FPGAs and Emulation

Steven P. Smith

SoC Design

EE382V
Fall 2010

Overview

- Programmable Logic Devices –History and Types
- The CPLD and the FPGA
- FPGAs versus ASICs
- FPGA-based Emulation
- Conclusions

Early Programmable Logic Devices

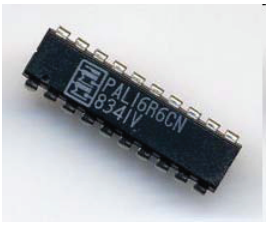
- Programmable Read-Only Memory (PROM) devices (1956)
  - Used to implement arbitrary combinational functions
    - Combinational inputs wire to PROM address bits
    - Combinational outputs driven by PROM data bits
    - PROM programmed to realize desired function
- Mask-programmable gate arrays (MPGA) were introduced by Motorola in 1969.
- Texas Instruments introduced a similar device in 1970.
  - TI coined "Programmable Logic Array" - PLA
- MPGAs are customized during fabrication by the device vendor.
  - Relatively high non-recurring engineering (NRE) charge and long lead times.

Early Programmable Logic Devices (2)

- In 1971, General Electric combined PROM technology with gate array structures to produce the first programmable logic device not requiring mask programming
  - Experimental only – never released
- The GE device was the first field programmable logic device.
  - Customized by the end-user
  - Low NRE costs and fast time-to-market

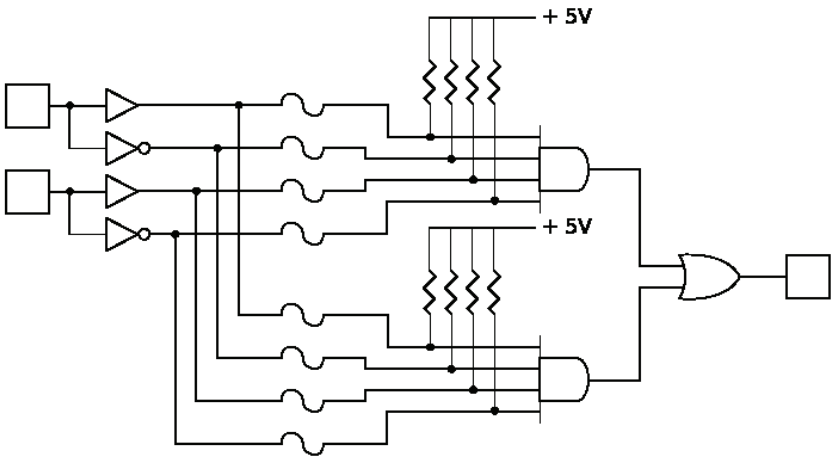
Early Programmable Logic Devices (3)

- Monolithic Memories, Inc. (MMI) incorporated GE ideas and introduced the Programmable Array Logic (PAL) device using fuses in 1978.
  - Structure of PALs was simpler and faster than earlier PLAs.
  - MMI developed a simple design flow and tools (PALASM).
  - PALs were a remarkable advance for board developers, enabling the elimination of SSI glue logic.



Early Programmable Logic Devices (4)

- PALs used two programmable planes, an AND and an OR plane.
- Each junction in the PAL was a fuse.



Simplified programmable logic device

Early Programmable Logic Devices (5)

- Data I/O introduced low cost, simple to use programmers

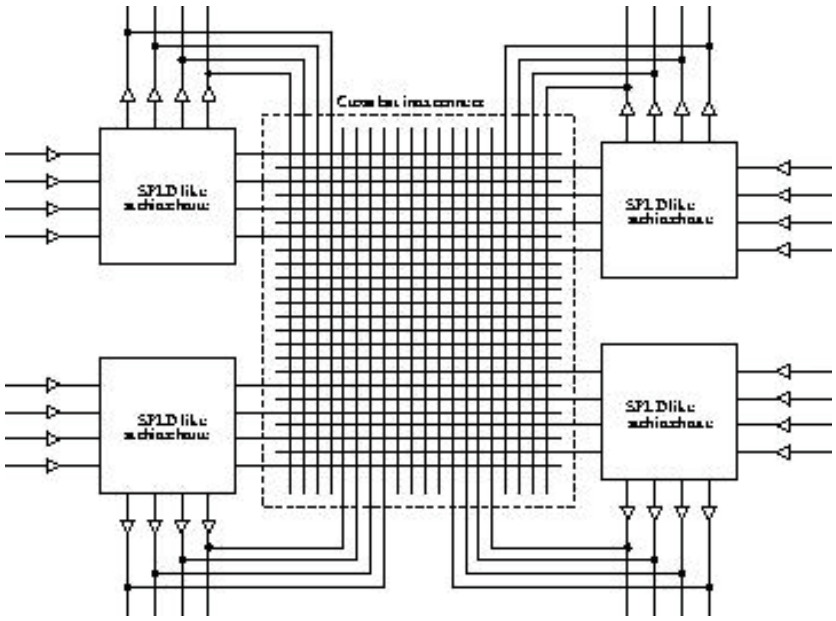
Early Programmable Logic Devices (6)

- Altera, formed in 1983, introduced the reprogrammable Electrically Programmable Logic Device (EPLD) in 1984.
- Lattice Semiconductor introduced Generic Array Logic (GAL) devices in 1985
  - Basically a reprogrammable PAL
- Complex Programmable Logic Device (CPLD) technology emerged in the mid 1980s, first released by Altera.
  - Basically a number of PAL-like structures with programmable interconnect.
- Xilinx, founded in 1984, introduced the first Field Programmable Gate Array (FPGA) in 1985, the XC2064.
  - Contained 64 complex logic blocks (CLBs), each with two 3-input look-up tables (LUTs)

Complex Programmable Logic Devices

- CPLDs have a higher gate density and less interconnect density than FPGAs and only limited support for multi-level logic.
- CPLDs offer better timing uniformity and are generally faster than FPGAs built in equivalent device technology.
- Modern CPLDs generally use non-volatile memory (reprogrammable) or fuse/anti-fuse (one-time programmable –OTP) technology for programming.
- Architecture typically combines coarse-grained simple programmable logic device structures with a programmable crossbar interconnect structure.
- Architecture doesn't scale well because of the crossbar interconnect.

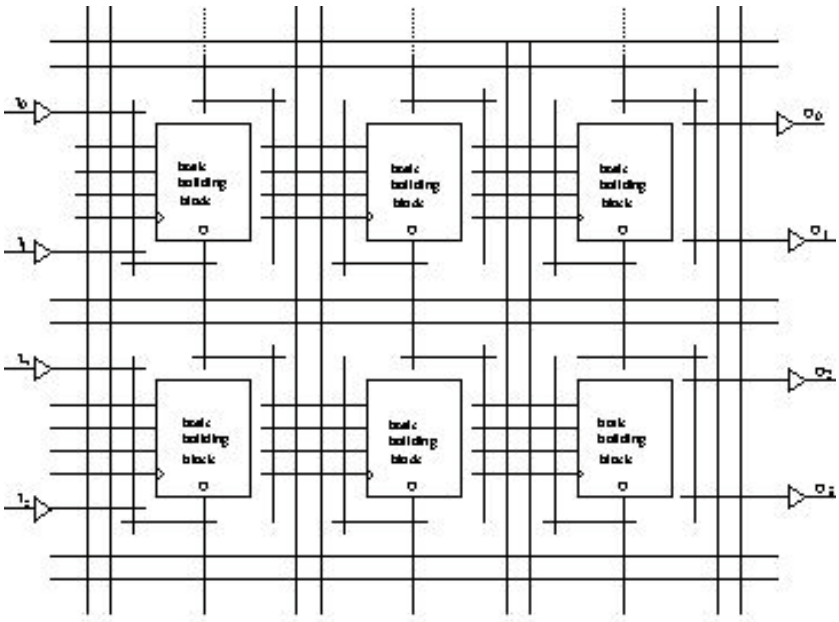
Complex Programmable Logic Devices



Field Programmable Gate Arrays

- Comparatively lower gate density with much more complex programmable interconnect capabilities than CPLDs.
- Basic architecture of FPGAs is a two-dimensional array of customizable logic blocks combined with an interconnect array.
  - Each logic block must offer functional completeness.
  - Logic blocks may be based on look-up tables or any other functionally complete behavior.
- Interconnect based on wire segments with interspersed switches for greater interconnect flexibility than CPLDs.
- The FPGA combines the advantages of the mask programmable gate array (MPGA) and the programmable logic device (PLD).

Field Programmable Gate Arrays



Field Programmability

- Field programmable capabilities derive from switches.
- Devices based on fuses (bi-polar) or anti-fuses (CMOS) are one-time programmable (OTP).
- Devices based on memory are reprogrammable.
  - Non-volatile memory-based devices support instant-on functionality (as do OTP devices) and don't require external memory to store device configuration information.
    - Flash, EPROM, or EEPROM
  - SRAM-based devices offer faster configuration, but require an external non-volatile memory to store configuration information.

FPGAs versus ASICs

- Design Time
  - FPGA-based design time is typically much shorter than that required for an ASIC.
    - ~Less than a year versus 2-3 years for an ASIC.
- Development Cost
  - No NRE for FPGAs versus $millions for an ASIC.
- Device Unit Cost
  - Unit costs for FPGAs are much higher than those for ASICs.
- Power Consumption: ~7 times dynamic power*
- Area Consumption: ~18 times the area*

* Kuon, I.; Rose, J. (2006). Measuring the gap between FPGAs and ASICs. Intl. Symposium on FPGAs, 2006.

FPGAs versus ASICs: Trade-Offs

- Time-to-Market
- Development Costs
- Development Risk
  - Cost of design spins
- Anticipated Unit Volume
  - NRE versus recurring expenses:
    - Project cost = NRE + (RE * Volume)
  - A question of "when the lines cross"
- IP protection
  - ASICs at risk during third-party fabrication
  - FPGAs may be more vulnerable when fielded, although the latest devices include bit stream encryption using AES or 3DES.

FPGAs versus ASICs

- Development costs and risks have led to a dramatic decline in ASIC design starts:
  - 11,000 in 1997
  - 1,500 in 2002
- As a percentage of the logic market, FPGAs have grown from 10% to approximately 25% in recent years.
- FPGAs are the fastest growing segment of the semiconductor market.

Major FPGA Device Vendors

- Xilinx and Altera are market leaders, controlling some 80% of the FPGA and CPLD market.
  - SRAM, non-volatile, and OTP devices
  - New Virtex-7: 28 nm process, 1.0 V, up to 8 Mbytes RAM, up to 1200 user I/O pins, up to 2.4 M flip-flops, up to 1.9 M logic cells.
- Actel offers anti-fuse and flash-based devices.
  - Igloo and Igloo Nano devices have very low power and sophisticated sleep mode options
    - Finally a programmable logic solution suitable for battery-powered applications.
- Lattice Semiconductors offers SRAM-based devices with integrated configuration flash

Partial Dynamic Reconfiguration

- Xilinx first released the XC6000 in the mid 1990s, the first device capable of *partial* reconfiguration.
  - The device continues to operate functionally while portions of it are reconfigured.
  - XC6000 had comparatively fine-grained addressing of the configuration fabric. Newer devices, beginning with the Virtex-2 Pro, had more coarse addressability at the bank or slice level.
  - Xilinx support for partial dynamic reconfiguration has been sporadic and tentative at best. Repeatedly, they've announced tool and AE support only to determine it's too burdensome.
  - They're currently supporting it (again, and only if $10K/year is paid for access), and for the first time the tools actually help.

Partial Dynamic Reconfiguration (2)

- Altera has not yet developed devices capable of partial reconfiguration, although there are (again) rumors that they may.
- Partial reconfiguration has many potential applications:
  - Key and algorithm specific cipher engines (use one engine instance to decode the configuration bit set for another engine without going offline)
  - Self-modifying, dynamic instruction set architectures
  - Scene-specific image analysis
  - Dynamically configurable HW accelerators for FPGA-based SoCs

FPGA-Based Emulation

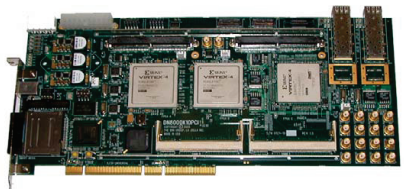
- FPGA-based emulation is typically viewed as a higher performance alternative to SW-based simulation for verification of complex systems.
- Complex SoCs are impractical to simulate at the whole-system level.
  - Simulation more tractable at the block level
  - SoCs depend upon complex interactions between SW and among disparate HW elements.
    - Execution of application SW required
- Emulation is 100 to 10,000 time faster than SW-based simulation.

FPGA-Based Emulation

- Verification now accounts for roughly 70% of SoC design effort.
- Complex SoCs are impractical to simulate at the whole-system level.
  - Simulation more tractable at the block level
  - SoCs depend upon complex interactions between SW and among disparate HW elements.
    - Execution of application SW usually required
- Emulation is on the order of 50 to 10,000 time faster than SW-based simulation.

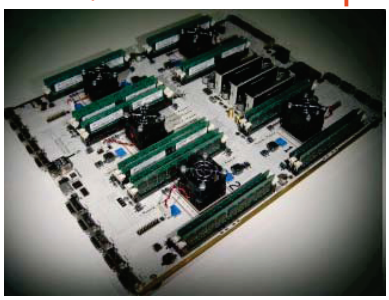
FPGA-Based Emulation Platforms

- Platform can range from a simple FPGA development kit to a complex, highly expensive emulation system with many FPGA devices.
- High-end, multi-FPGA systems
  - Ikos (now part of Mentor Graphics)
  - BEE - Berkeley Emulation Engine (UC Berkeley)
  - BEEcube (spin out from Berkeley)
- Single FPGA systems
  - FPGA evaluation/development kits



Berkely Emulation Engine

- BEE2 (2004)
  - Each module (pictured) hosts 5 Xilinx Virtex-2 Pro 70 FPGAs. 1 FPGA for system control, 4 for user models
  - Multiple modules use a non-blocking Infiniband crossbar for communications
- BEE3 (2007)
  - Based on Virtex-4, shows 4-6x performance gains



FPGA Emulation Challenges

- Most "interesting" designs will require multiple FPGAs for the purposes of emulation.
- The quality of the partitioning of behaviors across FPGAs is a primary determinate of emulation performance.
  - Tool support is vendor-specific and not always particularly effective.
  - Often the difference between 10 MHz system clock speeds and 400 MHz rates
  - Manual intervention often necessary to obtain improved performance, which may be costly and time consuming
- Interface signals among FPGAs may be insufficient for otherwise more optimal partitions.

FPGA Emulation Challenges (2)

- HDL targeting ASIC design flows doesn't always map easily into FPGAs.
  - Clock and initialization logic may need to be rewritten for emulation environment.
  - Memory technology and I/O interfaces may differ
    - E.g., intended implementation uses flash for code storage but emulation only has DRAM
  - Bus models and their implementation may differ.
    - Generally no tri-state signals internal to FPGAs
  - Debug support may require addition of HDL to support controllability and visibility where needed.
- Best practice is to develop HDL with both FPGA and ASIC in mind.

FPGA Emulation Challenges (3)

- Third-party IP may not be available in suitable (HDL) form.
  - Co-verification interface to simulator or C/C++ model running on a general-purpose host
    - Always ends up being gating factor on performance, severely constraining achievable emulation speeds
  - Discrete HW instantiation of third-party IP may require custom interface and models
  - May need to use different processor architecture (e.g., an FPGA's internal PowerPC hard core instead of, say, a target ARM processor) for performance
- Emulation speed may be limited by data collection I/O bottlenecks.

FPGA Emulation Challenges (4)

- Design partitioning and generation of bit streams may be time consuming
  - Recompilation may take hours (or worse)
- In the end, you are executing an approximate model of your target SoC or ASIC.
  - Important to bear this fact in mind when interpreting results.
  - Still need to do extensive verification through simulation of those blocks known to be different between the emulated system and the target design.
  - Same is true for interfaces between blocks and clock and reset logic.

FPGA Emulation Challenges (5)

- Emulating systems that interact with the environment or with other systems
  - If emulated speed is less than the target operational speed, need to consider the impact on real-time operation.
  - Network interfaces can often be scaled to retain effective equivalence with real-time operation
    - E.g., Use 10 Mbps Ethernet on emulator running at 1/10 the rate of the target operational speed which is intended to work with 100 Mbps networks

FPGA Emulation Benefits

- Unmatched performance
- Cost effective, especially if FPGA evaluation boards are used as an ad hoc emulator
  - Commercial system can be quite expensive, but are still cheaper than an extra ASIC spin.
- Robust verification possible
- Application software may be used in verification process, where it is typically impractical for simulation
- Reduces design risk for ASICs
- Facilitates the fastest path to the market for complex SoC design

Conclusions

- Programmable logic devices emerged in the 1970s and have advanced steadily since.
- CPLDs and FPGAs have fundamentally different structures and, typically, different applications.
- The emergence of very low-power devices has opened up potential applications in battery-powered applications, previously a complete non-starter.
- FPGAs are the fastest growing segment of the semiconductor market.
- All but very high volume consumer applications are likely better served by FPGAs than ASICs.
- Emulators offer an effective and powerful means of reducing design risks, development time and costs for ASIC designs.