HW/SW Co-Design of FPGA-based SOC

→ Design Tools - Collaborative Engineering - event
→ Technische Universiteit Eindhoven (the Netherlands)
→ November 26, 2008

Ramses Valvekens – ramses@easics.be
Easics N.V., Leuven, Belgium

www.fhi.nl/designtools
www.easics.com
About Easics

Nowadays’ FPGA Design Flow

FPGA versus ASIC ⇒ FPGA SOC

{ Vision Towards Future FPGA Design Flow
Two Ways to Approximate that Today

Case Study: High-performance Sorting System

References

Questions & Answers
About Easics

Your System-on-Chip Design Services Partner since 1991

FPGA
ASIC

◮ Independent design services company
◮ System-on-Chip:
  ◦ FPGA
  ◦ Digital & Mixed-signal ASIC
◮ Hardware/Software co-design
◮ Future-proof design style
◮ Unique verification strategy
◮ Expertise:
  ◦ Wired & Wireless Connectivity
  ◦ Digital Signal Processing
  ◦ Multi-Processor
  ◦ Low Power
◮ Silicon Technologies: 0.35\(\mu\)m ... 45nm
◮ Based in Leuven, Belgium

© 2008 Easics N.V. – ramses@easics.be
FPGA Architecture

Field Programmable “Gate” Array (since 1984)

- **Logic**
  - Combinatorial
  - Sequential
    - Flip-flops
    - Static RAM

- **Interconnect**

- **I/O**

- **Configuration Storage**
  1. Volatile
  2. Non-volatile
  3. OTP

- **IP: Hard + Soft**
System-on-PCB containing FPGA as Glue Logic
FPGA History

- Historical use:
  - Glue logic → designed by PCB designers
  - ASIC prototyping → designed by ASIC designers

- EDA Tools:
  - Design Entry (Schematic- & HDL-based)
  - Simulation
  - Synthesis
  - Place & Route
  - Integration with PCB-design

- Tool suppliers:
  - FPGA vendors (almost for free)
    - own tools or bundled from EDA tool vendors
  - Large EDA tool vendors (ASIC history)

- "FPGA Abstraction" (Synplicity, Inc)
  - DSM effects tackled by FPGA vendors + EDA tool vendors
  - Only think about the Application: logic design + timing
**Comparing FPGA & ASIC Technology**

**Measuring the Gap**: FPGA & ASIC in same process (90nm) SRAM/LUT-based FPGA vs. Standard Cell ASIC

<table>
<thead>
<tr>
<th>FPGA/ASIC Ratio</th>
<th>Logic only</th>
<th>Logic + Memory + DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Area</strong></td>
<td>34</td>
<td>18</td>
</tr>
<tr>
<td>Critical-Path Delay</td>
<td>3.4 ... 4.6</td>
<td>3.0 ... 4.1</td>
</tr>
<tr>
<td>Dynamic Power Consumption</td>
<td>14</td>
<td>7</td>
</tr>
</tbody>
</table>

Source: “Measuring the Gap Between FPGAs and ASICs”
IEEE Transactions on Computer-Aided Design of Integrated Circuits
Vol. 26, No. 2, February 2007
Ian Kuon & Jonathan Rose
Electrical and Computer Engineering Department
University of Toronto, Canada
Comparing Apples to Oranges?

ITRS Roadmap (Moore’s Law):
New process generation every 2 years
Silicon area per transistor = 1/2 of previous generation
..., 0.18µm, 0.13µm, 90nm, 65nm, 45nm, 32nm, 22nm, ...

Closing the Gap: FPGA in 65nm vs. ASIC in 0.18µm
SRAM/LUT-based FPGA vs. Standard Cell ASIC

3 generations difference: area = 1/8 (and cost \(\propto\) area !)

<table>
<thead>
<tr>
<th>FPGA/ASIC Ratio</th>
<th>Logic only</th>
<th>Logic + Memory + DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Calculated numbers, to be compensated for architecture, routing, ...

"Moore’s Law is still valid but it does not work for most sectors of the industry today"
Wim Roelandts, chairman of Xilinx in EE Times, April 17, 2008:
www.eetimes.com/showArticle.jhtml?articleID=207400596
FPGA Cost per function drops over time & process generations

This information is quoted from Xilinx, Inc.
cost per function

- 90 nm ASIC
- 130 nm ASIC
- 180 nm ASIC
- 180 nm FPGA
- 130 nm FPGA
- 90 nm FPGA
- 65 nm FPGA

Volume

$
FPGA SOC: Ever More Demanding Requirements

Requirements for new hardware & software:

- high performance (processing power): e.g., DSP: *, +
- parallel processing
- tight integration (small physical size)
- flexible (reprogrammable: changing requirements)
- short time-to-market (in spite of ”in flux” requirements)
- low power consumption (heat dissipation)
- low total cost
- future-proof (reuse)
- less dependent on component suppliers (component obsoleteness problem)
FPGA SOC: Definition

- **SOC** = integrated circuit with the ambition to fit your entire application in a single chip:
  - digital logic
  - microprocessors
  - DSP
  - memories
  - analog circuitry
  - ...

- **FPGA SOC** = SOC implemented in FPGA
  - fully flexible
  - OK for small series
  - migration path to ASIC
Modern System-on-Chip Design Challenges

FPGA SOC

Additional abstraction layer: inside the FPGA SOC

Besides application-specific digital logic, the FPGA SOC contains Processors (softcore, diffused), DSP blocks, Block-RAM, PLLs, ...
Future Vision on FPGA Tools

- There are much more SW engineers than HW engineers.
- Design the entire application in SW engineer’s language.
  - using his/her native language (e.g., C++, Matlab, ...)
  - using his/her native developments tools (compiler, debugger, ...)
  - ideally without constraints on what language constructs can be used or not
  - the result can be reused in non-HW environments: verification, software development, ...
- Followed by high-level behavioral synthesis / compilation (performance parameters can be set in the tool)
- Higher level of abstraction ⇒ higher profitability
  - faster time-to-market (higher productivity)
  - better design quality (first time right)
  - platform-independent design (future-proof)
Today’s Approximation #1

- Manual design of application-platform using VHDL + processors by HW engineers

- Delivery of Board Support Package (BSP) incl. API

- SW engineers program the application on ”their” SOC using their regular language & tools (”no VHDL in sight”)

© 2008 Easics N.V. – ramses@easics.be
Today’s Approximation #1 - Software View: BSP

FPGA SOC

- microprocessor
- JTAG

memory map
- mailbox
- custom hardware
- timer
- sdram

controller

custom hardware

Board Support Package
- linker script
- memory map description
- optional: device drivers
- optional: operating system

software debugger

Modern System-on-Chip Design Challenges
Today’s Approximation #2

- Engineers program in their preferred language (with some limitations)

- This program is directly supplied to a tool (no custom-hardware layer is designed)

- Practical limitations:
  - selection of a vendor / tool
  - input language = vendor-specific sub- and/or super-set of an existing language, or even a new language
  - stringent coding guidelines → hardware
  - domain-specific?
Case Study: BEST

”Belgian Electronic Sorting Technology”
High-Tech Optical Sorter Manufacturer
for Food and Non-Food Industry
R&D centers in Leuven & Eindhoven
Case Study: Sorting principle

- Optical techniques:
  - camera:
    - visible spectrum
    - ultra-violet
    - infra-red
  - laser
  - LED
  - X-ray

- Sorting criteria: hardware & software

- Eject mechanism: pneumatic valves
Case Study: a Complete Sorter

© 2008 Easics N.V. – ramses@easics.be

Modern System-on-Chip Design Challenges
Case Study: Hyperion = FPGA SOC + SDRAM + SRAM
Case Study: Hyperion plugged into existing Motherboard
Case Study: Hyperion Design Methodology

- Mix of logic & multiple processors
- Development both on Linux & Windows
- Version control: git
- Makefiles
- Hardware/Software co-development
- Platform Independent
- Regression Testing
Case Study: Hyperion Design Languages & Tools

- **Hardware inside the FPGA SOC**
  - RTL language: VHDL
  - Modeling & verification language: C++
  - Hardware/Software register map generation tool: Easics - VCI Compiler
  - RTL simulator: Mentor Graphics - Modelsim SE
  - Hardware/Software co-simulation: VHDL/C++ using Modelsim’s FLI (foreign language interface)
  - Synthesis tool: Synplicity - Synplify Pro
  - Place & Route tool: Xilinx - ISE

- **Software running on the FPGA SOC**
  - language: C
  - develop environment: GNU - gcc
  - debug environment: Xilinx - XMD (via JTAG) → gdb

- **PCB**
  - schematic & layout tool: Mentor Graphics incl. signal integrity
Case Study: Why C++?

- Verification code outlives RTL design phase: reuse everywhere
  - pre-VHDL: modeling, architecture exploration, design budgeting
  - post-VHDL: driver development, software testing, lab testing, customer demo, inside final product

- Platform-independent:
  - OS: Linux, Windows, MacOSX, Solaris, uCLinux, eCos, Windows XP embedded, ...
  - uP: Workstation (Intel-PC), embedded uP (ARM, PowerPC, Nios, Microblaze, ...), embedded PC, mix of previous

- Mature, with good debug capabilities, strongly-typed
- Nice: object-oriented, external libraries, editor support, IP friendly, fast, low memory footprint
- Limit to C, where needed
Case Study: Hyperion = Flexible + Scalable

1. Modify the Software (running on the FPGA SOC) 
   ”no VHDL or modelsim in sight!”

2. Add a number of (softcore) Processors (in the FPGA SOC), 
or tuning: units (MMU, cache, ...) & hardware instructions

3. Changes in Digital Logic (in the FPGA SOC) 
   - small hardware changes 
   - new hardware accelerator blocks 
   - even completely new architecture possible

4. Replace the FPGA SOC component with a larger one, 
   that is footprint compatible (on the same PCB)

5. Use extension connector to plug in additional PCB 
   (idem for Hyperion plugged into motherboard)

Bonus: reuse parts of the design (VHDL RTL, C++, C) 
in other projects (no component obsolescence)
Some related articles in Bits&Chips (©Techwatch B.V.):

▶ http://bits-chips.nl/nieuws/interviews/bekijk/artikel/easics-ontwerpt-hardware-met-softwaretechnieken.html

▶ http://bits-chips.nl/nieuws/bekijk/artikel/talenknobbel.html

▶ http://bits-chips.nl/nieuws/bekijk/artikel/een-andere-tak-van-sport.html


Thank You – Any Questions?

- based in Leuven, Belgium
- in the Arenberg Science Park
- www.easics.com
- business@easics.com
- tel +32 16 395 611