Floating-Point to Field-Tests:  
A Unified Development Environment for Algorithm Research

Jared Dulmage  
Dr. Michael P. Fitz

University of California, Los Angeles  
Annual Research Review  
1/13/2008
Traditional Hardware Development Process

Floating-pt sim

Fixed-pt sim

HW design sim

HW bench test

HW field test

Matlab
HLL (C,C++) compiler
Batch tools/scripting

Conversion tools/libraries
Fixed-point C,C++ model

HDL editor/simulator
Schematic capture
Signal visualizations

HDL batch tools/scripts
Test vector generator

Synthesis tools
Hardware probes

Test equipment

• 3+ programming languages
• 3+ tool chains
UnWiReD Hardware Development Tools

Floating-pt sim

Fixed-pt sim

HW design sim

HW sync cosim

HW async cosim

<table>
<thead>
<tr>
<th>Matlab</th>
<th>Simulink</th>
<th>System Generator</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Batch scripts</td>
<td>• Long-run, floating-point simulation</td>
<td>• Hardware design and simulation</td>
</tr>
<tr>
<td>• Parameter computation</td>
<td>• Source generation</td>
<td>• Automatic synthesis</td>
</tr>
<tr>
<td>• Post-processing</td>
<td>• Signal visualizations</td>
<td>• HW-in-the-loop interfaces for hw</td>
</tr>
<tr>
<td>• Off-line analysis</td>
<td>• On-line analysis</td>
<td>co-sim</td>
</tr>
<tr>
<td>• General environment and I/O</td>
<td>• Library of pre-built functions</td>
<td>• Library of pre-built hardware</td>
</tr>
</tbody>
</table>

• 2 languages, 1 integrated tool
Generic Environment for SW/HW Development

- Design may be arbitrarily partitioned between hardware and software

**Software**
- Programmable
- Advanced debug
- Access PC resources

**Synchronous interface**
- Non-real-time
- Single sample interface
- Limited performance

**Hardware**
- Relatively static
- Limited debug
- Access hardware resources

**Asynchronous interface**
- Real-time capable
- High performance
- Block data interface

- Real-time operation for access to external hardware (e.g. ADC/DAC)

- Fine-grained control simplifies debugging
- Straightforward implementation
Development Methodology

- Each subsystem generally evolves according to the diagramed stages
- A testbench is created in the first stage and used in each subsequent stage
- Subsystem integration follows the same progression
- The final collection of models provides a consistent and comprehensive development history
Fixed point, cycle-true simulation

- Implemented with System Generator blocks
- Determine bit widths and precisions
- Estimate performance degradation
- Debug hardware design using single-sample interfaces
- Determine latency properties of design
Fixed point synchronous co-simulation

- Hardware co-simulation block generated from fixed-point DUT model
- Ensure timing closure and correct synthesis
- Determine resource usage
- Faster, comprehensive performance testing
Fixed point asynchronous co-simulation

- Hardware co-simulation block generated from fixed-point DUT model using shared-memory interfaces
- Integrate with block data interface
- Run real-time/accelerated tests
Conclusions

• Unified software/hardware development environment improves efficiency
  – Fewer tools means shorter learning curve for developers
  – Abstracted block-diagram interface reduces required expertise
  – Consistent environment facilitates inter-stage coordination among engineering teams

• Hybrid software/hardware architecture facilitates rapid system development
  – Single, consistent testbench across all design iterations reduces duplicated effort
  – Combines flexibility of general purpose processing (GPP) and high performance of programmable hardware (FPGA)
Floating Point Simulation

- Implemented with Simulink blocks
- Debug algorithms, define data interfaces
- Measure performance baseline
DSRC Testbed Hardware Architecture

Host PC

PCI

FPGA board

ADC

DAC

RF front-end

RX I/Q

coax

RF

TX I/Q

UnWiReD Lab
UCLA Wireless Research and Development

Slide 11