MATLAB EXPO 2018
Designing and Prototyping Digital Systems on SoC FPGA

Hitu Sharma
Application Engineer

Vinod Thomas
Sr. Training Engineer
What is an SoC FPGA?

A typical SoC consists of:

- A microcontroller, microprocessor or digital signal processor (DSP) core
- Programmable Logic (FPGA)
- Memory blocks
- External interfaces such as USB, FireWire, Ethernet, USART, SPI, etc.
- Analog interfaces including ADCs and DACs
- Voltage regulators and power management circuits

Combines **High-speed compute capabilities of FPGAs** and the ability to perform **Complex operations on DSPs or MCUs**
Conventional SoC Design Workflow

1. Algorithm Design
2. System Architecture
3. C Implementation
   - C/C++
   - Integrate with Peripheral Drivers
4. HDL Implementation
   - VHDL/Verilog
   - Integrate with Peripheral IPs
5. Integrate C Code Verification
6. Integrate HDL Code Verification
7. Integrate and Prototype Application on SoC

MATLAB EXPO 2018
Application Examples

Vision

SDR

Motor

MATLAB EXPO 2018
Customer Case Study: Punch Powertrain

“\[This\ would\ not\ have\ been\ possible\ to\ design\ at\ all\ previous\ to\ adopting\ HDL\ code\ generation\ from\ Simulink\ model-based\ design\]”

Requirements

- New switched reluctance motor – new complex control strategies
- Fast: 2x the speed of their previous motor
- Target to a Xilinx® Zynq® SoC 7045 device
- Needed to get to market quickly
- No experience designing FPGAs!

☑ Designed integrated E-drive: Motor, power electronics and software
☑ 4 different control strategies implemented
☑ Done in 1.5 years with 2FTE's
☑ Models reusable for production
☑ Smooth integration and validation due to development process – thorough validation before electronics are produced and put in the testbench
Challenges in SoC design

Partitioning of Algorithm
- What goes on FPGA and what goes on Processor
- Change in architecture involves reprogramming

Programming and Verification Expertise
- Proficiency in both HDL and C programming
- Both have different verification methodologies
- Late detection of errors

Interface Between Software & Hardware
- Ensure Reliable transfer of data between FPGA and Processor

Verification of the Design
- SoC Verification

Applications & Custom Board
- This involves configuring different peripherals
From Behavioral Model to Implementation Model

Behavioral algorithm

Hardware micro-architecture

Fixed-point implementation

MATLAB EXPO 2018
Target Receiver/Transmitter on Systems-on-Chip (ARM/FPGA)

Run on Programmable Logic

Run on ARM Processing System
Addressing Challenges in SoC Design

- **Partitioning of Algorithm**
- **Programming and Verification Expertise**
  - Proficiency in both HDL and C programming
  - Both have different verification methodologies
  - Late detection of errors
- **Interface Between Software & Hardware**
  - Ensure Reliable transfer of data between FPGA and Processor
- **Verification of the Design**
  - SoC Verification
- **Applications & Custom Board**
  - This involves configuring different peripherals

**Simulink for System Architecture Modeling**
Implement and Prototype Algorithms in Hardware

MATLAB and Simulink

- Embedded Coder
  - C, C++
- HDL Coder
  - VHDL, Verilog

Platforms:
- DSP / MCU
- Programmable SoC
- FPGA
- ASIC

Steps:
1. Prepare model for IP core generation
2. Configure Interface Logic
3. RTL Code Generation for IP Core
4. Generate Software/Hardware Model
5. Synthesis/Bit File Generation
6. Deployment
HDL code Generation

HW/SW Co-design QPSK Transmitter and Receiver Using Analog Devices AD9361/ AD9364 Hardware Generation Model

Copyright 2016 The MathWorks, Inc.
Generate Software Interface model

- Prepare model for IP core generation
- Configure Interface Logic
- RTL Code Generation for IP Core
- Generate Software/Hardware Model
- Synthesis/Bit File Generation
- Deployment
Algorithm Partitioning and HW-SW Co-Design
HDL Optimizations

- **Speed Optimization**
  - Distributed Pipelining
  - Adaptive Pipelining
  - Clock-Rate Pipelining

- **Area Optimization**
  - Streaming
  - RAM Mapping
  - Resource Sharing
Hardware Verification

- “Validation Model” generation by HDL Coder

- RTL code against Implementation Model
  - HDL Cosimulation through HDL Verifier

- Hardware Results against Implementation Model
  - FPGA-in-the-loop verification through HDL Verifier
Software in the Loop (SIL) Verification
HDL Library Summary
# HDL Library Summary

- **HDL Coder**
  - Stateflow
  - DSP System Toolbox HDL Support
    - Filtering
    - Math Functions
    - Signal Management
    - Signal Operations
    - Sinks
    - Sources
    - Statistics
    - Transforms
- **Communications System Toolbox HDL Support**
  - Comm Filters
  - Comm Sinks
  - Comm Sources
  - Error Detection and Correction
  - Interleaving
  - Modulation
- **Vision HDL Toolbox**
  - Analysis & Enhancement
  - Conversions
  - Filtering
  - Geometric Transforms
  - I/O Interfaces
  - Morphological Operations
  - Statistics
  - Utilities
- **HDL Verifier**
  - For Use with Cadence Incisive
  - For Use with Mentor Graphics ModelSim
- **LTE HDL Toolbox**
  - Error Detection and Correction
  - I/O Interfaces
  - Utilities
  - Recently Used

---

### MATLAB EXPO 2018
LTE HDL-optimized blocks

Reference applications

Utilities

- Turbo Encoder
- Turbo Decoder
- Convolutional Encoder
- Convolutional Decoder
- CRC Encoder
- CRC Decoder

- PSS/SSS Detection
- MIB Recovery
- LTE Frequency Scanner

- Frame-to-samples / samples-to-frame
- Sample bus creator / selector
- Templates to connect MATLAB tests/golden reference to Simulink HW implementation

MATLAB EXPO 2018
Vision HDL Toolbox

*Design and prototype video image processing systems*

- Modeling hardware behavior of the algorithms
  - Pixel-based functions and blocks
  - Conversion between frames and pixels
  - Standard and custom frame sizes

- Prototyping algorithms on hardware
  (With HDLCoder) Efficient and readable HDL code
  (With HDL Verifier) FPGA-in-the-loop testing and acceleration
## C code supported libraries

<table>
<thead>
<tr>
<th>Product</th>
<th>Extends Code Generation Capabilities for ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aerospace Blockset™</td>
<td>Aircraft, spacecraft, rocket, propulsion systems, and unmanned airborne vehicles</td>
</tr>
<tr>
<td>Audio System Toolbox™</td>
<td>Audio processing systems</td>
</tr>
<tr>
<td>Automated Driving System Toolbox™</td>
<td>Designing, simulating, and testing ADAS and autonomous driving systems</td>
</tr>
<tr>
<td>Communications System Toolbox™</td>
<td>Physical layer of communication systems</td>
</tr>
<tr>
<td>Computer Vision System Toolbox™</td>
<td>Video processing, image processing, and computer vision systems</td>
</tr>
<tr>
<td>Control System Toolbox™</td>
<td>Linear control systems</td>
</tr>
<tr>
<td>DSP System Toolbox™</td>
<td>Signal processing systems</td>
</tr>
<tr>
<td>Embedded Coder</td>
<td>Embedded systems, rapid prototyping boards, and microprocessors in mass production</td>
</tr>
<tr>
<td>Fixed-Point Designer™</td>
<td>Fixed-point systems</td>
</tr>
<tr>
<td>Fuzzy Logic Toolbox™</td>
<td>System designs based on fuzzy logic</td>
</tr>
<tr>
<td>HDL Verifier™</td>
<td>Direct programming interface (DPI) component and transaction-level model (TLM) generation from Simulink</td>
</tr>
<tr>
<td>IEC Certification Kit</td>
<td>ISO 26262 and IEC 61508 certification</td>
</tr>
<tr>
<td>Model-Based Calibration Toolbox™</td>
<td>Developing processes for systematically identifying optimal balance of engine performance, emissions, and</td>
</tr>
<tr>
<td></td>
<td>fuel economy, and reusing statistical models for control design, hardware-in-the-loop (HIL) testing, or</td>
</tr>
<tr>
<td></td>
<td>powertrain simulation</td>
</tr>
</tbody>
</table>
Addressing Challenges in SoC Design

Partitioning of Algorithm

Simulation for System Architecture Modeling

Programming and Verification Expertise

HDL and Embedded Coder

Interface Between Software & Hardware

- Ensure Reliable transfer of data between FPGA and Processor

Verification of the Design

- SoC Verification

Applications & Custom Board

- This involves configuring different peripherals
Interface Between Hardware and Software

HDL Coder and Embedded Coder supports:
- AXI4 Stream Master/Slave
- AXI4 Lite
- AXI4 Stream Master/Slave Video
- AXI4
Interface Configuration
Addressing Challenges in SoC Design

Partitioning of Algorithm

Simulink for System Architecture Modeling

Programming and Verification Expertise

Automatic C and HDL Code Generation

Interface Between Software & Hardware

Generation of AXI Protocol Drivers

Verification of the Design

• SoC Verification

Applications & Custom Board

• This involves configuring different peripherals
SoC Verification using PIL

Top-Level Model

Input from MATLAB or Simulink

PIL Model
(On Zynq)

Zynq Model
(in Simulink)

Output from Zynq

Output from Simulink

=?
Addressing Challenges in SoC Design

- Proficiency in both HDL and C programming
- Both have different verification methodologies
- Late detection of errors

Partitioning of Algorithm
- Simulink for System Architecture Modeling

Programming and Verification Expertise
- Automatic C and HDL Code Generation

Interface Between Software & Hardware
- Generation of AXI Protocol Drivers

Verification of the Design
- Unified Verification Framework

Applications & Custom Board
- This involves configuring different peripherals
Board and Reference Design

Reference Design

Processor

Placeholder for Algorithm IP Core

AXI4-Lite

SoC Board (peripherals, daughter-cards)

Simulink/MATLAB algorithm

Algorithm Model

HDL Coder

Generic IP across platforms

AXI Interface

Algorithm HDL

HDL IP Core

HDL IP core

MATLAB EXPO 2018
Multiple Reference Designs

Processor

AXI4-Lite

Placeholder for Algorithm IP Core

Reference Design

Processor

AXI4-Lite

AXI DMA

Placeholder for Algorithm IP Core

SoC Board

MATLAB EXPO 2018
External and Internal Interfaces

Reference Design

Processor

AXI4-Lite

AXI DMA

HDL IP core

ADC Interface

Interface

LED

SoC Board

External Interface

Internal Interface

AXI Interface

External Interface

SoC Board

AXI4_Lite

IPCORE_CLK

IPCORE_RESETN

LEDs[3:0]

AXI4_Lite_ACLK

AXI4_Lite_ARESETN

MATLAB EXPO 2018
Reference Design for Software Defined Radio

- **Host Computer**: Interface with other components
- **Embedded Coder**: Performs data generation and post-processing
- **HDL Coder**: Contains AXI Interface driver and RF HDL IP
- **RF Card**: Transmits and receives data
- **Radio Configuration Settings**: Adjusts settings for the RF Card

Components include:
- **AXI Interface IP**: Transmitter and Receiver functionality
- **UDP/SPI/JTAG**: Interfaces for communication and configuration

This diagram illustrates how a reference design for a software-defined radio is implemented, integrating different hardware and software components for efficient data transmission and processing.
Software Defined Radio Workflow
Application Example: Software Defined Radio
Reference Design for Computer Vision Applications

Camera In (VDMA) | AXI4-Stream Video Module | Video Signals | Ready | Signal 3 | Registers | ARM

HDL Coder | Embedded Coder

AXI4-Stream Video to Pixels

Signal 1
Signal 2
Signal 3

Pixels
Control Signals

Ready

External Memory Buffer (VDMA)

Camera In

MATLAB EXPO 2018

Generated HDL IP core

Input HDL Viewer

Output HDL Viewer

Reference Design for Computer Vision Applications

Generated HDL IP core

FPGA Algorithm

Software on ARM
Application Example: Pot-Hole Detection
Application Example: Pot-Hole Detection
Supported SoC platforms and Reference Designs

- Xilinx SoCs
  - ZedBoard
  - ZC702 Evaluation Board
  - ZC706 Evaluation Board
  - Analog Devices RF SOM

- Video and Image Processing
- Software-Defined Radio
- Motor Control
Supported Intel SoC platforms and Reference Designs

- Intel SoCs
  - Arrow SoC
  - Intel Cyclone V SoC

Video and Image Processing

Motor Control
From the MATLAB Toolstrip:

- Add-Ons
- 
- Get Hardware Support Packages
- Download Support Packages

From the MATLAB Command Line:

```
>> supportPackageInstaller
```
SoC Custom Reference Design

Example shipping with product

Define and Register Custom Board and Reference Design for SoC Workflow

This example shows how to define and register a custom board and reference design in the HDL Coder™ SoC workflow. Using this example, you create and export a custom reference design in the HDL Workflow Advisor for the SoC workflow.

This example uses a ZYBO Zynq board, but in the same way, you can define and register a custom board or a custom reference design for any Altera SoC board.

Requirements

- Set up the ZYBO board
- Create and export a custom reference design using Vivado
- Register the ZYBO board in HDL Workflow Advisor
- Register the custom reference design in HDL Workflow Advisor
- Execute the SoC Workflow for the ZYBO board

Understand the Board

Create and Export a Reference Design

Define the Board

Register the Board

Register the Reference Design

Create or Reuse Linux Image

Zynq/Altera SoC Workflow

Custom Board Work Flow

% Construct reference design object
hRD = hdlcoder.ReferenceDesign('SynthesisTool', 'Xilinx Vivado');
hRD.ReferenceDesignName = 'Default system with OLED and XADC';
hRD.BoardName = 'My ZedBoard';
Custom Board Work Flow

1.1. Set Target Device and Synthesis Tool

Set Target Device and Synthesis Tool for HDL code generation

- Input Parameters
  - Target workflow: IP Core Generation
  - Target platform: My ZedBoard
  - Synthesis tool: Xilinx Vivado
  - Family: Zynq
  - Package: clg484
  - Project folder: hdl_prj

Result: Passed Set Target Device and Synthesis Tool.
Addressing Challenges in SoC Design

- Proficiency in both HDL and C programming
  - Both have different verification methodologies
  - Late detection of errors

Programming and Verification Expertise
- Ensure Reliable transfer of data between FPGA and Processor
- Interface Between Software & Hardware
- Verification of the Design
- Applications & Custom Board

Partitioning of Algorithm
- Simulink for System Architecture Modeling
- Automatic C and HDL Code Generation
- Generation of AXI Protocol Drivers
- Unified Verification Framework
- Reference Designs
SoC FPGA Design Flow

RESEARCH -> REQUIREMENTS

DESIGN

Top-Level System Model

Software Model

Hardware Model

IMPLEMENTATION

Embedded Coder®

HDL Coder™

ARM

FPGA

SoC FPGA Template

Xilinx/Altera Embedded System Integration

User defines partitioning

MathWorks automates code and interface-model generation

MathWorks automates the build and download through the FPGA tools

MATLAB EXPO 2018
What’s New in FPGA and SoC Design?

- Floating Point Support **R2016b**
- FPGA Data Capture **R2017a**
- MATLAB as AXI Master **R2017a**
  - Access FPGA External Memory
- HDL Coder Support for Avnet Minize **R2017b**

**HDL Coder Native Floating Point**

- IEEE-754 Single precision support
- Extensive math and trigonometric operator support
FIL Verification: Supported Development Kits

- **PolarFire Eval Kit**
  - High Performance Kit for full development & testing

- **SmartFusion2 Advance Development Kit**
  - Full Feature Kit with Advanced Peripherals
Training Services

Exploit the full potential of MathWorks products

Flexible delivery options:

▪ Public training available in several cities
▪ Onsite training with standard or customized courses
▪ Web-based training with live, interactive instructor-led courses

More than 48 course offerings:

▪ Introductory and intermediate training on MATLAB, Simulink, Stateflow, code generation, and Polyspace products
▪ Specialized courses in control design, signal processing, parallel computing, code generation, communications, financial analysis, and other areas

www.mathworks.in/training
DSP for FPGAs

This three-day course will review DSP fundamentals from the perspective of implementation within the FPGA fabric. Particular emphasis will be given to highlighting the cost, with respect to both resources and performance, associated with the implementation of various DSP techniques and algorithms.

Topics include:

- Introduction to FPGA hardware and technology for DSP applications
- DSP fixed-point arithmetic
- Signal flow graph techniques
- HDL code generation for FPGAs
- Fast Fourier Transform (FFT) Implementation
- Design and implementation of FIR, IIR and CIC filters
- CORDIC algorithm
- Design and implementation of adaptive algorithms such as LMS and QR algorithm
- Techniques for synchronisation and digital communications timing recovery
Generating HDL Code from Simulink

two-day course shows how to generate and verify HDL code from a Simulink® model using HDL Coder™ and HDL Verifier™

Topics include:
- Preparing Simulink models for HDL code generation
- Generating HDL code and testbench for a compatible Simulink model
- Performing speed and area optimizations
- Integrating handwritten code and existing IP
- Verifying generated HDL code using testbench and cosimulation
Programming Xilinx Zynq SoCs with MATLAB and Simulink

two-day course focuses on developing and configuring models in Simulink® and deploying on Xilinx® Zynq®-7000 All Programmable SoCs. For Simulink users who intend to generate, validate, and deploy embedded code and HDL code for software/hardware codesign using Embedded Coder® and HDL Coder™. A ZedBoard™ is provided to each attendee for use throughout the course. The board is programmed during the class and is yours to keep after the training.

Topics include:

- Zynq platform overview and environment setup, introduction to Embedded Coder and HDL Coder
- IP core generation and deployment, Using AXI4 interface
- Processor-in-the-loop verification, data interface with real-time application
- Integrating device drivers, custom reference design
New: Software Defined Radio with Zynq using Simulink

- Learn the Model-Based Design workflow from simulation of RF chain, testing with Radio I/O to moving design to chip
- Get hands-on experience with PicoZed
  - Setting up and communicating with board
  - Capture over-the-air signal and process in MATLAB
  - AD9361 configuration
  - HW/SW co-design for SDR
### MathWorks Public Training

<table>
<thead>
<tr>
<th>Upcoming Public Trainings</th>
<th>Dates</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP for FPGAs</td>
<td>May 7-9</td>
<td>Bangalore</td>
</tr>
<tr>
<td>DSP for FPGAs</td>
<td>June 11-13</td>
<td>Hyderabad</td>
</tr>
<tr>
<td>Generating HDL Code from Simulink</td>
<td>Aug 27 - 28</td>
<td>Hyderabad</td>
</tr>
<tr>
<td>Programming Xilinx Zynq SoCs using MATLAB and Simulink</td>
<td>Aug 29 - 30</td>
<td>Hyderabad</td>
</tr>
<tr>
<td>Software Defined Radio with Zynq using Simulink</td>
<td>Aug 31</td>
<td>Hyderabad</td>
</tr>
<tr>
<td>Generating HDL Code from Simulink</td>
<td>Oct 8 -9</td>
<td>Bangalore</td>
</tr>
<tr>
<td>Programming Xilinx Zynq SoCs using MATLAB and Simulink</td>
<td>Oct 10 - 11</td>
<td>Bangalore</td>
</tr>
<tr>
<td>Software Defined Radio with Zynq using Simulink</td>
<td>Oct 12</td>
<td>Bangalore</td>
</tr>
</tbody>
</table>

Guaranteed to run
Call to Action

Videos and Webinars

- White Paper: Deploying LTE Wireless Communications on FPGAs: A Complete MATLAB and Simulink Workflow

- Webinars:
  - Modeling HDL Components for FPGAs in Control Applications
  - Prototyping SoC-based Motor Controllers with MATLAB and Simulink
  - Radio Deployment on SoC Platforms

- Video Series: Vision Processing for FPGA (5 Videos)

- Upcoming webinar on Microsemi Integration with MATLAB Tools

- Custom Reference Design

MATLAB EXPO 2018
Your feedback is valued.

Please complete the feedback form provided to you.
SIL MOdel

Intel SoC support
Training Dates