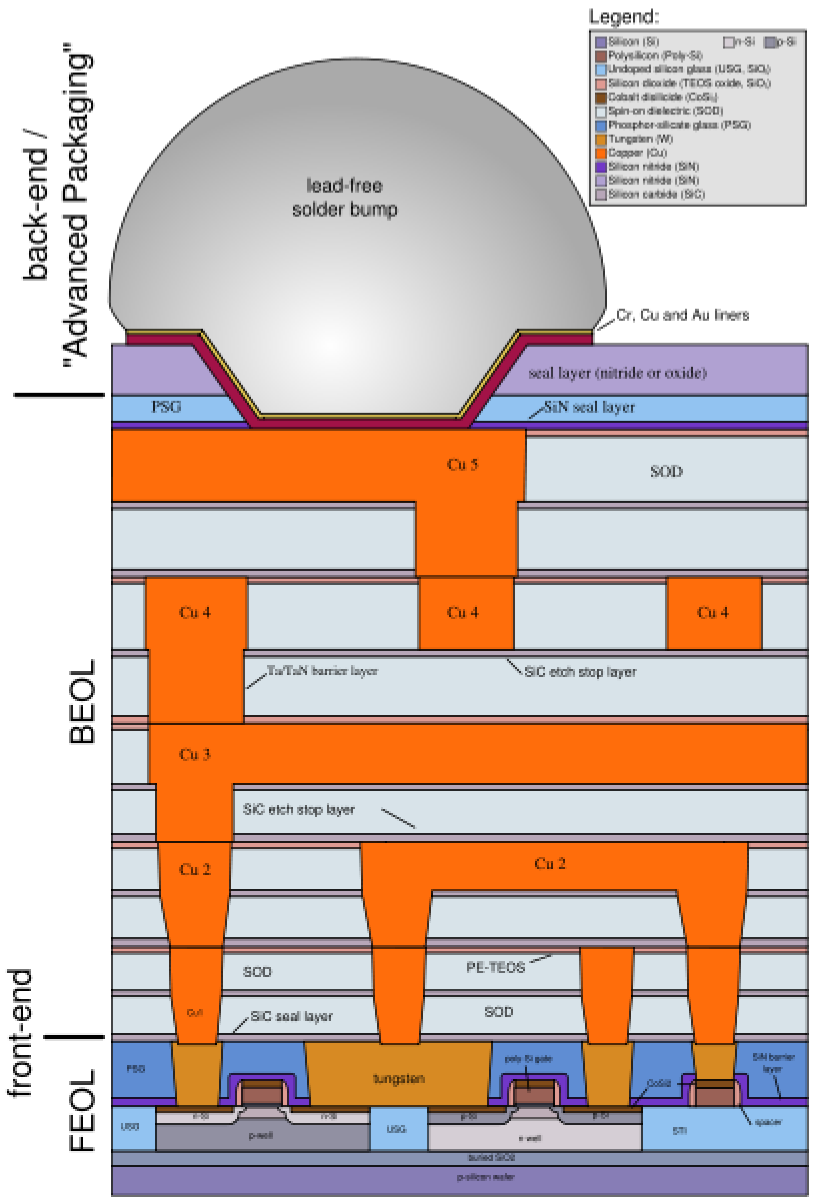
ICs can also combine analog and digital circuits on a single chip to create functions such as A/D converters and D/A converters. Such circuits offer smaller size and lower cost, but must carefully account for signal interference.

Manufacturing

Fabrication



Rendering of a small standard cell with three metal layers (dielectric has been removed). The sand-colored structures are metal interconnect, with the vertical pillars being contacts, typically plugs of tungsten. The reddish structures are polysilicon gates, and the solid at the bottom is the crystalline silicon bulk.



Schematic structure of a CMOS chip, as built in the early 2000s. The graphic shows LDD-MISFET's on an SOI substrate with five metallization layers and solder bump for flip-chip bonding. It also shows the section for FEOL (front-end of line), BEOL (back-end of line) and first parts of back-end process.

The semiconductors of the periodic table of the chemical elements were identified as the most likely materials for a *solid state vacuum tube*. Starting with copper oxide, proceeding to germanium, then silicon, the materials were systematically studied in the 1940s and 1950s. Today, silicon monocrystals are the main substrate used for *integrated circuits (ICs)* although some III-V compounds of the periodic table such as gallium arsenide are used for specialized applications like LEDs, lasers, solar cells and the highest-speed integrated circuits. It took decades to perfect methods of creating crystals without defects in the crystalline structure of the semiconducting material.

Semiconductor ICs are fabricated in a layer process which includes these key process steps:

- Imaging
- Deposition
- Etching

The main process steps are supplemented by doping and cleaning.

Mono-crystal silicon wafers (or for special applications, silicon on sapphire or gallium arsenide wafers) are used as the *substrate*. Photolithography is used to mark different areas of the substrate to be doped or to have polysilicon, insulators or metal (typically aluminium) tracks deposited on them.

- Integrated circuits are composed of many overlapping layers, each defined by photolithography, and normally shown in different colors. Some layers mark where various dopants are diffused into the substrate (called diffusion layers), some define where additional ions are implanted (implant layers), some define the conductors (polysilicon or metal layers), and some define the connections between the conducting layers (via or contact layers). All components are constructed from a specific combination of these layers.

- In a self-aligned CMOS process, a transistor is formed wherever the gate layer (polysilicon or metal) crosses a diffusion layer.

- Capacitive structures, in form very much like the parallel conducting plates of a traditional electrical capacitor, are formed according to the area of the "plates", with insulating material between the plates. Capacitors of a wide range of sizes are common on ICs.

- Meandering stripes of varying lengths are sometimes used to form on-chip resistors, though most logic circuits do not need any resistors. The ratio of the length of the resistive structure to its width, combined with its sheet resistivity, determines the resistance.

- More rarely, inductive structures can be built as tiny on-chip coils, or simulated by gyrators.

Since a CMOS device only draws current on the *transition* between logic states, CMOS devices consume much less current than bipolar devices.

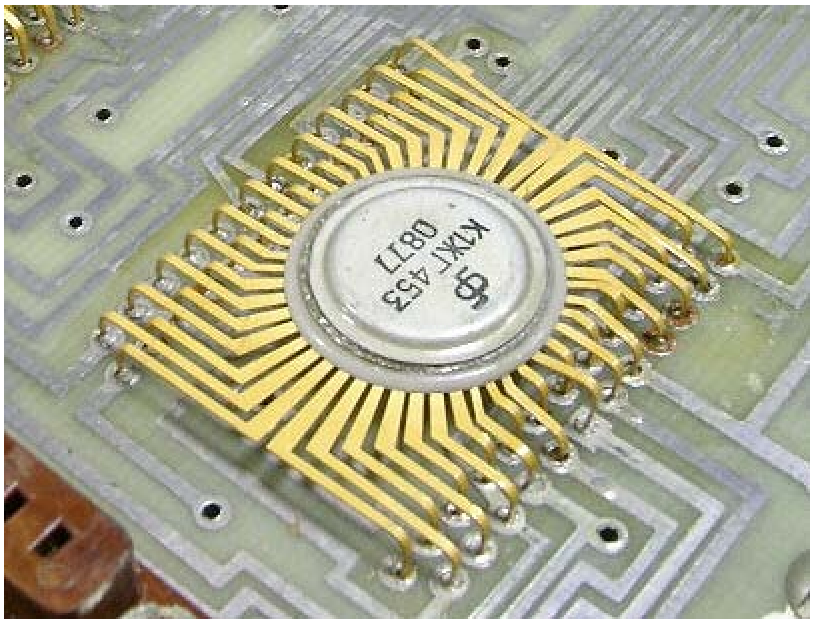
A random access memory is the most regular type of integrated circuit; the highest density devices are thus memories; but even a microprocessor will have memory on the chip. Although the structures are intricate – with widths which have been shrinking for decades – the layers remain much thinner than the device widths. The layers of material are fabricated much like a photographic process, although light waves in the visible spectrum cannot be used to "expose" a layer of material, as they would be too large for the features. Thus photons of higher frequencies (typically ultraviolet) are used to create the patterns for each layer. Because each feature is so small, electron microscopes are essential tools for a process engineer who might be debugging a fabrication process.

Each device is tested before packaging using automated test equipment (ATE), in a process known as wafer testing, or wafer probing. The wafer is then cut into rectangular blocks, each of which is called a *die*. Each good die (plural *dice*, *dies*, or *die*) is then connected into a package using aluminium (or gold) bond wires which are welded and/or Thermosonic Bonded to *pads*, usually found around the edge of the die. After packaging, the devices go through final testing on the same or similar ATE used during wafer probing. Test cost can account for over 25% of the cost of fabrication on lower cost products, but can be negligible on low yielding, larger, and/or higher cost devices.

As of 2005, a fabrication facility (commonly known as a *semiconductor fab*) costs over $1 billion to construct, because much of the operation is automated. The most advanced processes employ the following techniques:

- The wafers are up to 300 mm in diameter (wider than a common dinner plate).
- Use of 65 nanometer or smaller chip manufacturing process. Intel, IBM, NEC, and AMD are using 45 nanometers for their CPU chips. IBM and AMD are in development of a 45 nm process using immersion lithography.
- Copper interconnects where copper wiring replaces aluminium for interconnects.
- Low-K dielectric insulators.
- Silicon on insulator (SOI)
- Strained silicon in a process used by IBM known as strained silicon directly on insulator (SSDOI)

Packaging



Early USSR-made integrated circuit

The earliest integrated circuits were packaged in ceramic flat packs, which continued to be used by the military for their reliability and small size for many years. Commercial circuit packaging quickly moved to the dual in-line package (DIP), first in ceramic and later in plastic. In the 1980s pin counts of VLSI circuits exceeded the practical limit for DIP packaging, leading to pin grid array (PGA) and leadless chip carrier (LCC) packages. Surface mount packaging appeared in the early 1980s and became popular in the late 1980s, using finer lead pitch with leads formed as either gull-wing or J-lead, as exemplified by small-outline integrated circuit -- a carrier which occupies an area about 30 – 50% less than an equivalent DIP, with a typical thickness that is 70% less. This package has "gull wing" leads protruding from the two long sides and a lead spacing of 0.050 inches.

In the late 1990s, plastic quad flat pack (PQFP) and thin small-outline package (TSOP) packages became the most common for high pin count devices, though PGA packages are still often used for high-end microprocessors. Intel and AMD are currently transitioning from PGA packages on high-end microprocessors to land grid array (LGA) packages.

Ball grid array (BGA) packages have existed since the 1970s. Flip-chip Ball Grid Array packages, which allow for much higher pin count than other package types, were developed in the 1990s. In an FCBGA package the die is mounted upside-down (flipped) and connects to the package balls via a package substrate that is similar to a printed-circuit board rather than by wires. FCBGA packages allow an array of input-output signals (called Area-I/O) to be distributed over the entire die rather than being confined to the die periphery.

Traces out of the die, through the package, and into the printed circuit board have very different electrical properties, compared to on-chip signals. They require special design techniques and need much more electric power than signals confined to the chip itself.

When multiple dies are put in one package, it is called SiP, for *System In Package*. When multiple dies are combined on a small substrate, often ceramic, it's called an MCM, or Multi-Chip Module. The boundary between a big MCM and a small printed circuit board is sometimes fuzzy.

Chip labeling and manufacture date

Most integrated circuits large enough to include identifying information include four common sections: the manufacturer's name or logo, the part number, a part production batch number and/or serial number, and a four-digit code that identifies when the chip was manufactured. Extremely small surface mount technology parts often bear only a number used in a manufacturer's lookup table to find the chip characteristics.

The manufacturing date is commonly represented as a two-digit year followed by a two-digit week code, such that a part bearing the code 8341 was manufactured in week 41 of 1983, or approximately in October 1983.

Legal protection of semiconductor chip layouts

Like most of the other forms of intellectual property, IC layout designs are creations of the human mind. They are usually the result of an enormous investment, both in terms of the time of highly qualified experts, and financially. There is a continuing need for the creation of new layout-designs which reduce the dimensions of existing integrated circuits and simultaneously increase their functions. The smaller an integrated circuit, the less the material needed for its manufacture, and the smaller the space needed to accommodate it. Integrated circuits are utilized in a large range of products, including articles of everyday use, such as watches, television sets, washing machines, automobiles, etc., as well as sophisticated data processing equipment.

The possibility of copying by photographing each layer of an integrated circuit and preparing photomasks for its production on the basis of the photographs obtained is the main reason for the introduction of legislation for the protection of layout-designs.

A diplomatic conference was held at Washington, D.C., in 1989, which adopted a Treaty on Intellectual Property in Respect of Integrated Circuits (IPIC Treaty). The Treaty on Intellectual Property in respect of Integrated Circuits, also called Washington Treaty or IPIC Treaty (signed at Washington on May 26, 1989) is currently not in force, but was partially integrated into the TRIPs agreement.

National laws protecting IC layout designs have been adopted in a number of countries.

Other developments

In the 1980s, programmable integrated circuits were developed. These devices contain circuits whose logical function and connectivity can be programmed by the user, rather than being fixed by the integrated circuit manufacturer. This allows a single chip to be programmed to implement different LSI-type functions such as logic gates, adders and registers. Current devices named FPGAs (Field Programmable Gate Arrays) can now implement tens of thousands of LSI circuits in parallel and operate up to 1.5 GHz (Achronix holding the speed record).

The techniques perfected by the integrated circuits industry over the last three decades have been used to create microscopic machines, known as MEMS. These devices are used in a variety of commercial and military applications. Example commercial applications include DLP projectors, inkjet printers, and accelerometers used to deploy automobile airbags.

In the past, radios could not be fabricated in the same low-cost processes as microprocessors. But since 1998, a large number of radio chips have been developed using CMOS processes. Examples include Intel's DECT cordless phone, or Atheros's 802.11 card.

Future developments seem to follow the multi-core multi-microprocessor paradigm, already used by the Intel and AMD dual-core processors. Intel recently unveiled a prototype, "not for commercial sale" chip that bears 80 microprocessors. Each core is capable of handling its own task independently of the others. This is in response to the heat-versus-speed limit that is about to be reached using existing transistor technology. This design provides a new challenge to chip programming. Parallel programming languages such as the open-source X10 programming language are designed to assist with this task.

Silicon labelling and graffiti

To allow identification during production most silicon chips will have a serial number in one corner. It is also common to add the manufacturer's logo. Ever since ICs were created, some chip designers have used the silicon surface area for surreptitious, non-functional images or words. These are sometimes referred to as Chip Art, *Silicon Art*, *Silicon Graffiti* or *Silicon Doodling*.

Notable ICs and IC families

- The 555 common timing circuit
- The 741 operational amplifier
- 7400 series TTL logic building blocks
- 4000 series, the CMOS counterpart to the 7400 series
- Intel 4004, the world's first microprocessor, which led to the famous 8080 CPU and then the IBM PC's 8088, 80286, 486 etc.

- The MOS Technology 6502 and Zilog Z80 microprocessors, used in many home computers of the early 1980s
- The Motorola 6800 series of computer-related chips, leading to the 68000 and 88000 series (used in some Apple computers and in the 1980s Commodore Amiga series).