Experiences & Lessons Learned from A Complex SoC Design

Xianfeng Li (李险峰)

Microprocessor Research and Development Center (MPRC), Peking University
Outline

• Background Information

• Experiences & Lessons
SuperK – SoC for Single-Chip Computers
The Progress

• First tape-out projected: December 2007

• Now, it is ......August 2008, we are still in the IC back-end stage...

Reasons behind the delay?
Challenges - Design Complexity

- Over 30 IP modules, each with 8 milestones in 3 macro-stages
Challenges – Backend IC Issues

• **High-speed devices** (DDR2, bus)
  – Routing congestion
  – Timing convergence
  – Signal integrity

• **Power grid design**
  – The needs of multi-power domains
  – Dynamic circuit function variation
  – IR-drop problem

• **Clock tree synthesis**
  – Multiple clock domains
  – Timing and power optimization on non-critical paths

• **Convergence of signal integrity**
  – Multiple corners (worst-case, best-case, typical-case,...)
  – No EDA tools can completely automate the procedure, manual tuning resulting in convergence in one corner may destroy SI convergence in another corner.
Challenges – Limited Resources

- Human resource: roughly 100 people involved, but ...

![Pie chart showing the distribution of human resources by category: Faculty, PhD, "Experienced" MSc, Novice MSc, and Graduating MSc. Each category is represented by a different color and percentage. Faculty: 10%, PhD: 15%, "Experienced" MSc: 25%, Novice MSc: 25%, Graduating MSc: 25%.]
Experiences and Lessons

• System-level

• Module-level

• Platform

• Management
Overlook on Tech Trends

- Two concurrent projects

<table>
<thead>
<tr>
<th></th>
<th>SK-M</th>
<th>SK-F</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRAM</td>
<td>DDR-I</td>
<td>DDR-II</td>
</tr>
<tr>
<td>Secondary Storage</td>
<td>NAND</td>
<td>NAND + SATA</td>
</tr>
<tr>
<td>Graphics/Display</td>
<td>Third-party (SiS-6326)</td>
<td>UniGFX (Integrated)</td>
</tr>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

SK-M cancelled in Nov 2007
Underestimation on Design Complexity

• Too optimistic plans and deadlines

• Insufficiently prepared human resources

• Psychological effects (depression, confusion, lost of confidence...)

• Making realistic estimation is very important
Insufficient Considerations on Non-Functional Metrics

• Non-functional metrics
  – Performance
  – Power
  – Temperature
  – Area
  – Pin count

• Non-functional metrics should be explicitly provided together with functional specification

• Early evaluation of non-functional metrics is still a challenge
Efforts on Non-strategic IP Modules

- **PCI bridge IP**
  - Industry standard
  - Does not differentiate your product from others
  - Very complicated protocol, difficult to verify

- **UniGFX (graphics and multimedia)**
  - Important for system performance, power, cost, ...
  - A strategic IP

- **Lesson:**
  - IP-reuse improves productivity
  - Only design in-house IPs with strategic importance
Experiences and Lessons

- System-level
- Module-level
- Platform
- Management
Late-stage Functionality Changes

• **Examples**
  – The addition of Gbps UMAL
  – UniGFX 2D zooming functionality

• **Problems**
  – An expensive iteration of an outer loop
  – The original designer may have left the lab, or assigned another task
Problem on IP Design Flow

- Current flow: goes directly from the most abstract level to the most detailed level
  - Cumbersome for evaluating non-functional metrics at both levels
  - Validation difficulty: mixture of design & implementation errors
Problem on IP Design Flow

• An alternative flow: multiple refining processes
  – Evaluating non-functional metrics are easier with better speed/accuracy trade-off
  – Better support for design space exploration
    • Design problems can be discovered at an earlier stage
    • Changes are easier compared to RTL
  – Facilitating validation:
    • Separation of design & implementation errors
    • Co-verification: higher-level models can serve as reference model for more detailed models
  – But it might be a tedious process for the designer working on multiple models at different abstraction levels
Insufficient Communication between System Architect and Module Designer

• Example: MME
  – System architect and module designer had different clocking methods in mind
    • System architect: bus clock division
    • Module designer: independent clocking
  – The mismatch was found until MME sign-off!
  – This is also a problem of incomplete spec & doc

• Lessons
  – All aspects affecting design & implementation should be documented & communicated
  – Module interface should be carefully considered
Misconceptions on IP-Reuse

• Commercial IPs are bug-free
  – Wrong! We sometimes found bugs in commercial IPs, especially for communication-intensive IPs.

• Commercial IPs are plug-and-play
  – Wrong! Dealing with mismatch on performance, incompatibility of interface, unworkable driver and the needs of customization for commercial IPs is a routine task in our project. Delays of verification and prototyping facilities by vendors are not uncommon (VIP, FPGA card, ...)

http://mprc.pku.edu.cn
RTL Coding Is Not Software Programming

- Hardware design is becoming more and more like software programming (RTL coding using HDL languages)

- But it will never be software programming
  - Sequentiality vs Concurrency
  - Functional-only vs Multi-constraints

- Common problems for RTL coding novices
  - Abuse of registers (increases layout & routing difficulties)
  - Triple-nested “case statements”
  - No concept and care for potential critical paths

- Example
  - In the initial design of MME, RTL coder caring only about functional correctness failed to meet planned clock rate
Experiences and Lessons

- System-level
- Module-level
- Platform
- Management
Prototyping Platform

• The entire design cannot be mapped onto a single FPGA chip

• The dual-chip solution
  – Not transparent, needs changes to the design
  – Uncertainty on the source of problems across chips (from the target design or from the host platform?)
Experiences and Lessons

• System-level

• Module-level

• Platform

• Management
SoC design is a multi-stage process, so should we assemble a set of teams working in a pipeline manner?

The answer: **YES** and **NO**

- **YES**: decomposing a complex project into manageable smaller sub-tasks is a must. And each team takes the primary responsibility for a specific task.
- **NO**: drawing clear lines between teams are often harmful (See following sides...)
Problems with Team Barriers (1)

- **HW/SW team barrier**
  - IP designers write specification w/o involvement of SW people
  - Device driver developers may find unnecessary complications introduced for driver development

- **Breaking the barrier**
  - Let HW & SW people work together on the specification
Problems with Team Barriers (2)

- Arch/IC team barrier
  - RTL developers complete the design and “sign-off” to IC engineers
  - IC engineers find the RTL code with poor consideration for timing, layout constraint...

- Breaking the barrier
  - RTL developers are not IC experts
  - During RTL coding, IC engineers are idle anyway
  - Let IC engineers serve as helpers by working with RTL developer before the “sign-off”
  - Can result in a better physical constraint-aware design
Problems with Team Barriers (3)

- Another Arch/IC team barrier
  - IC engineers take over the design from arch people after sign-off, and work on their own for layout, routing, CTS, ...
  - But backend IC engineers on their own do not understand the RTL design very well, and design-aware optimization is difficult

- Breaking the barrier
  - RTL developer might be idle after sign-off
  - Let them serve as a helper for backend IC – Let the “sign-off” mean the exchanging of roles
  - Example: memory layout
Management Problems in A University Lab

• In a University Lab
  – Only a small number of faculty members and students
  – Worse, every year, novices are in (new students), Veterans are out (graduating students)

• Solution
  – It’s like a mountain climbing process, rest on the plateau, not on the slope!
Reaping

- A chip with 28 million transistors in flip-chip package
- A complex SoC platform
  - With almost any popular functionalities integrated on-chip
  - Linux 2.6 Kernel ported, toolchain matured
  - Evolving from IP reuse to platform reuse
- A project documentation flow, along with rich documentation for references and reflections
- Lessons learned from practice