A modular design flow for very large design space explorations

Thorsten Jungeblut¹, Sven Lütkemeier¹, Gregor Sievers¹, Mario Porrmann¹, Ulrich Rückert²

¹ System and Circuit Technology
University of Paderborn, Germany
² Cognitive Interaction Technology - Center of Excellence, Bielefeld University, Germany
Motivation

In Sub-90nm technologies:

• Wire delays more and more important
• Design flow up to Place & Route
• Automatic tool flow for design space exploration necessary

entity cpu is port (...);
...
end cpu;
Design Space Exploration
Tool Flow

Goal: Highly automated design flow
Environment Modules Project

- Hierarchical approach to adapt users environment by loading and unloading modules, e.g.:
  ```
  module load cadence/soc/8.1
  module load designkit/65nm/v3.1
  ```
- Easy changing of applications version definition of default versions
- Definition of dependencies
- Modules for target technologies define tool specific parameters for automated design flow
  - structure size
  - min/typ/max corners
  - metal layers, filler cells, ...
- Modules for EDA tools define reference scripts for automated design flow
  - Code quality checks, simulation, synthesis, P&R, simulation, power analysis, verification
- Environment of our group:
  - ~200 modules
  - ~50 commercial tools
  - 16 Systemumgebungen (Admin-Tools, Vorlesungen, …)
  - 8 design kits, >50 characterizations
Hardware Design Flow (Overview)

- Automated
- Concurrent execution with load balancing
- Reference directory structure
Synthesis
sct-synth, sct-synth-gbt

- Cadence® RTL Compiler
- PLE (Physical Layout Estimation)
- CPF (Common Power Format)
- GBT=Get Best Timing – Iterative algorithm to derive $f_{\text{max}}$

Source files (file/$\text{env}$)
Generics($\text{env}$)

user constraints
(global/design specific)
DEF (for PLE flow)

user constraints
(global/design specific)

Timing met?

Compile (incr.)
User script
(e.g. synthesize to placed)
user reports
(global/design specific)

Report

Save Design
**Synthesis**

*sct-synth, sct-synth-gbt*

- Cadence® RTL Compiler
- PLE (Physical Layout Estimation)
- CPF (Common Power Format)

- GBT=Get Best Timing – Iterative algorithm to derive $f_{max}$
**Place & Route**

sct-par

- **Floorplan flow (IP Block Design)**
  - Place hard macros.
  - Add well tap cells.
  - Global net connects.
  - Power routing.
  - Placement guides.

- **Floorplanless flow (Complete ASIC)**
  - Constraints from designkit module, e.g.:
    - Power rings.
    - Power stripes.
    - Metal layers.
    - Global net connects.

- **Netlist, Constraints**
  - Optimization Effort

- **Check Designkit**

- **Import Netlist**

- **Floorplan exists?**

- **Create Floorplan**

- **Place Design (STD cells, IOs, hard macros, ...)**

- **Report Timing**

- **Optimize**

- **Clock Tree Synthesis**

- **Optimize**

- **Report Timing**

- **Modify Floorplan**

- **Load Floorplan**

**= User defined scripts**

**Design entry points**
Place & Route

sct-par
Power Analysis

sct-record_sa/sct-power/sct-irdrop

- Record Switching Activity
  - RTL/Post-synthesis/Post P&R
  - SAIF/VCD

- Calculate Power Consumption
  - Encounter Power System

- IR drop (Internal Voltage Drop)
- EM (Electromigration)
Design finishing

sct-finish

- sct-finish
  - Via fill
  - Metal fill
  - gdsII streamout
  - Checks for DRC, process antenna, connectivity, …
• Simple but flexible load balancing tool
• Custom extendable Host/CPU-list

```
tj@vitamalz[~]> sct-lb.pl $PWD encounter
Load on schweppes (4 CPUs): 1.528 (per CPU: 0.382)
Load on vitamalz (4 CPUs): 1.304 (per CPU: 0.326)
Load on bionade (4 CPUs): 2.1 (per CPU: 0.525)
Load on powerade (2 CPUs): 0.932 (per CPU: 0.466)
Load on gatorade (2 CPUs): 0.412 (per CPU: 0.206)
Load on hohesc (2 CPUs): 1.644 (per CPU: 0.822)
Load on africola (2 CPUs): 0.122 (per CPU: 0.061)
Load on redbull (2 CPUs): 1.166 (per CPU: 0.583)

Choosing host: africola
Directory: /homes/lift/tj
Command: encounter

Starting at: Do 18. Mär 10:52:45 MET 2010
Output:
=================================================================================
Starting Encounter...
```

Host list: vitamalz schweppes bionade powerade gatorade hohesc africola redbull
CPU list: 4 4 4 2 2 2 2

Load on vitamalz (4 CPUs): 1.176 (per CPU: 0.294) Users: 75.5 huebener 37.9 tj_local 4.0 root 0.2 tj
Load on schweppes (4 CPUs): 1.544 (per CPU: 0.386) Users: 28.8 blesken 16.8 dklimeck 101 tj 2.7 tj
Load on bionade (4 CPUs): 2.124 (per CPU: 0.531) Users: 99.2 gsievers 98.9 gsievers 3.8 tj 3.3 tj
Load on powerade (2 CPUs): 0.756 (per CPU: 0.378) Users: 71.5 huebener 2.0 root
Load on gatorade (2 CPUs): 1.376 (per CPU: 0.688) Users: 82.3 svenl 28.7 gsievers
Load on hohesc (2 CPUs): 1.802 (per CPU: 0.901) Users: 73.9 tj 53.2 huebener
Load on africola (2 CPUs): 0.078 (per CPU: 0.039) Users: 5.8 svenl 2.0 root
Load on redbull (2 CPUs): 1.244 (per CPU: 0.622) Users: 99.2 matkam 8.7 matkam
Simulation

cst-sim, cst-sim_all

• Validation by Simulation approach [1]

• Automated Validation of multiple design configurations, e.g.:
  – I/D-Cache enabled/disabled
  – Data cache write miss mode enabled/disabled
  – FPGA/ASIC design
  – RTL vs. synthesis vs. layout
  – Large test suites (>50 test cases)

• Lots of simulations!

• Runtime between 30s and >12h per simulation step

Simulation
Multi Domain Validation

Instruction Set Simulator

Machine Model
- Functional Units
- Register Banks
- Virtual Resource

Instruction $i$
- Cycle
- Resources
- Execution Time: 3
- Latency: 2

UPSLA Reference Specification

L1-Cache
Instruction Decode
Register Read
L1-D-Cache
Register Write

RTL Description

Validation

ASCI Realization
Modbusim Foreign Language Interface (FLI)

Tracediff

Table:
- FU
- Address
- Machine Instruction
- Disassembly
- State
- Difference

<table>
<thead>
<tr>
<th>FU</th>
<th>Address</th>
<th>Machine Instruction</th>
<th>Disassembly</th>
<th>State</th>
<th>Difference</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>[0x0000241]</td>
<td>0x70000000</td>
<td>ncp</td>
<td>r3=0x0 pc=0x27c[A]</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>[0x0000027c]</td>
<td>0x703c0302</td>
<td>mov r3, 0x2</td>
<td>x0=0x3 r0=0x10002f4 r3=0x2[A] r4=0x0</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>[0x00000280]</td>
<td>0x70040020</td>
<td>sub r0, r0, 0x20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>[0x00000284]</td>
<td>0x706000ff</td>
<td>mcr r4, 0xff</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>[0x00000288]</td>
<td>0x71046002</td>
<td>neq c0, r3, 0x2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

A = 0x0000241, r0 = 0x100024d[A], r4 = 0xff[B]
Case Study: The CoreVA architecture
Modular Design
Case Study: The CoreVA architecture
Instruction set, features

- Harvard architecture (LD/ST-Architecture)
- Orthogonal instruction set (very regular)
- Parallel execution due to VLIW, but stop bit
  → compact code at sequential parts
- 1 cycle instructions with 1 cycle latency (except BR, MAC, LDW (latency 2))
- Conditional execution (Multiple condition registers)

41 base instructions
15 SIMD instructions

Conditional execution:
c7 sub eq, c0, r0, r1
 c0 add r2, r2, 1
 c7 mov r3, 0

<table>
<thead>
<tr>
<th>st</th>
<th>cond</th>
<th>Opcode</th>
<th>Rn</th>
<th>Rd</th>
<th>Rm or immediate</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>27</td>
<td>17</td>
<td>12</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

32 bit RISC instructions
Case Study: The CoreVA architecture
ASIC realization

- 4-issue VLIW processor, 2x MLA,DIV
- 1-Port I-Cache (16kByte, 128 Bit),
- 2-Port D-Cache (16kByte, 32 Bit)
- 65nm ST Microelectronics, Low Power (Thick Oxide), 1.2V MixedVT, 1.8V I/Os (configurable pullups)
- Hardware extensions (incl. ECC)

<table>
<thead>
<tr>
<th>Frequency</th>
<th>300 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (112kB SRAM)</td>
<td>6.5 mm²</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>0.1 W</td>
</tr>
</tbody>
</table>

1.2 GOP/s in scalar mode
2.4 GOP/s in SIMD mode
Case Study: The CoreVA architecture

ASIC realization

- 4-issue VLIW processor
- 1-Port I-Cache (16kByte)
- 2-Port D-Cache (16kByte)
- 65nm ST Microelectronics
  1.2V MixedVT, 1.8V I/O
- Hardware extensions (incl. ECC)

<table>
<thead>
<tr>
<th>Frequency</th>
<th>300 MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area (112kB SRAM)</td>
<td>6.5 mm²</td>
</tr>
<tr>
<td>Power Consumption</td>
<td>0.1 W</td>
</tr>
</tbody>
</table>

1.2 GOP/s in scalar mode
2.4 GOP/s in SIMD mode
Questions?

Design-space exploration

CoreVA VLIW architecture
Thank you for your attention!

Heinz Nixdorf Institute
University of Paderborn
System and Circuit Technology
Dipl.-Ing. Thorsten Jungeblut
Fürstenallee 11
33102 Paderborn

Tel.: 05251/60 63 39
Fax.: 05251/60 63 51
Email: tj@hni.upb.de
http://www.hni.upb.de/sct