Low Power Algorithm Implementation And Verification Using C++

Dan Gardner
Design Creation and Synthesis Division
Mentor Graphics
Improved Power and Cost Through Improved System Architecture

- Physical Design
- Gate Level
- RTL
- Behavioral Level
- Algorithmic
- System Level

Ability to Impact Design

Ability to Quantify
Traditional Flow vs. Catapult Flow

- Avoid bugs
- Lower costs
- More flexible
Numerical Refinement & Closed Loop Verification

- Verification/Validation depends on application and granularity of algorithm
  - Bit Error Rate
  - Mean Square Error
  - No overflows requirement
- Floating-point may be optional step
  - Code fixed-point from the start
- Simulation speed essential for validation/verification
- Use exact bit-widths required to meet specification and save power/area

Matlab/C++ Algorithm using floating-point

Refine/Explore Precision

Measure/Verify

C++ Algorithm using bit accurate integer/fixed-point

Catapult C

SCVerify

Equivalent bit accurate RTL

C++ Algorithm using bit accurate integer/fixed-point

Catapult C

SCVerify

Matlab/C++ Algorithm using floating-point

C++ Algorithm using bit accurate integer/fixed-point

Catapult C

SCVerify

2008 MAPLD Session D - Design and Verification Tools and Methodologies
Micro-Architecture Optimization

```c
void MAC(int8 taps[4], int8 coefs[4], int10 *out)
{
    int18 accum = 0;
    for (int i=0; i<4; i++)
        accum += taps[i] * coefs[i];
    *out = accum >> 8;
}
```
int multaddadd (short A[4], short B1[4])
{
    return (A[0]*B[0]) + (A[1]*B[1]) + \\
}

Architecture-neutral description

Architectural Constraints 1
200MHz
Slow speed grade

RTL 1

Architectural Constraints 2
100MHz
Fast speed grade

RTL 2

Frequency is a parameter
Multi-clock Design

- Blocks with lower data rates run with slower clock
  - Reduction in switching power
  - Reduction in static power by decreasing block area

Technology Constraints

Architectural Constraints

Multiple clocks specified with unique parameters

Clk1 = 200 MHz  Decimation by 2

Clk2 = 100 MHz  Decimation by 4

Clk3 = 25 MHz  Decimation by 8

Each hierarchical block can be assigned to any clock domain
Closed-loop Power Analysis and Optimization

- Power consumption data annotated into Catapult using leading power analysis tools
- Micro-architecture optimizations used to balance power/area/performance
- Average 30% power savings using this flow

<table>
<thead>
<tr>
<th>Solution</th>
<th>Leakage Power</th>
<th>Internal Power</th>
<th>Switching Power</th>
<th>Total Est Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIR_FILTER (extract)</td>
<td>8.78uW</td>
<td>1.05mW</td>
<td>1.69mW</td>
<td>2.74mW</td>
</tr>
<tr>
<td>UNROLLED_2 (extract)</td>
<td>10.9uW</td>
<td>1.23mW</td>
<td>2.64mW</td>
<td>3.68mW</td>
</tr>
<tr>
<td>UNROLLED_4 (extract)</td>
<td>16.7uW</td>
<td>1.71mW</td>
<td>4.83mW</td>
<td>6.56mW</td>
</tr>
<tr>
<td>UNROLLED_8 (extract)</td>
<td>30.3uW</td>
<td>1.76mW</td>
<td>4.24mW</td>
<td>6.03mW</td>
</tr>
<tr>
<td>PIPELINED (extract)</td>
<td>20.4uW</td>
<td>1.90mW</td>
<td>4.55mW</td>
<td>6.46mW</td>
</tr>
<tr>
<td>rtl-vhdl-msim</td>
<td>20.8uW</td>
<td>2.24mW</td>
<td>7.37mW</td>
<td>9.64mW</td>
</tr>
<tr>
<td>gate-vhdl-msim</td>
<td>20.4uW</td>
<td>1.90mW</td>
<td>4.55mW</td>
<td>6.46mW</td>
</tr>
</tbody>
</table>

Low power / Small area

Higher power / Larger area
ESL Flow

Catapult C
High Level Synthesis

Vista Model Builder
Modeling & Extraction

Visual Elite
Design Assembly

Vista
System Analysis & Validation

Questa
Verification

TLM Reference Platform

RTL Reference Platform

Reused (Golden) RTL
VHDL, Verilog, PSL, OVM, SystemVerilog

Algorithms & System Level IP
C/C++
SystemC

Software

RTL

Testbench

Redesigned (Golden) RTL

Specification

2008 MAPLD Session D - Design and Verification Tools and Methodologies
Special Challenges with Mil/Aero DSP

“High Cost of Failure”

- **Design reuse**
  - Very long product life cycles
  - Legacy design difficult to retarget
  - Switching between FPGA vendors is very expensive

- **Design quality**
  - Achieving optimal numerical precision is difficult
  - Finding optimal hardware architecture is time consuming
  - Designs are typically overbuilt to guardband design goals

- **Functional correctness**
  - Mandatory for mission critical hardware
  - Up to 60% of design errors come from disconnect between functional spec and RTL implementation
  - RTL is too slow for system verification

- **Time to Market**
  - Tight milestones in government projects
  - Late changing requirements
Value of Algorithmic Synthesis

Functional
Function and Operate Well
Avoid Bugs

Optimized
Optimized Resources
Lower Power
Reduce Cost

Flexible
Quickly Deliver Derivative Designs
First to Market
Optimized Design Architecture

- RTL confines your implementation to few solutions in close proximity
  - Architectural details embedded in the source

- Structural languages offer limited trade-offs
  - Limited reuse
  - Complicates coding style
  - Prevents bit-accurate modeling & numerical refinement

- Restricted ANSI C
  - Limits reuse
  - Complicates coding style
  - Prevents bit-accurate modeling & numerical refinement

- Pure ANSI C++ allows exhaustive exploration of design space
  - Extremely compact
  - Object oriented hardware reuse
  - Optimization through interactive constraints
  - Optimize serial vs. parallel
  - Optimize sequential vs. pipelined

![Diagram showing optimization scopes and architectural scenarios]
Interface Optimization With Interface Synthesis

- C++ source and testbench independent of HW interface
- Designers focus on architecture and function
- Micro-architecture tuned to the interface
  - Memories
  - Busses
  - Streaming data
- Adjust bit-widths to balance performance and power

```c
void func ( int A[5][16],
           char B[16],
           bool mode  ) {
  ...
}
```

Patent-pending
Memory Architecture in C++

- Power, performance and area for many algorithms are highly dependent on memory architecture
- C++ makes various memory architectures easy to explore
  - For example, something as simple as a FIR filter can take numerous “forms”

**Shift register**

```
input
\[ D \quad D \quad D \quad D \quad D \]
\[ \times \quad + \quad D \]
output
```

**Circular Buffers**

```
Write
\[ M \quad M \quad M \quad M \quad M \]
Read
\[ \times \quad + \quad D \]
output
```

**Rotational shift**

```
input
\[ D \quad D \quad D \quad D \quad D \]
\[ \times \quad + \quad D \]
output
```
System Level Capabilities

Automated Streaming Interfaces Between Hierarchical Blocks

Optimized hardware creation using Catapult Synthesis
Starting from High Speed pure ANSI C++ Algorithmic Model
Catapult Verification Extension
SCVerify

- SystemC Transactors
- Original C++ testbench reused to verify the RTL
- Transactors convert function calls to pin-level signal activity
- Push button solution creates Makefiles and Simulation Scripts

Diagram:

- Original C++ Testbench
- Original C++ Algorithm
- Transactor
- RTL
- Transactor
- Comparator
- Golden results
- DUT results
Catapult & Mathworks Partnership

- Provides link between Catapult and MATLAB/Simulink
  - System Simulation
  - Numerical refinement
  - HW verification
- Closes the gap between algorithm design and implementation
- Focus on high-end FPGA and ASIC

Diagram showing the flow from Catapult to Simulink with steps such as Text Editor, C/C++ Function, High Level Synthesis, RTL, and ASIC Flow.