Programming Modern FPGAs

Ivo Bolsens
Xilinx
MPSOC August, 2006
Outline

• Modern FPGA
• FPGA programmable platform
• Programming the FPGA
• Conclusions
Modern FPGA

- 65nm technology, 40-nm gate length (Poly)
- 1.6nm oxide thickness (16 Angstrom)
  - ~5 atomic layers
- Triple-Oxide Technology
  - 3 oxide thicknesses for optimum power and performance
- 1.0 Vcc core
  - Lower dynamic power
- 12 layer copper
- Strained silicon transistor
  - Maximum performance at lowest AC power

Over 1 Billion Transistors
FPGA Roadmap

New process technology drives down cost
FPGAs can take advantage of new technology faster than ASICs and ASSPs

FPGA 2010: 32 nm, 5 Billion transistors

The cost of IC development increases. Therefore customers want to buy reconfigurable and programmable platforms, instead of developing their own.
FPGA Fabric

- High-Performance Logic Fabric
- Many Configuration Options
- 36Kbit Dual-Port Block RAM / FIFO with ECC
- 25x18 Multiplier DSP Slice with Integrated ALU
- Gigabit Serial Transceivers
- General IO with ChipSync + XCITE DCI
- 550 MHz Clock Management Tile DCM + PLL
- Gigabit Serial Transceivers

MPSOC 2006 slide 5
Logic Architecture

True 6-input Lookup Table (LUT)
with dual 5-input LUT option

64-bit RAM per M-LUT
about half of all LUTs

32-bit or 16-bit x 2
shift register per M-LUT
Virtex-5 Routing

Symmetric pattern, connecting CLBs

Same pattern for all outputs

Fast Connect
1 Hop
2 Hops
3 Hops
General Purpose I/O (Select I/O)

- All I/O pins are “created equal”
- Compatible with >40 different standards
  - Vcc, output drive, input threshold, single/differential, etc
- Each I/O pin has dedicated circuitry for:
  - On-chip transmission-line termination (serial or parallel)
  - Serial-to-parallel converter on the input (CHIPSYNC)
  - Parallel -to-serial converter on the output (CHIPSYNC)
  - Clock divider, and high-speed “regional” clock distribution

*Ideal for source-synchronous I/O up to 1 Gbps*
Platform FPGAs
Digital System Design Simplified

System Design
- RTL
- HW / SW partition
- 0, 1 and delay
- High-level synthesis
- Standards and interfaces
- Timing

Platform FPGA
- Embedded IP
- DFM
- IR drop
- Noise Margin
- Termination
- Clock generation
- Repeaters
- Transmission lines
- Clock distribution
- Crosstalk
- Startup init
Xilinx Strategic Directions

Integration
Hard IP
System Tools

New

Existing

APPS

Embedded Processor
Gb Transceivers
DSP
Algorithmic Logic
Glue Logic

Existing
• Network Infrastructure
• Computing Infrastructure
• Industrial, medical
• Military

New
• Consumer Electronics
• Automotive
• Portable

Cost
Power
Quality

Markets

Time
Domain Optimized Platforms
One Family – Multiple Platforms

Column based features

LX
Logic Domain
Highest logic density

SX
DSP Domain
Highest DSP performance

FX
Connectivity Domain
Embedded Processors
High-speed Serial I/O

 Enables “Dial-In” hard IP Mix
Logic, DSP, BRAM, I/O, MGT, DCM, PowerPC

Enabled by Flip-Chip Packaging
I/O Columns Distributed Throughout the Device
8 MPEG4 decoders

- Eight Ports of Compressed Video In
- Memory Controllers
- RAM
- Off Chip Frame Memories
- Eight Ports of De-Compressed 720p Video Out
- Mpeg4 Decoder
Application Example: MicroBlaze 5.0

- 1400 LUT6
- 230 Dhrystone Mips
- > 200 fit in V5
Future Proof Architecture

- Parallelism
  - Performance & Power
- Distributed Memory
  - Data transfer bottleneck
- Regular
  - Manufacturability
  - Redundancy
- Scalable
  - Future Proof
- 2010
  - 5 cent/32bit MB
  - 2$ for 1 Mgates

"If FPGAs didn’t exist today, people would have to invent them…"
FPGA for Embedded Systems

• An embedded system is a system that
  – has a complex concurrent behavior
  – is characterized by stringent timing requirements
  – has non-trivial communication between its components and the rest of the world
Outline

• Modern FPGA
• FPGA programmable platform
• Programming the FPGA
• Conclusions
FPGA Memory Options
Choose the Right Memory for the Application

**Distributed RAM/SRL32**
- Very granular, localized memory
- Minimal impact on logic routing
- Great for small FIFOs

**On-chip BRAM/FIFO**
- Efficient, on-chip blocks
- Flexible + optional FIFO logic
- Ideal for mid-sized FIFOs/buffers

**Fast Memory Interfaces**
- Cost-effective bulk storage
- Memory controller cores
- Large memory requirements

Granularity ➔ Capacity

**Virtex-5**
- DRAM
- SDRAM
- DDR SDRAM
- FCRAM
- RLDRAM
- SRAM
- Sync SRAM
- DDR SRAM
- ZBT
- QDR
- FLASH
- EEPROM
Memory Bandwidth Envelope

- Bandwidth to Registers: 500x that of a processor register file
- Bandwidth to LUT-Rams: 50x that of L1 cache of processor
- Bandwidth to BRAMS: 5x that of L1 to L2 cache of a processor

Intel; Xilinx
Programmable interconnect

- Can connect compute and registers, small memory and larger memory \textit{arbitrarily}
- 80\% of the FPGA resource, but often neglected as the key differentiator
- Contrast this with processors: 4 pre-specified architectural (von Neumann) \textit{bottlenecks}.

\begin{center}
\begin{tikzpicture}
  \node[draw, text centered] (ALUs) at (0,0) {ALUs};
  \node[draw, text centered] (REGs) at (1.5,0) {REGs};
  \node[draw, text centered] (L1) at (3,0) {L1};
  \node[draw, text centered] (L2) at (4.5,0) {L2};
  \node[draw, text centered] (Mem) at (6,0) {Mem};

  \draw[->, thick] (ALUs) -- (REGs);
  \draw[->, thick] (REGs) -- (L1);
  \draw[->, thick] (L1) -- (L2);
  \draw[->, thick] (L2) -- (Mem);
\end{tikzpicture}
\end{center}
## FPGA vs Microprocessor

<table>
<thead>
<tr>
<th></th>
<th>Microprocessor</th>
<th>FPGA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Itanium 2</td>
<td>Virtex 2VP100</td>
</tr>
<tr>
<td>Technology</td>
<td>0.13 Micron</td>
<td>0.13 Micron</td>
</tr>
<tr>
<td>Clock Speed</td>
<td>1.6GHz</td>
<td>180MHz</td>
</tr>
<tr>
<td>Internal Memory Bandwidth</td>
<td>102 GBytes per Sec</td>
<td>7.5 TBytes per Sec</td>
</tr>
<tr>
<td># Processing Units</td>
<td>5 FPU(2MACs + 1FPU)</td>
<td>212 FPU or 300+ Integer Units</td>
</tr>
<tr>
<td></td>
<td>+ 6 MMU + 6 Integer Units</td>
<td>or ...............</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>130 WATTS</td>
<td>15 WATTS</td>
</tr>
<tr>
<td>Peak Performance</td>
<td>8 GFLOPs</td>
<td>38 GFLOPS</td>
</tr>
<tr>
<td>Sustained Performance</td>
<td>~2 GFLOPs</td>
<td>~19 GFLOPS</td>
</tr>
<tr>
<td>I/O / External Memory Bandwidth</td>
<td>6.4 GBytes/sec</td>
<td>67 GBytes/sec</td>
</tr>
</tbody>
</table>

Courtesy Nallatech
High Performance Compute

- Computation (GOPS)
- Memory Bandwidth (GB/sec)
- IO Bandwidth (Gbps)

Comparison of Pentium, V2Pro, and V4:}

- Pentium
- V2Pro
- V4
Processor Use Models

1. State Machine
   - Lowest Cost, No Peripherals, No RTOS & No Bus Structures
   - VGA & LCD Controllers
   - Low/High Performance

2. Microcontroller
   - Medium Cost, Some Peripherals, Possible Bus Structure
   - Control & Instrumentation
   - Moderate Performance

3. Custom Embedded
   - Highest Integration, Extensive Peripherals, RTOS & Bus Structures
   - Networking & Wireless
   - High Performance
Application-Specific Hardware Acceleration

- When the processor core begins to reach software task capacity, then Fabric Acceleration to the rescue
  - Use Fast Simplex Link (FSL) to interface to customer-defined accelerators
  - Enables dramatic improvements in performance
PowerPC Architecture

Full System Customization & High Performance
Comparison with Traditional Bus-based

**APU**

- Processor Block → APU I/F → Soft Aux. Processor
  - Write Instruction and operands: 1 APU cycle
  - Execution: $N_{EX}$ APU cycle
  - Read Result and Status: 1 APU cycle + 1 CPU cycle

**PLB**

- Processor Block → APU I/F → Soft Aux. Processor
  - Write Operand1: 5 PLB cycles + 2 CPU cycles
  - Write Operand2 and Instruction: 5 PLB cycles + 2 CPU cycles
  - Execution: $N_{EX}$ PLB cycle
  - Read Status: 6 PLB cycles + 3 CPU cycles
  - Read Result: 6 PLB cycles + 3 CPU cycles
Processor Performance and Fabric Acceleration

- PowerPC 405 at 450 MHz
- APU
- MicroBlaze - FSLs

Fabric Acceleration
PowerPC – APU
MicroBlaze - FSLs

Next generation
PowerPC

“Traditional”
Frequency Scaling

Virtex-II Pro
Virtex-4
Virtex-5

2002
2004
2006
2008
2010
Reconfigurable System

- Fixed configuration
  - Data loads from PROM or other source at power on
  - Configuration fixed until the end of the FPGA duty cycle
- Used extensively during traditional design flow
  - Evaluate functionality of design as it is developed
Dynamic Partial Reconfiguration

- A subset of the configuration data changes…
  - But logic layer continues operating while configuration layer is modified…
  - Configuration overhead limited to circuit that is changing…
Read / Modify / Write

1. Read back frame and load into BRAM
2. Modify configuration data in BRAM
3. Write modified frame to configuration memory
4. Repeat “Read-Modify-Write” sequence for all frames
Outline

• Modern FPGA
• FPGA programmable platform
• Programming the FPGA
• Conclusions
Programming FPGAs

Just a Matter Of Software
Bridging the Gap

System expert

Misery of details

the full power of the FPGA
Domain Specific Flow

System expert

Programming models

tools

Soft architecture

the full power of the FPGA
Domain Specific Models

- Networking perspective: The racing track pit stop
  - lots of concurrent threads (=engineers), on individual packets (=cars)

- DSP perspective: The manufacturing line
  - Lots of data tokens (=cars), processed in a pipelined fashion (=dataflow)

- Processor perspective: Human operator
  - Central control (=human), accelerators (=tools)

Different application domains require different methodologies to exploit capabilities hardware
Programming models

- Network Processing: concurrent application of rules to packets

- Digital Signal Processing: concurrent compute on streams of samples, e.g. video pixels

- High Performance Computing: concurrent compute with random access on datasets; compute with floating point and complex numbers
Architecture components

• Network Processing: FSM, micro-coded datapath, processors, pipelines, wide datapaths
• Digital Signal Processing: buffers with flow control, FSM, processors, synthesized expressions, fixed point
• High Performance Compute: Partition the algorithm, specialized instructions, small efficient cache components, floating point units
Bridging the gap

- Domain-specific data model and programming language
- API to access features of the domain specific soft architecture

Hyper-programmed soft architecture

Efficiently exploit logic, immersed IP, processing blocks, memory, interconnection, and programmability of FPGA
Abstracting away FPGA detail

Before:
• Complex single C program
• PPC/CoreConnect detail, and special PLB interface block
• Complex BPort backend block written in VHDL
• Tailored inter-block interfaces

After:
• Blocks with C code modules
• Blocks from existing IP cores
• Just click the blocks together

Old: User busy with FPGA details
e.g. clocks, bus protocols, memory map

New: User focuses on application,
including system performance
Challenge

• Specifying complex computational algorithms in a way that...
  ... is productive,
  ... permits efficient implementation on FPGAs,
  ... allows leveraging enormous concurrency of an FPGA,
  ... provides portability
    • across alternative implementations (e.g. fabric vs processor)
    • across different devices
Productivity

• Quality of result (QoR) is a design constraint
  – Performance, power, cost budgets make QoR a design constraint
• The real problem is to meet the QoR target and minimize:
  – Non-recurring engineering costs (NRE)
  – Time-to-market (TTM)
• Methodology saves design cost by enabling
  – Design of portable, retargetable, composable IP blocks
  – Rapid design space exploration and system composition
Technical Challenges

• Methodology must address...
  – implementability, concurrency, portability

• ---a programming model that...
  – ... is simple to understand.
  – ... matches the application domain.
  – ... exposes essential architectural detail, hides the rest.
  – ...induces the programmer to make the right choices.
Combine the Best of Both Worlds
Software - Hardware

- Encapsulation
- Abstraction
- Portability
- Re-use

Implementation Detail
- Control Logic
- Interface Glue
- Concurrency
- Communication
- Architecture

- Events
- Protocols
- Ordering
- Sequential execution
- Clocks
- Signals
- Timing

Combining the strengths of both paradigms results in a radical improvement in hardware/software system design productivity.
DSP solution space

Best Clock-to-Sample Ratio

“Massive parallelism often allows FPGAs to handle data rates much higher than what DSPs and general-purpose processors can manage, and in today’s world of rapidly evolving applications and standards FPGAs’ programmability is an advantage over hard-wired solutions.”

- Amit Shoham, BDTI, June 15, 2005

DSP: Actor/Dataflow Programming

- Actors
- Guarded atomic actions
- Autonomous schedule
- Encapsulated state
- Point-to-point, buffered token-passing connections

UC Berkeley (Janneck et al)
Benefits of the Actor Model

- Dataflow is a natural concept in DSP.
- Explicit description of concurrency and disciplined access to shared state make design and debug of concurrent systems feasible.
- Complete abstraction of time.
- Extensive abstraction of control.
- Same description can target HW and SW implementations.
- Can be visualized easily
- Works naturally with run-time reconfiguration.
Actor/Dataflow Implementation

class MyActor
{
  schedule();
  readPort( portNum );
  writePort( portNum );
}

software + network

high-level synthesis

simulation

hardware

actor source + network
Concurrent Model

- Model entered as
  - Hierarchical, structural composition of actors.
  - Textual code for actor contents.
- Verified with dataflow simulation (Ptolemy-II).
Network Processing Solutions

2005 2007 2009

FPGA
Flexible system on chip used for tailored system architectures

Network processor (NPU)
Processor / memory bottlenecks worsen

Packet processing per second

Today: FPGA: logic bound NPU: architecture bound

20m IP packets routed per sec.

2005 2007 2009
Flexibility and scalability

Scalable in performance, and re-usable solution

FPGA
Flexible system on chip used for tailored system architectures

Fixed processor
Processor / memory bottlenecks worsen

No re-use, architecture dependent

Next fixed architecture

2005 2007 2009
Processing rate

Next fixed architecture

No re-use, architecture dependent
Flexibility opportunity

• “Despite the modest size of the NPU market, the recent trend in this market has surprisingly been segmentation. Vendors are discovering that a single general-purpose network processor cannot meet the needs of a broad set of applications. New products, and even some vendors, now focus on a single segment: access, metro, or enterprise.” - The Linley Group, January 2006
Example: typical line card

Network interface block and physical interface

Programmed block specified in high-level PitStop language

FPGA

- Traffic Classifier
- Policy Engine
- Packet Statistics
- Traffic Policing
- Packet Manipulation
- Ingress/Egress Queuing and Scheduling, traffic shaping
- Security

Embedded processor(s)

Software Interface, API

Specialized highly parameterized block

Blocks (plus other glue) assembled into system, by compilation of high-level Click language description
Network Packet Processing: Object Oriented Programming

E.g. Click programming

Each block is described as a Click element

Connections are made between elements, forming a graph

Can be used to describe designs at different granularities: from coarse-grain blocks to fine-grain blocks

MIT (Kohler et al)
Compact description in Click

IP router with ICMP offload

Graphical representation:

```
FromDevice(GMAC0) -> IPC::IPClassifier -> Queue -> ToDevice(GMAC0);
IPC[2]  -> ICMPHandler::EmbeddedSoftware(ICMP.c);
FromDevice(GMAC1) -> [1]IPC[1]  -> Queue -> ToDevice(GMAC1);
ICMPHandler  -> [2]IPC;
```
Top-level architecture choice

High-level packet processing description language

Example 1: Protocol stack handling in end system style setting

Collection of communicating threads implemented by logic or processor:
- prioritize low total latency
- ... then throughput (say, 1 Gb/s)
- one thread per protocol

Example 2: Layer 2 packet handling in line card style setting

Blocks arranged in pipeline implemented by logic:
- prioritize throughput (say, 10+ Gb/s)
- ... then latency
- staged according to dependencies
**Example: IP-enabled DSLAM**

**Downstream processing pipeline**
- Parsing, key extraction, initiate search
- Prepending search result
- VLAN processing
- LLC, SNAP encapsulation

**Search processing**
- ConnectionTable: 256 contexts overall
  - For each context, the chosen type of encapsulation technique is stored, as well as VLAN processing information, QoS, ATM VC, VP, port number, and whether this is an ADSL or VDSL port.

**Upstream processing pipeline**
- IP TOS-DCSP Modification, Checksum, TTL
- VLAN processing
- LLC, SNAP decapsulation
- Prepending search result
- Parsing, key extraction, initiate search

**FPGA**

**GEMAC**

**AAL5 SAR**
Results

Quantifiable:

• silicon cost: competitive (comparable to a low end NPU)
• performance: easily achieve 6.4Gbps in a V4 LX25
• power consumption: below 2W

More qualitative:

• programmability & flexibility of the end solution
  high abstraction level plus FPGA flexibility
• maintainability
  high abstraction level
  hiding of implementation specifics  } offer maintainability
• development time
  only 6-8 weeks for prototype & simulation
# FPGA versus NPU
## DSLAM implementation

<table>
<thead>
<tr>
<th>Device</th>
<th>Datapath Type</th>
<th>Throughput (pps)</th>
<th>Power (W)</th>
<th>Cost (~$)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>V4 LX25</strong></td>
<td>32-bit</td>
<td>12.5m</td>
<td>0.9</td>
<td>~35</td>
</tr>
<tr>
<td></td>
<td>128-bit</td>
<td>50m</td>
<td>1.1</td>
<td>~35</td>
</tr>
<tr>
<td><strong>Low end NPU</strong></td>
<td>Agere APP300</td>
<td>4m</td>
<td>5</td>
<td>~35</td>
</tr>
<tr>
<td></td>
<td>Intel IXP2350</td>
<td>2.5m</td>
<td>11</td>
<td>~125</td>
</tr>
<tr>
<td><strong>DSLAM specialized NPU</strong></td>
<td>Infineon Convergate-C</td>
<td>0.5m</td>
<td>1.5</td>
<td>~35</td>
</tr>
<tr>
<td></td>
<td>Wintegra 717</td>
<td>0.2m</td>
<td>2.7</td>
<td>~50</td>
</tr>
<tr>
<td><strong>High end NPU</strong></td>
<td>Intel IXP2800</td>
<td>30m</td>
<td>25</td>
<td>~400</td>
</tr>
<tr>
<td></td>
<td>Xelerated X11-S200</td>
<td>30m</td>
<td>11</td>
<td>~295</td>
</tr>
</tbody>
</table>
Summary

• Domain specialist can get efficient access to FPGAs without being a hardware expert

• Compile/synthesize for a problem-specific optimized combination of logic, embedded processors, and memory

• The 80% routing in the silicon is the secret sauce to outperform fixed processing solutions

• FPGA opportunity requires new thinking and new tools
Xilinx System Workbench for Students

Virtex-II Pro XC2VP30 FPGA
- 30,816 Logic Cells
- 2448 Kbits of BRAM
- 2 PowerPC 405 processors
- 8 MultiGigabit Serial Tranceivers
- 8 Digital Clock Management blocks
- 136 18x18 multipliers

USB port for FPGA Configuration using standard USB cable

Compact Flash card interface for individual project back-up or IBM Miicrodrives with upto 8Gbit capacity

High-speed Gigabit serial I/O
- Serial ATA connectors

Expandable memory up to 2 Gigabytes
- DDR SDRAM DIMM Slot

Support for supply current monitoring

Self-test / configuration Flash memory

I/O under and over voltage protection
Block Diagram

- High-speed and low-speed I/O expansion connectors
- Additional I/O via four user-supplied 60-pin headers
- Three Serial ATA connectors
- One 3.125 Gbps port via 4 user-supplied SMA connectors
- 10/100 Ethernet PHY
- RS232
- PS-2 (x2)
- SVGA
- Buttons (5), LEDs (4), switches (4)
- DDR SDRAM DIMM
- AC97 Audio CODEC & Stereo AMP
- Compact Flash Configuration
- USB Configuration
- Platform Flash Configuration
- External Power
- Internal Power Supplies 3.3V, 2.5V, and 1.5V
- 100 MHz system clock
- 75 MHz SATA clock
- 2 user supplied clocks
www.xilinx.com/univ

• Online donation forms for Xilinx SW products

• Purchase university boards

• Donations from Xilinx
  – See XUP donation request form at www.xilinx.com/univ

• Educational clip-art
Stanford NetFPGA

NetFPGA is a PCI Board

- Program in Verilog
- Industry-standard design flow
- Contains embedded CPUs

NetFPGA is a Programmable 4 x 1GE “switch” or any packet processor

http://yuba.stanford.edu/NetFPGA/
MIT Labkit

• http://www-mtl.mit.edu/Courses/6.111/labkit/
Berkeley BEE2

- [http://bee2.eecs.berkeley.edu/](http://bee2.eecs.berkeley.edu/)

![Image of Berkeley BEE2](image-url)