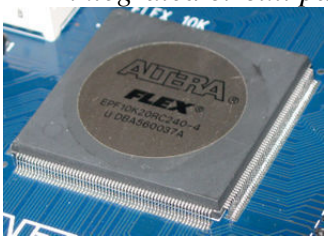
Field-programmable gate array

From Wikipedia, the free encyclopedia

By Zahra Maghsoodi

(Redirected from FPGA)
Jump to: navigation, search

FPGAs should not be confused with the flip-chip pin grid array, a form of integrated circuit packaging.



An Altera FPGA with 20,000 cells.

A **field programmable gate array** (FPGA) is a semiconductor device containing programmable logic components and programmable interconnects. The programmable logic components can be programmed to duplicate the functionality of basic logic gates such as AND, OR, XOR, NOT or more complex combinational functions such as decoders or simple math functions. In most FPGAs, these programmable logic components (or logic blocks, in FPGA parlance) also include memory elements, which may be simple flip-flops or more complete blocks of memories.

A hierarchy of programmable interconnects allows the logic blocks of an FPGA to be interconnected as needed by the system designer, somewhat like a one-chip programmable breadboard. These logic blocks and interconnects can be programmed after the manufacturing process by the customer/designer (hence the term "field programmable", i.e. programmable in the field) so that the FPGA can perform whatever logical function is needed.

FPGAs are generally slower than their application-specific integrated circuit (ASIC) counterparts, can't handle as complex a design, and draw more power. However, they have several advantages such as a shorter time to market, ability to re-program in the field to fix bugs, and lower non-recurring engineering costs. Vendors can sell cheaper, less flexible versions of their FPGAs which cannot be modified after the design is committed. The development of these designs is made on regular FPGAs and then migrated into a fixed version that more resembles an ASIC. Complex programmable logic devices, or CPLDs, are another alternative.

Contents

History

The historical roots of FPGAs are in complex programmable logic devices (CPLDs) of the early to mid 1980s. Ross Freeman, Xilinx co-founder, invented the field programmable gate array in 1984. CPLDs and FPGAs include a relatively large number of programmable logic elements. CPLD logic gate densities range from the equivalent of several thousand to tens of thousands of logic gates, while FPGAs typically range from tens of thousands to several million.

The primary differences between CPLDs 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. A related, important difference is that many modern FPGAs support full or partial in-system reconfiguration, allowing their designs to be changed "on the fly" either for system upgrades or for dynamic reconfiguration as a normal part of system operation. Some FPGAs have the capability of partial re-configuration that lets one portion of the device be re-programmed while other portions continue running.

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". 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. An alternate approach is to make use of "soft" processor cores that are implemented within the FPGA logic. These cores include the Xilinx MicroBlaze and PicoBlaze, the Altera Nios and Nios II processors, and the open source LatticeMico32 and LatticeMico8, as well as third-party (either commercial or free) processor cores. For a given CPU architecture, a hard (embedded) CPU core will outperform a soft-core CPU (i.e., a programmable-logic implementation of the CPU).

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. Current FPGA tools, however, do not fully support this methodology.

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. Other devices (such as Mathstar's Field Programmable Object Array, or FPOA) provide arrays of higher-level programmable objects that lie somewhere between an FPGA's logic block and a more complex processor.

Applications

Applications of FPGAs include DSP, software-defined radio, aerospace and defense systems, ASIC prototyping, medical imaging, computer vision, speech recognition, cryptography, bioinformatics, computer hardware emulation 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).

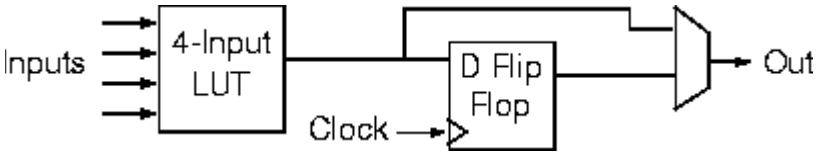
FPGAs especially find applications in any area or algorithm that can make use of the massive parallelism offered by their architecture. One such area is code breaking, in particular brute-force attack, of cryptographic algorithms.

Architecture

The typical basic architecture consists of an array of configurable logic blocks (CLBs) and routing channels. Multiple I/O pads may fit into the height of one row or the width of one column. Generally, all the routing channels have the same width (number of wires).

An application circuit must be mapped into an FPGA with adequate resources.

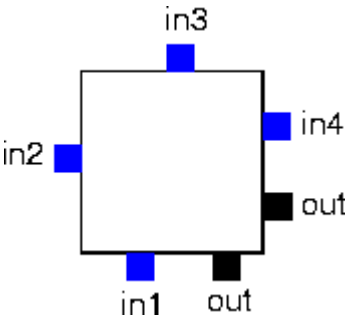
The typical FPGA logic block consists of a 4-input lookup table (LUT), and a flip-flop, as shown below.



Logic block

There is only one output, which can be either the registered or the unregistered LUT output. The logic block has four inputs for the LUT and a clock input. 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.

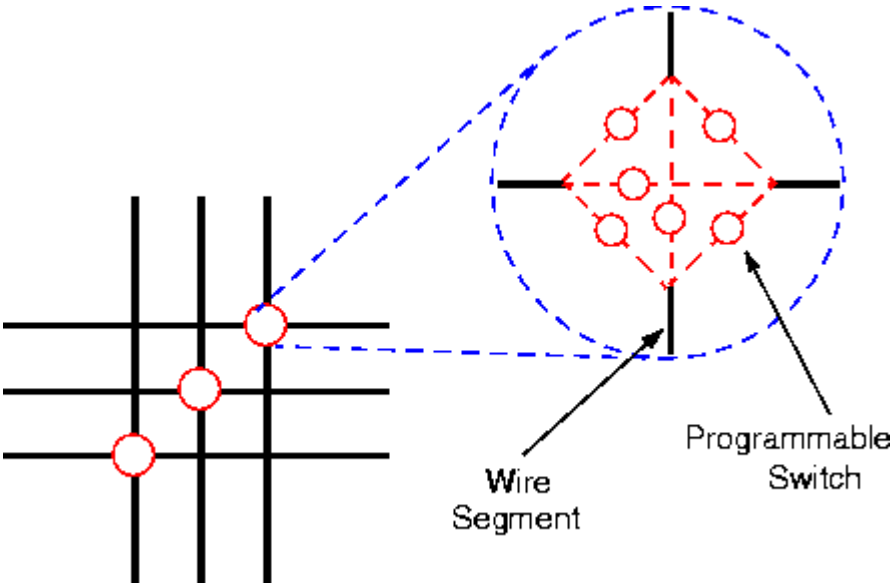
Each logic block output pin can connect to any of the wiring segments in the channels adjacent to it.

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. Common HDLs are VHDL and Verilog. 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.

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 of

the design. Companies such as Cadence, Synopsys and Celoxica are promoting SystemC as a way to combine high level languages with concurrency models to allow faster design cycles for FPGAs than is possible using traditional HDLs. Approaches based on standard C or C++ (with libraries or other extensions allowing parallel programming) are found in the Catapult C tools from Mentor Graphics, and in the Impulse C tools from Impulse Accelerated Technologies. Annapolis Micro Systems, Inc.'s CoreFire Design Suite provides a graphical dataflow approach to high-level design entry. Languages such as SystemVerilog, SystemVHDL, and Handel-C (from Celoxica) seek to accomplish the same goal, but are aimed at making existing hardware engineers more productive versus making FPGAs more accessible to existing software engineers.

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.org (typically "free", and released under 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 stimulate 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.
- EPROM - Erasable Programmable Read-Only Memory technology. Usually one-time programmable in production because of plastic packaging. Windowed devices 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.

FPGA manufacturers and their specialties

As of late 2005, the FPGA market has mostly settled into a state where there are two major "general-purpose" FPGA manufacturers and a number of other players who differentiate themselves by offering unique capabilities.

- Xilinx and Altera are the two big FPGA leaders.
- Lattice Semiconductor provides both SRAM and non-volatile, flash-based FPGAs.
- Actel has antifuse and reprogrammable flash-based FPGAs, and also offers mixed signal flash-based FPGAs.
- Atmel provides fine-grain reconfigurable devices, as the Xilinx XC62xx were. They focus on providing AVR Microcontrollers with FPGA fabric on the same die.

See also

- CPLD: Complex Programmable Logic Device
- FPAA: Field Programmable Analog Array
- VHDL: VHSIC (Very High Speed Integrated Circuit) Hardware Description Language
- Verilog: Hardware Description Language
- JHDL: Just-Another Hardware Description Language
- Embedded System Design in an FPGA
- configware