Entwurf und Verifikation von System-on-Chips – Effizient mit Model-based Design

Dr. Werner Bachhuber
Application Engineering

Dr. Marc Segelken
Application Engineering
Model-Based Design Workflow

Modelling Algorithms: Ideal, Floating Point

Detailed Design: Fixed-Point Architectural System context

Implementation: C and RTL generation

ALGORITHM DESIGN
- Environment Models
- Digital Models
- Analog Models
- RF Models
- Timing and Control Logic
- Algorithms

ALGORITHM IMPLEMENTATION
- C/C++
- HDL
- MCU
- DSP
- FPGA
- ASIC
- Transistor

RESEARCH

REQUIREMENTS

INTEGRATION

ALGORITHM TEST & VERIFICATION

Validation of Requirements

Verification against Requirements

Verification against Detailed Design
Model-Based Design
Development Process

- REQUIREMENTS
- DESIGN
  - Environment Models
  - Physical Components
  - Algorithms
- Component Design
- Subsystem Design
- System-Level Specification
- Code Verification and Validation
  - Complete Integration & Test
  - System-Level Integration & Test
  - Subsystem Integration & Test
- Subsystem Implementation
- IMPLEMENTATION
  - MCU, DSP, FPGA, ASIC, PLC
  - C, C++, VHDL, Verilog, Structured Text
- Research
  - Data analysis
  - Algorithm Development
  - Data Modeling

User Acceptance Testing

Integration Testing

Integration & Test Complete
ZedBoard

Zynq SoC (XC7Z020)

Load motor

Mechanical coupler

FMC module: control board + low-voltage board

Motor under test (with encoder)
Model-Based Design
Multi-Domain Modeling

Methods for modeling systems in different domains

- Data Flow (Block diagram)
- Physical Modeling (Schematic)
- Modeling of Event-Driven Systems (State - Machines)
- Programing Language (Textual)
Field-Oriented Control of Velocity Hardware/Software Test Bench

Copyright 2015-2017 The MathWorks, Inc.
Conceptual Workflow Targeting SoCs
Model-Based Design
Component Refinement

- Refinement and Optimization of Components
- HW/SW Partitioning
- Possibly Fixed-Point Scaling
  - Includes Automatic Scaling Support
- System Simulation
- Open-Loop Testing
Hardware/Software Partitioning

Target to ARM

Target to Programmable Logic
C - Code Generation
Performance Profiling

- On desktop profiling
  - Tasks execution time on desktop machines

- On target profiling
  - Tasks execution time when running on target

- SIL/PIL – based profiling
  - Functions and tasks execution time when running in-the-loop type of simulations
HDL - Code Generation
Model And Code Traceability
SoC HW/SW Co-design Workflow

- **HW Design**
  - SW Interface Model
  - SW Build

- **IP Core Generation**
  - AXI Lite Accessible registers
  - Algorithm from MATLAB and Simulink
  - FPGA IP Core

- **SW I/O Driver Blocks**
  - SW Interface Model

- **Generate SW Interface Model**
  - Processor
  - AXI4-Lite Bus
  - AXI Lite Accessible registers
  - Algorithm from MATLAB and Simulink
  - FPGA IP Core

- **Embedded System Project**
  - External Ports

- **Simulink Model**
  - Embedded System Integration

- **FPGA IP Core**
  - Algorithm from MATLAB and Simulink
  - External Ports

- **AXI Lite**
  - Accessible registers

- **External Ports**
  - AXI4-Lite Bus
  - FPGA Bitstream
Model-Based Design

Verification and Validation

Requirements Traceability
- Navigation and Reports

Modeling Standards Checking
- Automatic Review

Testing and Test Automation
- Test Authoring
- Coverage Analysis
- Automatic Test Case Generation

Property Proving
- Quickly find incorrect behavior
- Prove correctness of behavior

Static Analysis on Model and C Code
- Identify defects
- Prove absence of runtime errors
- Compliance to coding standards

Compliance to DO-178C / DO-254
- Workflow and tool qualification
Simulink Requirements
Traceability down to Code
Simulink Check
Verify compliance with guidelines and standards

- Compliance checking for industry and security standards
  - DO-178, ISO 26262, IEC 61508, IEC 62304, MAAB, MISRA
  - Security: CERT C, CWE, ISO/IEC TS 17961

- Edit-time checking to fix violations earlier

- Metrics dashboard to assess design quality

- Refactoring of model clones and model
Measuring Model Coverage
Simulink Design Verifier – Main Components

Use formal methods to identify design errors

**Design Error Detection**
- **Uncover** hard to find dead logic and design flaws

**Test Generation**
- **Automate** test case generation to complete coverage

**Requirements Proving**
- **Prove** formally design meets requirements

**Model Slicer**
- **Simplify** models to isolate behavior
Test Case Generation – Results

Tests

Test#  Started execution  Ended execution

Summary

Model Hierarchy Complexity

<table>
<thead>
<tr>
<th>Decision</th>
<th>Condition</th>
<th>MCDC</th>
<th>Execution</th>
<th>Relational Boundary</th>
</tr>
</thead>
<tbody>
<tr>
<td>65</td>
<td>90%</td>
<td>98%</td>
<td>93%</td>
<td>100%</td>
</tr>
<tr>
<td>64</td>
<td>68%</td>
<td>98%</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

1. focZynqHDl_harness1
2. Size Type
3. Test Unit (copied from focZynqHdl)
4. FOC Velocity Encoder
5. ADC Count To Current
6. Encoder To Position And Velocity
7. Encoder Count To Rotor Position
8. Mid Two Pi Once
9. Rotor Position Delta
10. Detect Change
11. Wrap Neg Pi To Pi Once
Formal Verification – Property Proving

Proof Objective

Summary

1. Detection of current surge
   Detect if current exceeds limits. If either Phase A Current (current[0]) or Phase B Current are above the threshold for a given duration, generate an error. The error is reset when over current detection is disabled.

Model Item: Proof Objective
Property: Objective: T
Status: Valid
Model-Based Design Workflow for DO-254

- **Requirements Tools**
- **MathWorks Tools**
  - High Level Requirements
  - Conceptual Design
  - Detailed Design (RTL)
  - Detailed Design (Net-List)
- **EDA Partner Tools**
- **CoSim / FIL**
  - Verify
  - Trace
  - Verify
  - Verify
  - Verify
  - Verify
  - Verify

- **Requirements**
- **Tools**
- **MathWorks Tools**
- **EDA Partner Tools**
- **CoSim / FIL**
Cosimulation
Model Equivalence Checking

- During cosimulation the Simulink model and the generated HDL code are simulated simultaneously and their generated outputs are compared to show they are identical.
Cosimulation
HDL Code Coverage

Questa Coverage Report

Coverage Summary by Structure:

<table>
<thead>
<tr>
<th>Design Scope</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>test_unit.copied_from_foczynghdl</td>
<td>73.13%</td>
</tr>
<tr>
<td>u_Test_Unit.copied_from_foczynghdl</td>
<td>86.66%</td>
</tr>
<tr>
<td>u_FOC_Velocity_Encoder</td>
<td>72.28%</td>
</tr>
<tr>
<td>outputpm(0)</td>
<td>100.00%</td>
</tr>
<tr>
<td>outputpm(1)</td>
<td>100.00%</td>
</tr>
<tr>
<td>outputpm(2)</td>
<td>100.00%</td>
</tr>
</tbody>
</table>

Coverage Summary by Type:

<table>
<thead>
<tr>
<th>Coverage Type</th>
<th>Hits</th>
<th>Misses</th>
<th>Weight</th>
<th>% Hit</th>
<th>Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statements</td>
<td>667</td>
<td>659</td>
<td>8</td>
<td>1</td>
<td>98.80%</td>
</tr>
<tr>
<td>Branches</td>
<td>308</td>
<td>205</td>
<td>12</td>
<td>1</td>
<td>95.77%</td>
</tr>
<tr>
<td>FEC Expressions</td>
<td>30</td>
<td>24</td>
<td>6</td>
<td>1</td>
<td>80.00%</td>
</tr>
<tr>
<td>FEC Conditions</td>
<td>150</td>
<td>112</td>
<td>42</td>
<td>1</td>
<td>73.33%</td>
</tr>
<tr>
<td>Toggles</td>
<td>2036</td>
<td>13736</td>
<td>6340</td>
<td>1</td>
<td>68.39%</td>
</tr>
<tr>
<td>FSMs</td>
<td>14</td>
<td>3</td>
<td>111</td>
<td>1</td>
<td>21.42%</td>
</tr>
<tr>
<td>States</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>25.00%</td>
</tr>
<tr>
<td>Transitions</td>
<td>10</td>
<td>2</td>
<td>8</td>
<td>1</td>
<td>20.00%</td>
</tr>
</tbody>
</table>

Report generated by Questa (ver. 10.4d) on Dienstag 10 Oktober 2017 14:39:53 with command line:
coverage report -html CodeCoverage.html
Summary

- Model-based Design fits well for SW / HW Code-Design
- All Model-based Design Verification and Validation methods fully applicable, including DO-178C / DO-254 Support
- HDL Code Generation fully automatic
- Test suite is reused for HDL Code results verification