MATLAB EXPO 2016
KOREA
4월 28일 (목)
등록 하기 matlabexpo.co.kr
MATLAB과 Simulink를 이용한 프로그래머블 SoC 설계

이웅재 부장
Application Engineering Group
Agenda

- Introduction
  - What is Zynq?
  - Design Challenges

- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow

- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop

- Advanced Features

- Conclusions
Top Story

Zynq Book Reaches Top-10 Best Seller

The Zynq Book, written by a team at University of Strathclyde, Glasgow, UK, is already a tremendeous seller with high status on Amazon.com. The book is a comprehensive guide to using Xilinx Zynq devices, covering all aspects of development and implementation. The book also features a free PDF download of the book for a limited time.

Get the free PDF download »
Buy a hard copy on Amazon.com.
Learn more about the Zynq SoC from Xilinx.
What is Zynq?

- New product family from Xilinx®
  - All Programmable System on Chip (SoC)
- FPGA Fabric + ARM® on one a single chip
  - Enables high-performance system development
  - Reduces BOM cost over multi-chip solutions
Zynq 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
How can I address these challenges and get my project onto Zynq quickly?

- *Model-Based Design* provides a single environment from requirements to prototype *seamlessly*.

- A *guided workflow* for hardware and software development.
Hardware platform to prototyping

- **Board Name:** ZedBoard™
  - Zynq-7000 AP SoC XC7Z020-CLG484
- **Memory:**
  - 512 MB DDR3
  - 256 Mb Quad-SPI Flash
  - 4 GB SD card
- **Onboard USB-JTAG Programming**
- 10/100/1000 Ethernet
- USB OTG 2.0 and USB-UART
- PS & PL I/O expansion (FMC, Pmod™, XADC)
- Multiple displays (1080p HDMI, 8-bit VGA, 128 x 32 OLED)
- I²S Audio CODEC
System configuration for prototyping
Support Package Installer
Agenda

- Introduction
  - What is Zynq?
  - Design Challenges

- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow

- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop

- Advanced Features

- Conclusions
Model-Based Design for Zynq

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
Code Generation Scheme
Multi-Domain, Multi-Target Technology

Multiple analyses and optimizations

Intermediate Representation (IR)

Analysis/Transform

Target Language Backend

Multiple Targets

Multiple Domains

Simulink

Stateflow

MATLAB Function

Simscape

C
C++
HDL
PLC
Code Generation Products for C/C++

**Embedded Coder**
Automatically generate C and C++ optimized for embedded systems

**Simulink Coder**
Automatically generate C and C++ from Simulink models and Stateflow charts

**MATLAB Coder**
Automatically generate C and C++ from MATLAB code
Optimize the C/C++ code for performance

- SIMD intrinsics
- Fixed-point intrinsics
- Assembly
- Optimized libraries

1. Optimize the generated C/C++ code
2. Use the ARM NEON Media Processing Engine
Code Generation Products for VHDL/Verilog

**HDL Coder™**
Automatically generate synthesizable RTL code (VHDL or Verilog) from MATLAB code and Simulink Model

**MATLAB® Coder™**
Automatically generate C and C++ from MATLAB code
HDL Coder Key Features

- **Code Generation**
  - Target-independent Synthesizable RTL Code
  - IEEE 1376 compliant VHDL®
  - IEEE 1364-2001 compliant Verilog®

- **Verification**
  - Generate HDL test-bench
  - Co-simulate with ModelSim and Incisive*

- **Design automation**
  - Synthesize using integrated Xilinx and Altera synthesis tool interface
  - Optimize for area-speed
  - Program Xilinx and Altera boards

* HDL Verifier required for co-simulation and FPGA-in-the-loop verification
Zynq Model-Based Design Workflow
Zynq Model-Based Design Workflow

MATLAB® and Simulink® Algorithm and System Design

HDL IP Core Generation

HDL IP Core Generation

Programmable Logic IP Core

Algorithm from MATLAB/ Simulink

AXI4-Stream Video In

AXI4-Stream Video Out

AXI Lite Accessible Registers

External Ports
Zynq Model-Based Design Workflow

MATLAB® and Simulink®
Algorithm and System Design

HDL IP Core Generation

EDK/Vivado Integration

FPGA Bitstream

Zynq Platform

AXI4-Stream Video In
AXI4-Stream Video Out

Algorithm from MATLAB/Simulink

EDK/Vivado Integration

AXI Lite Accessible Registers

Programmable Logic IP Core

External Ports

Processing System

AXI Video DMA

AXI4-Lite

Algorithm from MATLAB/Simulink

Programmable Logic IP Core

EDK Project

External Ports
Zynq Model-Based Design Workflow

MATLAB® and Simulink®
Algorithm and System Design

HDL IP Core Generation

EDK Integration

SW Interface Model Generation

FPGA Bitstream

SW Build

Zynq Platform

SW

HW

SW I/O Driver Blocks

SW Interface Model Generation

SW Interface Model
Zynq Model-Based Design Workflow

- **MATLAB® and Simulink® Algorithm and System Design**
- **HDL IP Core Generation**
- **EDK Integration**
- **SW Interface Model Generation**
  - **FPGA Bitstream**
  - **SW Build**
- **Zynq Platform**
- **External Mode / PIL / UDP**

- **Real-time Parameter Tuning and Verification**
  - External Mode
  - Processor-in-the-loop
  - UDP

- **More probe and debug capability in the future**
Communication between ARM & FPGA

- Defines the synchronization behavior between ARM and FPGA
- Free running mode
  - No explicit synchronization between ARM and FPGA
- Co-processing mode
  - Explicit blocking synchronization between ARM and FPGA
- Choose in HDL Workflow Advisor
  - xPC/FPGA Turnkey Workflow
  - Zynq IP Core Generation
Agenda

- Introduction
  - What is Zynq?
  - Design Challenges

- **ZYNQ Design Process**
  - MBD on Programmable SoC
  - Code Generation
  - Workflow

- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop

- Additional Features

- Conclusions
Verification Challenges:

- Design the test bench twice
  - 10 – to – 1 ratio of Test bench LOC – to – Design LOC (for HDL)
- Many stimuli-files from MATLAB
- How to analyze results?
Parameter tuning through External Mode

- Tuning model parameters and evaluate the effects of different parameter values on model results in real time.

- Helping you find the optimal values to achieve desired performance.
Verification through UDP Interface

Fast prototyping, iteration, and live probing/tuning directly on ZYNQ hardware
Design Partitioning for UDP

Top-Level Model
• From Multimedia File block
• To Audio Device block
• Resampling, buffering, unbuffering
• Data type conversion
• UDP Send and UDP Receive blocks

PL Subsystem
• Equalizer filters
• Filter selector switches

PS components
• UDP Send and UDP Receive blocks
• Supports

Source Components
Zynq Algorithm
Sink Components

Source Components
UDP Send
UDP Receive
Zynq Algorithm
UDP Send
UDP Receive
Sink Components

demo
Processor-In-the Loop (PIL) Verification

Help you **evaluate the behavior** of a candidate algorithm on the target both PS/PL and **profile** the execution times
Partitioning Scheme via PIL

demo
## Verification Tradeoffs

<table>
<thead>
<tr>
<th>Feature</th>
<th>External Mode</th>
<th>PIL</th>
<th>UDP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-time execution</td>
<td>✔ Yes</td>
<td>✗ No</td>
<td>✔ Yes</td>
</tr>
<tr>
<td>Parameter tuning</td>
<td>✔ Yes</td>
<td>✗ No</td>
<td>✗ No</td>
</tr>
<tr>
<td>Test bench options</td>
<td>✗ Limited</td>
<td>✔ Unlimited</td>
<td>✔ Unlimited</td>
</tr>
<tr>
<td>Code verification</td>
<td>✗ No</td>
<td>✔ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>Execution profiling</td>
<td>✗ No</td>
<td>✔ Yes</td>
<td>✗ No</td>
</tr>
<tr>
<td>Single model</td>
<td>✔ Yes</td>
<td>✔ Yes/No</td>
<td>✗ No</td>
</tr>
<tr>
<td>Data synchronization</td>
<td>✗ Limited</td>
<td>✔ Yes</td>
<td>✗ Limited</td>
</tr>
</tbody>
</table>
Agenda

- Introduction
  - What is Zynq?
  - Design Challenges

- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow

- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop

- Advanced Features

- Conclusions
Recently updated features in R2015a/R2015b

- **R2015a**
  - Internal-interface API in custom reference design
  - Zynq AXI Stream support
  - Support Front Digital IO of Speedgoat IO331 board
  - Save target setting as model properties

- **R2015b**
  - Command-line API for WFA
  - Export/Import between API and WFA
  - Map tunable parameters to AXI interfaces
  - AXI stream vector mode
  - Internal-interface API in custom reference design
Custom Board and Reference Design API for IP Core Generation Workflow

Define your own Zynq or Altera SoC Board and Reference Design

- Enable fast on-board prototyping and iteration
- Access to SW interface model generation, AXI driver, External Mode

Simulink/MATLAB algorithm

Algorithm Model

HDL Coder

Generic IP across platforms

AXI Interface

Algorithm HDL

HDL IP Core

Prototyping the generated IP on custom Reference Designs and SoC Boards

Processor

HDL IP core

Custom Reference Design

Custom Reference Design

HDL Coder

R2015a
Custom Reference design API extension for Internal Interface

- Define an interface that connects to another IP in the reference design

```matlab
% add reference design internal interface
hRD.addInternalIOInterface( ... 'InterfaceID', 'ADC Data In', ... 'InterfaceType', 'IN', ... 'PortName', 'ADCDataIn', ... 'PortWidth', 16, ... 'InterfaceConnection', 'hdlooder_adc_interface_ipcore_0/DataOut');
```
Zynq Streaming Interface Support

- Generate HDL IP core with AXI4-Stream interface
- Enable high speed data transfer
- Simplify streaming protocol
AXI4-Stream Vector Mode

- Modeling HW and SW together
- Automatic generation of SW DMA driver
- Focus on HW/SW Rapid Prototyping
Agenda

- Introduction
  - What is Zynq?
  - Design Challenges

- ZYNQ Design Process
  - MBD on Programmable SoC
  - Code Generation
  - Workflow

- Verification & Partitioning
  - Parameter Tuning
  - UDP Interface
  - Processor In the Loop

- Advanced Features

- Conclusions
Conclusions: Design Concept Summary

- Focus on algorithm and system design
- Stay on higher level of abstraction
- Automatic code generation and HW/SW integration
- Fast Prototyping and Easy integration with IDE
- Partitioning through Profiling
## Conclusions: Related Training Summary

<table>
<thead>
<tr>
<th>Training Title</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Signal Processing with MATLAB</td>
<td>Signal Analysis and Algorithm Design (2 days)</td>
</tr>
<tr>
<td>Image Processing with MATLAB</td>
<td>Image Analysis and Processing (2 days)</td>
</tr>
<tr>
<td>Signal Processing with Simulink</td>
<td>Signal Processing Based Modeling &amp; Dynamic Simulation (3 days)</td>
</tr>
<tr>
<td>Communication System Modeling with Simulink</td>
<td>Communication System Modeling &amp; Simulation (1 day)</td>
</tr>
<tr>
<td>Stateflow for Logic-Driven System Modeling</td>
<td>Flow Chart and FSM Modeling (2 days)</td>
</tr>
<tr>
<td>Interfacing MATLAB with C</td>
<td>Interfacing MATLAB and C Code via APIs (1 days)</td>
</tr>
<tr>
<td>MATLAB to C with MATLAB Coder</td>
<td>C/C++ code generation for Embedded System from Simulink Model (3 days)</td>
</tr>
<tr>
<td>Embedded Coder for Production Code Generation</td>
<td></td>
</tr>
<tr>
<td>Generating HDL Code from Simulink</td>
<td>HDL Code Generation &amp; Verification (2 days)</td>
</tr>
<tr>
<td>Programming Zynq with MATLAB and Simulink</td>
<td>HW/SW Co-design on Zynq (include ZedBoard) (2 days)</td>
</tr>
</tbody>
</table>

**Questions?**
Thank You