Integrated Software-Defined Radio on Zynq®-7000 All Programmable SoC Design Seminar
Course Objectives

During this seminar, you will gain insight into

- Avnet Zynq-7000 AP SoC / AD9361 Software-Defined Radio Kits
- Principles of wireless communication with examples of IEEE 802.11
- Model-based Design using MATLAB® and Simulink® for simulation, algorithm validation and automatic code generation for wireless communications
- Integrating Simulink models into Zynq-based software-defined radio using Xilinx Vivado® Design Suite
Wireless Communications System Design Challenges

- Systems start as abstract mathematical representations using statistical signal processing to model noise through the signal chain.
- Relentless market demand for higher data throughput requires deep expertise in high speed analog and digital design.
- Signals span wide range of frequencies from RF to baseband through analog and digital domains.
- Requires robust, traceable and verifiable auto code-generation from mathematical model to efficient hardware / software code.
Advantages of Zynq in Software-Defined Radio

- High-speed hardware-based digital signal processing PHY layer + software-based upper layer protocol stack
- Up to 16 12.5 Gb/s GT for connectivity to latest generation JESD204B data converters
- Higher integration for simpler and lower cost PCB
- Tight coupling of HW & SW integration for seamless algorithm partitioning
# Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</strong></td>
<td></td>
</tr>
<tr>
<td>Demo 1</td>
<td></td>
</tr>
<tr>
<td>Zynq / AD9361 SDR Kit in Operation / Base Reference Design</td>
<td></td>
</tr>
<tr>
<td><strong>Xilinx Vivado® Support for Zynq / AD9361 SDR Kit</strong></td>
<td></td>
</tr>
<tr>
<td>Demo 2</td>
<td></td>
</tr>
<tr>
<td>Exploring the Base Reference Design in Xilinx Vivado®</td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td></td>
</tr>
<tr>
<td><strong>Model-Based Design for Wireless Communications</strong></td>
<td></td>
</tr>
<tr>
<td>Demo 3</td>
<td></td>
</tr>
<tr>
<td>Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
<td></td>
</tr>
<tr>
<td>Deploying Simulink models with Xilinx Vivado Design Suite</td>
<td></td>
</tr>
<tr>
<td>Demo 4</td>
<td></td>
</tr>
<tr>
<td>802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
<td></td>
</tr>
</tbody>
</table>
Typical OFDMA Basestation

**Analog / RF**
- Low noise Amp
- RF IQ De-Mod
- RF IQ Mod
- ADC
- Power Amplifier

**Baseband Processing**
- FEC Decode
- FEC Encode
- Constellation Demapper
- Constellation Mapper
- FFT
- iFFT
- Baseband Interface (OBSAI / CPRI)

**Digital / Analog**
- DDC
- DUC
- DPD & CFR
- Convolutional FEC
- De-Mod
- Mod
- Low noise Amp
- Power Amplifier
- Conversion

**Network Interface**
- Packet Processing
- Network Interface
- OBSAI / CPRI

**Main Processor**
- Control Processor
- Control Interface

**Commercial Radio Communications Architecture**
Zynq-7000 AP SoC / AD9361 SDR Evaluation Kit

- **Hardware**
  - Avnet ZedBoard featuring Zynq 7020 All-Programmable SoC
  - Analog Devices AD-FMCOMMS2-EBZ High-speed analog FMC module with integrated RF agile transceiver
  - (4) LTE-band antennas
  - 8 GB SD card

- **Software Tools**
  - Xilinx Vivado® Design Edition
  - UBUNTU desktop Linux on Zynq
  - Reference Designs

- **Optional Development tools**
  - MathWorks DSP Wireless Communications Design Package for Xilinx Kits*

* Available in North America only.

www.em.avnet.com/adizynqsdr2

AES-ZSDR2-ADI-G $1295
Zynq-7000 AP SoC / AD9361 SDR Evaluation Kit

- **RF Band**
  - Factory-tuned for optimal performance at 2400 – 2500 MHz
  - Aimed at RF engineer seeking datasheet performance connecting to an RF testbench (VSA, Signal generator)

- **Synchronization**
  - Supports MIMO radio, with less than 1 sample sync on both ADC and DAC

- **Analog Devices AD-FMCOMMS2**
  - Analog Devices AD9361 RF Agile Transceiver
  - RF 2 × 2 transceiver with integrated 12-bit DACs and ADCs
  - Supports TDD and FDD operation
  - Tunable channel bandwidth: < 200 kHz to 56 MHz
  - RX gain control
    - Real-time monitor and control signals for manual gain
    - Independent automatic gain control
Zynq-7000 AP SoC / AD9361 SDR Systems Development Kit

● **Hardware**
  ○ Xilinx ZC706 base board featuring XC7Z045 device
  ○ Analog Devices AD-FMCOMMS3-EBZ High-speed analog FMC module with integrated RF agile transceiver
  ○ (4) LTE-band antennas
  ○ 8 GB SD card

● **Software Tools**
  ○ Xilinx Vivado® System Edition
    ▪ Device locked to XC7Z045
  ○ UBUNTU desktop Linux on Zynq
  ○ Reference Designs

● **Optional Development tools**
  ○ MathWorks Wireless Communications Design Package for Xilinx Kits*

* Available in North America only.

[Link to Avnet website](www.em.avnet.com/adizynqsdr3)

AES-ZSDR3-ADI-G $3595
Zynq-7000 AP SoC / AD9361 SDR Systems Development Kit

- **RF Band**
  - Operates over wide tuning range (70 MHz – 6 GHz)
  - Aimed at the system architect seeking a single platform for fast prototype of wideband wireless systems

- **Synchronization**
  - Supports MIMO radio, with less than 1 sample sync on both ADC and DAC

- **Analog Devices AD-FMCOMMS3**
  - Analog Devices AD9361 RF Agile Transceiver
  - RF 2 × 2 transceiver with integrated 12-bit DACs and ADCs
  - Supports TDD and FDD operation
  - Tunable channel bandwidth: < 200 kHz to 56 MHz
  - RX gain control
    - Real-time monitor + control signals for manual gain
Zynq-7000 Family Highlights

- **Complete ARM®-based Processing System**
  - Dual ARM Cortex™-A9 MPCore™
  - L1, L2 Caches and On-Chip Memory
  - Fully Integrated Memory Controllers
  - I/O Peripherals (CAN, USB, Ethernet, UART, …)

- **Tightly Integrated Programmable Logic**
  - Used to extend Processing System
  - Scalable density and performance
    - 30k – 440k LCs, 80 – 2,020 DSP Blocks

- **Flexible Array of I/O**
  - Wide range of external multi standard I/O
  - High performance integrated serial transceivers
  - Analog-to-Digital Converter inputs

Performance/power of an ASIC, flexibility of an FPGA, ease of use of an ASSP
Design Flow

- Mathematical representation of software-defined radio signal chain
- Model-based design in Simulink
  - Captured data as repeatable stimuli
  - RF impairments
  - System performance analysis and visualization
    - Frequency domain
    - Digital modulation scatter plots, etc

Simulation

- Integration with RF
- Performance analysis and visualization
**Design Flow**

- HDL Coder auto-code generation to bit-true / cycle-true HDL
- Import using Vivado IP Packager
- Vivado IP Integrator connects user model into existing base design
Design Flow

- Update SD card with design changes
- Test realtime operation on SDR kit
<table>
<thead>
<tr>
<th>Topic</th>
<th>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demo 1</td>
<td>Zynq / AD9361 SDR Kit in Operation / Base Reference Design</td>
</tr>
<tr>
<td></td>
<td>Xilinx Vivado® Support for Zynq / AD9361 SDR Kit</td>
</tr>
<tr>
<td>Demo 2</td>
<td>Exploring the Base Reference Design in Xilinx Vivado®</td>
</tr>
<tr>
<td>Break</td>
<td>Model-Based Design for Wireless Communications</td>
</tr>
<tr>
<td>Demo 3</td>
<td>Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
</tr>
<tr>
<td></td>
<td>Deploying Simulink models with Xilinx Vivado Design Suite</td>
</tr>
<tr>
<td>Demo 4</td>
<td>802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
</tr>
</tbody>
</table>
Zynq SDR Kit in Operation

- UBUNTU desktop Linux
- Analog Devices IIO Scope
- Manage all settings of FMCOMMS2 radio card
# Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 1</strong> Zynq / AD9361 SDR Kit in Operation / Base Reference Design</td>
<td></td>
</tr>
<tr>
<td>Xilinx Vivado® Support for Zynq / AD9361 SDR Kit</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 2</strong> Exploring the Base Reference Design in Xilinx Vivado®</td>
<td></td>
</tr>
<tr>
<td>Break</td>
<td></td>
</tr>
<tr>
<td>Model-Based Design for Wireless Communications</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 3</strong> Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
<td></td>
</tr>
<tr>
<td>Deploying Simulink models with Xilinx Vivado Design Suite</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 4</strong> 802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
<td></td>
</tr>
</tbody>
</table>
Design Flow

- Start with base HDL reference design from ADI
Zynq / AD9361 SDR Kit Base Reference Design

- SD card image / includes Linux kernel + devicetree.dtb + BOOT.BIN
  http://wiki.analog.com/resources/tools-software/linux-software/zynq_images

- AD-FMCOMMS2-EBZ + ZedBoard HDL Reference Design [Analog Devices Wiki]
  http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz/reference_hdl
Zynq / AD9361 SDR Kit Base Reference Design

http://wiki.analog.com/resources/eval/user-guides/ad-fmcomms2-ebz/reference_hdl

Download

TABLE OF CONTENTS
- AD-FMCOMMS2-EBZ / AD-FMCOMMS3-EBZ / AD-FMCOMMS4-EBZ HDL Reference Design
- Functional Overview
- Interface
- Transmitter
- Receiver
- Control and SPI
- Supported Devices
- Supported Carriers
- Download
- Generating Xilinx netlist files

DOWNLOAD
FPGA Reference Designs on GitHub:
- Vivado Downloads
  - https://github.com/analogdevicesinc/hdl (This is our main repository and contents change frequently. A release (more stable) branch will be available soon)
- XPS/EDK Downloads
- Previous Releases & Tags
  - https://github.com/analogdevicesinc/pbgahd_ulima/tags
  - https://github.com/analogdevicesinc/pbgahd_ulima/releases
- Git Repository
  - https://github.com/analogdevicesinc/pbgahd_ulima.git
Zynq / AD9361 SDR Kit Base Reference Design

- Base reference design built with TCL scripts in Xilinx Vivado.
Vivado Project Management with Tcl

- **Vivado flows can be run using Tcl**
  - Run full flow without interaction—launch script and walk away
  - Enables repeatability and self-documentation

- **Tcl commands can be**
  - Interactively entered at the Tcl prompt in Vivado IDE
  - Called from a Tcl script within Vivado Tcl shell
Zynq / AD9361 SDR Kit Base Reference Design
SDR Kit Base Reference Design with User Peripheral

Processing System
- Dual Cortex™-A9 MPCore™
- NEON™ / FPU Engine

Programmable Logic
- LINUX
- User-space applications
- LINUX drivers
- VDMA
- HDMI
- IIO oscilloscope

AXI Interconnect
- ADI DMA
- AXI
- DDS
- AXI
- DAC Interface
- DAC
- DAC

ADC
- IQ Demod
- ADC
- ADC

FIFO
- User signal chain

User
- space
- applications

Applications
- LINUX

ADC Interface
- FIFO
- User signal chain

蜃
- DAC
- DAC
- DAC

IQ Mod
- AD-FMCOMMS2-EBZ

AVNET
- electronics marketing
## Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th>Demo 1</th>
<th>Demo 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</td>
<td>Zynq / AD9361 SDR Kit in Operation / Base Reference Design</td>
<td>Exploring the Base Reference Design in Xilinx Vivado</td>
</tr>
<tr>
<td>Demo 2</td>
<td></td>
<td>Model-Based Design for Wireless Communications</td>
</tr>
<tr>
<td>Break</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Demo 3</td>
<td>Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
<td>Deploying Simulink models with Xilinx Vivado Design Suite</td>
</tr>
<tr>
<td>Demo 4</td>
<td>802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
<td></td>
</tr>
</tbody>
</table>
# Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Demo 1</strong> Zynq / AD9361 SDR Kit in Operation / Base Reference Design</td>
<td></td>
</tr>
<tr>
<td><strong>Xilinx Vivado® Support for Zynq / AD9361 SDR Kit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Demo 2</strong> Exploring the Base Reference Design in Xilinx Vivado®</td>
<td></td>
</tr>
<tr>
<td><strong>Break</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Model-Based Design for Wireless Communications</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Demo 3</strong> Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 4</strong> 802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
<td></td>
</tr>
</tbody>
</table>
This lecture and demo will guide you through:

- Model-based Design with MATLAB and Simulink
- Spread Spectrum Modulation in Digital Communications
- Simulink Model of IEEE 802.11 Beacon Frame Receiver
Design Flow for IEEE 802.11 Beacon Frame Receiver
Design Flow

- Develop user signal chain (DUT) in Simulink
- Integrate into existing reference design
Challenges of System-Level Design

- Verifying implementation matches algorithm
  - Hand-coded HDL can be error prone and hard to debug
  - Fixed-point implementation may not match floating-point model
  - Test-benches for hardware and software from different teams may not match
Solution: Model-Based Design

- Design, simulate, and validate algorithms and system models in MATLAB and Simulink
- Automatically generate HDL code
- Verify the hardware implementation against the system model
Model-Based Design: From Concept to Production

**RESEARCH**
- Environment Models
- Physical Components
- Algorithms

**REQUIREMENTS**
- MCU
- DSP
- FPGA
- ASIC

**DESIGN**
- C, C++
- VHDL, Verilog
- Structured Text

**IMPLEMENTATION**
- PLC

**TEST & VERIFICATION**
- Generate efficient code
- Explore and optimize implementation tradeoffs
- Model concurrent systems
- Automate regression testing
- Detect design errors
- Support certification and standards

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

The leading environment for technical computing

MATLAB®

SIMULINK®

Toolboxes
(signal, comms, etc.)

Embedded Coder®

HDL Coder™

The leading environment for modeling, simulating, and implementing dynamic and embedded systems
MathWorks Design Package for Software-Defined Radio

- Develop advanced communications systems on Zynq with tools for algorithm development and simulation along with code generation for C and HDL

Algorithm Design and Simulation
- MATLAB
- Simulink
- Fixed-Point Designer™
- Signal Processing Toolbox™
- DSP System Toolbox™
- Communications System Toolbox™

Code Generation for C and HDL
- MATLAB Coder™
- Simulink Coder™
- Embedded Coder®
- HDL Coder™

- Avnet part number AES-ZSDR2-ADI-G-MATW-ANUL
  www.em.avnet.com/adizynqsdr2
Device Under Test / 802.11 Beacon Frame Receiver

IEEE 802.11(R) WLAN
HDL Optimized Beacon Frame Receiver with Captured Data

Samples → Samples → Symbols → HDL Rx → Symbols

Start → DataVid → Start → Frame Buffer → 128

< SIGNAL > → hex 0A → PLCP

< SERVICE > → hex 04 → Detector

< LENGTH > → 2256

[PLCP] → Beacon Flag → [PLCP]

Detector → Beacon Counter → Beacon Number Display

Info

[802.11 Beacon Frame]

PLCP CRC: [ ]
MAC Header [ ]
Frame Body [ ]
Information Elements [ ]

MPDU CRC: [ ]

SSID: MathWorks SDR Team
Supported Rates: 1 2 5.5 11 18 24 36 54
DS Parameter Set: 5
TIM: [ ]
* DTIM Count: 0
* DTIM Period: 2
* Bitmap Control: [ ]
802.11 LAN architecture

- **Basic Service Set (BSS)**
  - wireless hosts
  - access point (AP)
- **Wireless hosts communicate with AP**
- **Mitigating interference and noise**
  - Frequency Hopping Spread Spectrum (FHSS)
  - Infrared (IR)
  - Orthogonal Frequency Division Multiplexing (OFDM)
  - Direct Sequence Spread Spectrum (DSSS)
• DSSS increases the transmit signal bandwidth to $R_c$, far beyond $R_s$ needed to transmit the underlying information
• De-spreading at receiver correlates with signal
• Noise remains uncorrelated (spread) for lower power spectral density in original bandwidth of interest
• Processing gain $G_p = \frac{\text{BW}_{ss}}{\text{BW}_{info}} = \frac{R_c}{R_s}$

*Assuming synchronization of transmitter / receiver spreading sequences
Direct Sequence Spread Spectrum

- Each symbol is multiplied with a pseudo-noise (PN) sequence (aka ‘chip’ sequence)

![Diagram of DSSS PN sequence]

- DSSS PN sequences generated using deterministic algorithms (LFSR) but have properties of random sequences (flipping a coin N times)
  - Equal number of ‘1’ and ‘0’
  - Short run-length
  - Shifting property: if shifted by any nonzero number of elements the resulting sequence has $\frac{1}{2}$ its elements the same as the original sequence and $\frac{1}{2}$ its elements different
11 chip BARKER sequence

- Used for DSSS in 802.11 frame pre-amble
- Good autocorrelation properties
- Minimal sequence allowed by FCC
- Coding gain = $10\log(11) = 10.4$ dB
Direct Sequence Spread Spectrum (DSSS)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_t$</td>
<td>Binary data with symbol rate $R_s = 1/T_s$</td>
</tr>
</tbody>
</table>
| $PN_t$ | Pseudo-noise code with chip rate $R_c = 1/T_c$  
($R_s/R_c = \text{integer}$) |
| $tx_b$ | Binary data $d_t$ multiplied with PN sequence $= d_t \cdot PN_t$ |
Direct Sequence Spread Spectrum (DSSS)

Binary data with symbol rate $R_s = 1/T_s$

Pseudo-noise code with chip rate $R_c = 1/T_c$

$R_s/R_c$ = integer

Binary data $d_t$ multiplied with PN sequence = $d_t$. PN

11-chip Barker sequence

Correlator synchronizer
Direct Sequence Spread Spectrum (DSSS)

\[
d_r = (d_t \cdot PN_t e^{j\Omega_T t} + i(t)) e^{-j\Omega_R t} \cdot PN_r
\]

\[
= d_t \cdot PN_t \cdot PN_r e^{j\Omega_T t} e^{-j\Omega_R t} + i(t) \cdot PN_r e^{-j\Omega_R t}
\]

When RX and TX synchronized \( = 1 \) = \( 0^0 = 1 \) when carriers synchronized

\[
d_r = d_t + i(t) \cdot PN_r e^{-j\Omega_T t}
\]

uncorrelated interference remains spread with low power spectral density
Direct Sequence Spread Spectrum (DSSS)

- At the receiver, the signal correlates but not the interference
**IEEE 802.11 Beacon Frame**

- **Host must *associate* with an AP**
  - scans channels, listening for *beacon frames*
  - contain AP’s service set ID (SSID) & MAC address
  - Host selects AP to associate with
  - May perform authentication
  - Will typically run DHCP to get IP address in AP’s subnet
802.11 Long PLCP Frame Format

- Preamble and Header always at 1Mb/s DBPSK, 11-chip Barker sequence
- SYNC field = 128 one bits (‘1’)
  - Scrambled by scrambler
  - Used for receiver synchronization
## Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Demo 1</strong> Zynq / AD9361 SDR Kit in Operation / Base Reference Design</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 2</strong> Xilinx Vivado® Support for Zynq / AD9361 SDR Kit</td>
<td></td>
</tr>
<tr>
<td><strong>Break</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Model-Based Design for Wireless Communications</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Demo 3</strong> Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 4</strong> Deploying Simulink models with Xilinx Vivado Design Suite</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 4</strong> 802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
<td></td>
</tr>
</tbody>
</table>
# Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Demo 1</strong></td>
<td>Zynq / AD9361 SDR Kit in Operation / Base Reference Design</td>
</tr>
<tr>
<td><strong>Demo 2</strong></td>
<td>Xilinx Vivado® Support for Zynq / AD9361 SDR Kit</td>
</tr>
<tr>
<td>Break</td>
<td></td>
</tr>
<tr>
<td><strong>Demo 3</strong></td>
<td>Exploring the Base Reference Design in Xilinx Vivado®</td>
</tr>
<tr>
<td><strong>Demo 4</strong></td>
<td>Model-Based Design for Wireless Communications</td>
</tr>
<tr>
<td><strong>Demo 3</strong></td>
<td>Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
</tr>
<tr>
<td><strong>Demo 4</strong></td>
<td>Deploying Simulink models with Xilinx Vivado Design Suite</td>
</tr>
<tr>
<td><strong>Demo 4</strong></td>
<td>802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
</tr>
</tbody>
</table>
HDL Simulation of Auto-Generated Code

- Auto-generate code from HDL coder with self-checking testbench
- Create Vivado project and import HDL through Tcl script
- Verify HDL simulation matches generated HDL Coder model (bit-true / cycle-true)
Simulink HDL Coder / Code Generation with Testbench
HDL Simulation in Vivado

HDL simulation matches Simulink generated gm_model

4026.0
Importing IP into Vivado

- Import DUT into project IP catalog
- Connect in Vivado IP Integrator
MathWorks HDL Peripheral with AXI-Lite Interface

- Generate HDL code with AXI-Lite interface
- Includes DUT / algorithm plus AXI-Lite register logic / address decoding for RD/WR
- Integrate within Vivado top-level project
Add HDL Code as AXI-Lite IP in Vivado Design Suite

**IP Packager**
- IPCORE_CLK
- IPCORE_RESETN
- rotor_velocity[17:0]
- phase_current_a[17:0]
- electrical_position[17:0]
- phase_current_b[17:0]
- phase_current_c[17:0]
- AXI_Lite_ARSETN
- AXI_Lite_ACLK

**Vivado Project**

**Vivado IP Integrator**

**Programmable Logic**

**AXI Interconnect**

**AXI-Lite Wrapper**

**Algorithm HDL**

**PL bitstream (.bit)**
Vivado IP Integrator

- Enables reuse to create fully functional IP subsystems

**IP Packager**
- Source (C, RTL, IP)
- Simulation models
- Documentation
- Example Designs
- Test bench

**Standardized IP-XACT**

- Uses multiple plug-and-play forms of IP to implement functional subsystem
- Includes software drivers and API
- Accelerates integration and productivity
Vivado IP Integrator

- **Accelerates hardware design productivity through design reuse**
  - Graphical IP assembly
  - Correct-by-construction
  - System centric
- **Generates IP subsystems**
  - Supports multiple plug-and-play IP formats
  - Generates software drivers and APIs
- **Board and silicon aware**
  - Built in support for Xilinx development baseboards

![Vivado IP Catalog - Standardized IP-XACT](image1)
![Graphical IP assembly](image2)
![Vivado software](image3)
How do I add AXI-Lite HW and SW interfaces
Add C Code in Xilinx Software Development Kit

Xilinx SDK
- PL bitstream
- FSBL + U-Boot
- User Space Application

C Code

Boot.bin
- .ELF Root File System
- DeviceTree (.dtb)
- Linux Kernel

Processing System
- LINUX
- User Application

AXI Interconnect

Boot SD Card
Linux DeviceTree must be updated with PL peripheral info.

User space controls peripheral via UIO driver calls.

Automated build processes are possible with MATLAB and TCL scripting.

---

**AXI Interconnect**

- Algorithm C
- User Space Application
- UIO Driver
- AXI-Lite Wrapper
- Algorithm HDL
The Linux Industrial I/O (IIO) subsystem is intended to provide support for devices that, in some sense, are analog-to-digital or digital-to-analog converters. Devices that fall into this category are:
- ADCs
- DACs
- Accelerometers, gyros, IMUs
- Capacitance-to-Digital converters (CDCs)
- Pressure, temperature, and light sensors, etc.
- RF Transceivers (like the AD9361)

Can be used on ADCs ranging from a 1MSPS SoC ADC to >250 MSPS industrial ADCs. Developed during 2009, committed Jan 2010, moved out of staging Nov 2011, now in all mainline Linux kernels.

The IIO Divers for the motor control solution require the HDL cores to have a specified register map.

A DMA interface is set up for high speed data transfer using multiple multiplexed data channels.
• Each IIO driver has in the device tree an entry related to the actual driver and an entry corresponding to the allocated DMA.

• Each HDL core has a base register map that can be extended to match the desired functionality.

• Having a well established framework allows new devices to be easily added into the system both from a Linux and a HDL perspective.

```c
adi_refadc@79020000 { 
    compatible = "xinlo,axi-adi-adc-adc-1.00-a";
    reg = <0x79020000 0x100000>;
    dma = <0x1 0x0>;
    dma-names = "adi-adc-dma";
};

axidma@40440000 { 
    #address-cells = <0x1>;
    #size-cells = <0x1>;
    #dma-cells = <0x1>;
    compatible = "xinlo,axi-dma";
    reg = <0x40440000 0x100000>;
    linux,phandle = <0xc>;
    phandle = <0xc>;
    dma-channel@40440000 { 
        compatible = "xinlo,axi-dma-s2mm-channel";
        interrupts = <0x0 0x23 0x4>;
        xinlo,datasize = <0x20>;
        xinlo,sg-length-width = <0x17>;
        xinlo,include-dre = <0x0>;
    }
};
```

<table>
<thead>
<tr>
<th>Register name</th>
<th>ADDRESS</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADD_GENERAL</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VERSION</td>
<td>0x00</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>0x04</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>SCRATCH</td>
<td>0x09</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>ADD_COMMON</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REG_SZTN</td>
<td>0x40</td>
<td>RW</td>
<td>Reset all modules</td>
</tr>
<tr>
<td>REG_CTRL</td>
<td>0x44</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>REG_CLK_FREQ</td>
<td>0x54</td>
<td>RO</td>
<td>Read clock frequency for module</td>
</tr>
<tr>
<td>REG_CLK_RATIO</td>
<td>0x58</td>
<td>RO</td>
<td></td>
</tr>
<tr>
<td>REG_STATUS</td>
<td>0x6c</td>
<td>RO</td>
<td>Display errors during conversion</td>
</tr>
<tr>
<td>REG_DMA_COUNT</td>
<td>0x80</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>REG_DMA_STATUS</td>
<td>0x88</td>
<td>RW1C</td>
<td></td>
</tr>
<tr>
<td>REG_DMA_BUSWIDTH</td>
<td>0x8c</td>
<td>RO</td>
<td></td>
</tr>
<tr>
<td>REG_USR_CTRL1</td>
<td>0x90</td>
<td>RW</td>
<td></td>
</tr>
<tr>
<td>ADD_CHANNEL_CURRENT_A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REG_CHAN_CNTL</td>
<td>0x400</td>
<td>RW</td>
<td>Enable channel</td>
</tr>
<tr>
<td>REG_CHAN_STATUS</td>
<td>0x404</td>
<td>RW1C</td>
<td></td>
</tr>
<tr>
<td>REG_CHAN_CNTL1</td>
<td>0x410</td>
<td>RW</td>
<td>N/A</td>
</tr>
<tr>
<td>REG_CHAN_CNTL2</td>
<td>0x414</td>
<td>RW</td>
<td>N/A</td>
</tr>
<tr>
<td>REG_CHAN_USE_CNTL1</td>
<td>0x430</td>
<td>RW</td>
<td>N/A</td>
</tr>
<tr>
<td>REG_CHAN_USE_CNTL2</td>
<td>0x434</td>
<td>RW</td>
<td>N/A</td>
</tr>
<tr>
<td>ADD_CHANNEL_CURRENT_B</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REG_CHAN_CNTL</td>
<td>0x440</td>
<td>RW</td>
<td>Enable channel</td>
</tr>
<tr>
<td>REG_CHAN_STATUS</td>
<td>0x444</td>
<td>RW1C</td>
<td></td>
</tr>
<tr>
<td>REG_CHAN_CNTL1</td>
<td>0x450</td>
<td>RW</td>
<td>N/A</td>
</tr>
<tr>
<td>REG_CHAN_CNTL2</td>
<td>0x454</td>
<td>RW</td>
<td>N/A</td>
</tr>
<tr>
<td>REG_CHAN_USE_CNTL1</td>
<td>0x460</td>
<td>RW</td>
<td>N/A</td>
</tr>
<tr>
<td>REG_CHAN_USE_CNTL2</td>
<td>0x464</td>
<td>RW</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Complete Design Flow for Stand-Alone Deployment

- IP Catalog
- IP Packager
- IP Integrator
  - AXI-Lite Wrapper
    - DUT
    - User IP

- HDL Coder + AXI interface

- Simulink
  - Device Under Test (DUT)

- SDK
  - FSBL
  - Bitstream
  - Uboot.elf
  - BOOT.bin

- Hardware platform + bitstream

- FMCOMMS2 Base Design
  - ANALOG DEVICES

- AVNET electronics marketing
Stand-alone SDR System for Zynq

Processing System
Dual Cortex™-A9
MPCore™
NEON™ / FPU Engine

Programmable Logic

LINUX
User-space applications
LINUX drivers

AXI Interconnect

ADC Interface
FIFO
User signal chain
FIFO

ADI DMA
ADI DMA

DDS

DAC Interface
DAC
DAC

VDMA
HDMI
VDMA

IIO oscilloscope

AVNET
electronics marketing

AD-FMCOMMS2-EBZ
# Agenda

<table>
<thead>
<tr>
<th>Topic</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Introducing the Zynq®-7000 AP SoC / AD9361 SDR Kit</strong></td>
<td></td>
</tr>
<tr>
<td>Demo 1</td>
<td>Zynq SDR Kit in Operation / Base Reference Design</td>
</tr>
<tr>
<td></td>
<td>Analog Devices AD9361 Integrated RF Transceiver</td>
</tr>
<tr>
<td>Break</td>
<td></td>
</tr>
<tr>
<td><strong>Model-Based Design for Wireless Communications</strong></td>
<td></td>
</tr>
<tr>
<td>Demo 3</td>
<td>Simulink® Model of IEEE 802.11 Beacon Frame Receiver</td>
</tr>
<tr>
<td></td>
<td>Deploying Simulink models with Xilinx Vivado Design Suite</td>
</tr>
<tr>
<td>Demo 4</td>
<td>802.11 Beacon Frame Receiver in Stand-Alone Operation</td>
</tr>
</tbody>
</table>
THANK YOU!
Appendix
Reference Links

- AD-FMCOMMS2-EBZ
  

- AD-FMCOMMS2-EBZ HDL Reference Design
  

- AD9361 Integrated RF Agile Transceiver
  
  [http://www.analog.com/AD9361_design_files](http://www.analog.com/AD9361_design_files)

- SD Card Image Files for Zynq + AD9361/64
  

- IEEE 802.11 HDL Optimized Beacon Frame Receiver
  
Xilinx DSP Slice

- Two DSP48E1 slices / tile connected by 5 high-speed interconnects
- Flexible access to dedicated pre-adder
- Pattern detector for efficient rounding hardware
- Independent access to accumulators
- Wide multiplies for greater numerical precision

**Resources per Family**

<table>
<thead>
<tr>
<th></th>
<th>Artix</th>
<th>Kintex</th>
<th>Virtex</th>
<th>Zynq</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max DSP48E1 Fmax</td>
<td>628 MHz</td>
<td>741 MHz</td>
<td>741 MHz</td>
<td>741 MHz</td>
</tr>
<tr>
<td>Max DSP48E1 Count</td>
<td>740</td>
<td>1920</td>
<td>3600</td>
<td>2020</td>
</tr>
</tbody>
</table>