Integrated Workflow to Implement Embedded Software and FPGA Designs on the Xilinx Zynq Platform

Puneet Kumar
Senior Team Lead - SPC
Agenda

- **Integrated Hardware / Software Top down Workflow for Zynq**, highlighting:
  - Automatic Code Generation:
    - HDL code generation for the FPGA fabric and C-Code generation for the ARM MCU
  - Automatic Interface Logic Generation:
    - Generation of the interface logic and software between the FPGA and ARM.
  - Integrated Verification:
    - Integrated HDL Verification using HDL Co-simulation and FPGA-in-Loop

- Q&A
Demo - Zynq Model-Based Design Workflow

Using IP Core Generation Workflow: Sobel Edge Detection

This example shows how to use HDL Coder to generate a custom IP core which performs Sobel edge detection processing on streaming video.

In MATLAB, type the following:

```
hdladvisor('hdlcoder_sobel_video/Sobel_HW')
```

Launch HDL Workflow Advisor

Run Demo

Copyright 2012 The MathWorks, Inc.
Who is Who???

- Who is a System Engineer?
- Who is an FPGA/ASIC designer?
- Who is using MATLAB?
- Who is using Simulink?
- Who is converting MATLAB to C or HDL?
Algorithm Development Process

- Explore and discover
- Gain insight into problem
- Evaluate options, trade-offs

Desktop
- .dll
- .exe
- .c, .cpp

Embedded
- C
- VHDL / Verilog
- Structured Text

Test & Verification

Requirements

Research & Design

Implementation
You May Have Some Questions?

- Can I generate HDL and C code from my MATLAB and Simulink models?

- Can I generate integrated test-bench to co-simulate it with HDL Simulators?

- Can I Generate the interface logic?

- What about porting it on SoC FPGA and verifying the DUT with the golden MATLAB/Simulink test-bench?

- And Many more questions…
Solution: C and HDL Code Generation

- Design, execute, and verify algorithms in MATLAB
- Automatically generate C or HDL code
- Deploy generated code on hardware
The Algorithm Design Challenge

- How can we:
  - Implement designs on SoC FPGA’s?
  - Partition the HW and SW?
  - Generate the Interface Logic?
What is SoC FPGA’s?

- FPGA + ARM® on one chip
  - Enables high-performance system development
  - Reduces cost over multi-chip solutions
Design Challenge

- Typically programmed in C
- Often runs a Linux operating system
- Well-established workflows exist

CHALLENGES
- FPGA Designers not familiar with programming processors
- What should run on the processor vs. the FPGA?
Design Challenge

- Typically programmed in VHDL/Verilog
- Established workflows exist

CHALLENGES
- DSP/Processor programmers not familiar with FPGA Design
- What should run on the FPGA vs. the processor?
Design Challenge

CHALLENGES

- No established rules for hooking up the interface
- Different versions of AXI interface for different bandwidth requirements

- Zynq uses “standard” AXI4 interface between FPGA and ARM
How can I address these challenges

- Model-Based Design provides a single environment from requirements to prototype
- A **guided workflow** for hardware and software development
Model-Based Design
Why Model-Based Design?

Requirements Development
Simulation
Code Generation
Continuous Verification
Model-Based Design:

From Concept to Production

- Model multi-domain systems
- Explore and optimize system behavior in floating point and fixed point
- Collaborate across teams and continents

- Generate efficient code
- Explore and optimize implementation tradeoffs

- Automate regression testing
- Detect design errors
- Support certification and standards
Design Challenges

- **FPGA Designers** not familiar with programming processors
- **DSP/Processor programmers** not familiar with FPGAs
- What should run on the FPGA vs. what should run on the ARM?
- No established rules for hooking up the interface between FPGA and ARM processor
High-Level Zynq Design Flow

User defines partitioning

MathWorks automates code and interface-model generation

MathWorks automates the build and download through the Xilinx tools
Model-Based Design for Zynq
Demo - Zynq Model-Based Design Workflow

Using IP Core Generation Workflow: Sobel Edge Detection

This example shows how to use HDL Coder to generate a custom IP core which performs Sobel edge detection processing on streaming video.

In MATLAB, type the following:

```matlab
hdldesigner('hdlcoder_sobel_video/Sobel_HW')
```
Model-Based Design flow using Simulink
from Algorithm to FPGA Implementation
Verification Landscape:

**Model**
- Requirements
- Functional
- Equivalence
- Coverage
- Property Proving
- Virtual Platforms

**VHDL / Verilog**
- Requirements
- Equivalence
- Coverage
- Assertions

**FPGA**
- Equivalence
- Regression
- Timing Analysis

```vhdl
-- <<1\>/Unit Delay
Unit_Delay_process : PROCESS (clk, reset)
BEGIN
  IF reset = '1' THEN
    Unit_Delay_out1 <= to_signed(0, 16);
  ELSIF clk\#EVENT AND clk = '1' THEN
    IF enb = '1' THEN
      Unit_Delay_out1 <= filter_subsystem_out1_signed;
    END IF;
  END IF;
END PROCESS Unit_Delay_process;

LeftOut <= std_logic_vector(Unit_Delay_out1);
```
Verification Challenges:
Stimuli-Driven Test Bench in HDL Simulators

- Digital waveforms are difficult to analyze
  - Application specific analysis methods are needed
- How to get test vectors to achieve 100% test coverage?
  - Formal methods to derive required test cases
HDL cosimulation to verify HDL
Re-use System Level Test Bench for HDL Verification

- Re-use test benches for equivalence checking
- Integrate with HDL code coverage analysis
- Flexible test bench creation: closed loop, multi domain
- Also works with handwritten code
- Integrate with Modelsim/Questa and Incisive
Verification Landscape Solution:
Re-use System Level Test Bench

Model
- Requirements
- Functional
- Equivalence
- Coverage
- Property Proving
- Virtual Platforms

VHDL / Verilog
- Requirements
- Equivalence
- Coverage
- Assertions

FPGA
- Equivalence
- Regression
- Timing Analysis
Demo - Zynq Model-Based Design Workflow

This example shows how to use HDL Coder to generate a custom IP core which perform Sobel edge detection processing on streaming video.

In MATLAB, type the following:
```
hdladvisor(hdlcoder_sobel_video/Sobel_HW)
```
Zynq Model-Based Design Workflow

This example shows how to use HDL Coder to generate a custom IP core which perform Sobel edge detection processing on streaming video. In MATLAB, type the following:

```matlab
hdltarget('hdlcoder_sobel_video/Sobel_HW')
```
Zynq Model-Based Design Workflow

MATLAB® and Simulink® Algorithm and System Design

HDL IP Core Generation

Programmable Logic IP Core

AXI4-Stream Video In
AXI4-Stream Video Out

Algorithm from MATLAB/Simulink

AXI Lite Accessible Registers

HDL IP Core Generation

External Ports
Zynq Model-Based Design Workflow

1. **MATLAB® and Simulink® Algorithm and System Design**
   - HDL IP Core Generation
   - EDK Integration

2. **Programmable Logic IP Core**
   - Algorithm from MATLAB/Simulink
   - AXI Lite Accessible Registers
   - AXI4-Stream Video In
   - AXI4-Stream Video Out

3. **EDK Project**
   - AXI Lite Accessible Registers
   - AXI Video DMA
   - AXI4-Stream Video In
   - AXI4-Stream Video Out

4. **Zynq Platform**
   - FPGA Bitstream

5. **EDK Integration**
   - External Ports
Zynq Model-Based Design Workflow

MATLAB® and Simulink® Algorithm and System Design

HDL IP Core Generation

EDK Integration

SW Interface Model Generation

H/W

FPGA Bitstream

SW Build

Zynq Platform

SW Interface Model Generation

SW Interface Model

SW I/O Driver Blocks

SW

SW

SW
Zynq Model-Based Design Workflow

- Real-time Parameter Tuning and Verification
  - External Mode
  - Processor-in-the-loop
Zynq in Action

Ball Tracking Demo

- Video stream filtered to find and mark greenish ball
- FPGA (HW): Ball Tracking
- ARM (SW): Draw marker
**Fast Prototyping and Iteration**

Using IP Core Generation Workflow: Sobel Edge Detection

Fast prototyping, iteration, and live probing/tuning directly on Zynq hardware.
Abstraction is Key

AXI4-Lite Interface

AXI4-Stream Interface
Abstraction is Key

- Focus on algorithm and system design
- Stay on higher level of abstraction
- Automatic code generation and HW/SW integration
Zynq HW/SW Co-design Workflow Summary

- **HW Design**
  - Simulink Model

- **IP Core Generation**
  - Algorithm from MATLAB and Simulink
  - FPGA IP Core

- **Generate SW Interface Model**
  - SW Interface Model

- **Processor**
  - AXI4-Lite Bus

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

- **Generate SW Interface Model**
  - SW Interface Model

- **FPGA Bitstream**
  - SW Build

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

- **External Ports**

- **External Ports**

- **Embedded System Project**

- **AXI Lite Accessible registers**

- **AXI Lite Accessible registers**

- **AXI Lite Accessible registers**
Thank You!