Modeling and Implementing Software-Defined Radio Communication Systems on FPGAs

Puneet Kumar
Senior Team Lead - SPC
Agenda

- Integrated Model-Based Design to Implement SDR on FPGA’s, highlighting:
  - Rapidly develop and verify a baseline transmitter and receiver using library blocks
  - Automatically generate HDL code and integrate the code with target hardware
  - Verify the design using HDL cosimulation and FPGA-in-the-loop on an FPGA Evaluation Kit

- Q&A
Software Defined Radio - Demo
Beacon Frame Receiver – SDR Demo
Software Defined Radio

Data Source → Data Compression → Error Correction Encoding → Digital Modulation → Digital to Analog Converter → RF Frontend

Data Sink → Data Expansion → Error Correction Decoding → Digital Demodulation → Equalization, Timing and Synchronization → Analog to Digital Converter → RF Frontend

Software Implementation

Hardware Implementation
Software Defined Radio

Software or Programmable Logic

Software Implementation

Data Source → Data Compression → Error Correction Encoding → Digital Modulation

Data Sink → Data Expansion → Error Correction Decoding → Digital Demodulation

Hardware Implementation

Digital to Analog Converter → RF Frontend

Air Channel

Analog to Digital Converter → RF Frontend
Software Defined Radio

Data Source → Data Compression → Error Correction Encoding → Digital Modulation

Data Sink → Data Expansion → Error Correction Decoding → Digital Demodulation → Equalization, Timing and Synchronization → Analog to Digital Converter → RF Frontend

Digital to Analog Converter → RF Frontend → Air Channel → Analog to Digital Converter → RF Frontend

Software Implementation

Hardware Implementation
### Some SDR systems (various vendors)

<table>
<thead>
<tr>
<th></th>
<th>USRP</th>
<th>USRP2</th>
<th>BEE2</th>
<th>KUAR</th>
<th>LYTRECH</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RF bandwidth (MHz)</strong></td>
<td>16</td>
<td>50</td>
<td>25</td>
<td>30</td>
<td>5, 7, 20 or 22</td>
</tr>
<tr>
<td><strong>Processing partition</strong></td>
<td>Off-board</td>
<td>Mixed</td>
<td>On-board</td>
<td>On-board</td>
<td>On-board</td>
</tr>
<tr>
<td><strong>Processing architecture</strong></td>
<td>GPP</td>
<td>GPP FPGA</td>
<td>FPGA</td>
<td>GPP FPGA</td>
<td>GPP FPGA</td>
</tr>
<tr>
<td><strong>Connectivity</strong></td>
<td>USB2</td>
<td>GigEthernet</td>
<td>USB</td>
<td>USB</td>
<td>Ethernet</td>
</tr>
<tr>
<td><strong>No. of antennas or RF paths</strong></td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td><strong>Cost</strong></td>
<td>$700.00</td>
<td>$1400.00</td>
<td>$20,000.00</td>
<td>NFS</td>
<td>$9900.00</td>
</tr>
<tr>
<td><strong>Strengths</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Weaknesses</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. 

2. $20,000.00

3. NFS
Implementing a Wireless Receiver on an FPGA

Build a Baseline QPSK Model
- Create Simulink executable model to explore design choices and determine baseline performance
- Use model to generate wireless test signal

Elaborate and Prototype the Design
- Designing the carrier and timing recovery loops for a QPSK receiver
- Verify the design with the wireless test signal

Implement on Target Hardware
- Convert the QPSK design to an implementation level model
- Generate HDL code and integrate with target hardware
Implementing a Wireless Receiver on an FPGA

Build a Baseline QPSK Model
- Create Simulink executable model to explore design choices and determine baseline performance
- Use model to generate wireless test signal

Elaborate and Prototype the Design
- Designing the carrier and timing recovery loops for a QPSK receiver
- Verify the design with the wireless test signal

Implement on Target Hardware
- Convert the QPSK design to an implementation level model
- Generate HDL code and integrate with target hardware
Build a Baseline QPSK Model

Steps to follow

1. Get the Design Specification
2. Build the Executable Testbench
3. Verify the Model
Design Specifications

Modulation: QPSK
Symbol Rate: 195 kilo-sym/sec
RRCOS Filter
  Stopband Attenuation: 30 dB
  Rolloff: 0.25;
  Interp/Decim Factor: 8
Carrier Frequency: 432 MHz
Communications System Toolbox

The Communications System Toolbox extends Simulink to design and simulate the physical layer of communication systems and components.

- Modulation
  - AM, PM, FM, CPM, TCM
    - Hard Decision, Log Likelihood
- Forward Error Correction
  - Reed Solomon, Convolutional
- RF Impairments
- Synchronization, Equalization
Build the Executable Test Bench

Simulink Library Browser

Simulink Executable Model

Visualization
Verify the Model with AWGN Channel and BER Analysis

BERTool

Establish Baseline

Simulated Results vs Theory
Verify the Model

Generate Wireless Test Signal using MATLAB/Simulink SDR Interface

Communications Model

MATLAB/Simulink SDR Interface

Gigabit Ethernet

USRP2

E100

N210

MAT file

Spectrum Analyzer
Implementing a Wireless Receiver on an FPGA

Build a Baseline QPSK Model
- Create Simulink executable model to explore design choices and determine baseline performance
- Use model to generate wireless test signal

Elaborate and Prototype the Design
- Designing the carrier and timing recovery loops for a QPSK receiver
- Verify the design with the wireless test signal

Implement on Target Hardware
- Convert the QPSK design to an implementation level model
- Generate HDL code and integrate with target hardware
Elaborate and Prototype the Design

Steps to follow

1. Add more realistic impairments to the Basic QPSK Model
2. Design mitigation algorithms
3. Test design with real captured data
Wireless Channel Impairments

- Additive thermal noise
- Multipath fading
  - Not a concern here due to close-range line-of-sight communication
- Synchronization
  - Carrier phase/frequency offset
  - Symbol timing offset
  - Frame boundaries
Effect of Carrier Frequency Offset

Function Block Parameters: Phase/ Frequency Offset

- Phase/Frequency Offset (mask) (link)
  - Apply a frequency and phase offset to the input signal.

Parameters

- Phase offset (deg):
  - -20

- Frequency offset from port

- Frequency offset (Hz):
  - -40
Effect of Carrier Frequency Offset

Phase/Frequency Offset (mask) (link)
Apply a frequency and phase offset to the input signal.

Parameters
Phase offset (deg):
-20

Frequency offset from port
Frequency offset (Hz):
0

Scatter Plot

- Quadrature Amplitude
- In-phase Amplitude
Effect of Carrier Frequency Offset

- Phase offset (deg): -20
- Frequency offset (Hz): -40
Carrier Synchronization

\[ e^{j(\Delta \omega_0 n - \theta)} \]

Phase Error Detector

\[ \text{arg}(.) \]

Loop Filter

\[ K1 \]

\[ K2 \]

\[ Z^{-1} \]

NCO

\[ e^{-j(\Delta \omega_0 n - \hat{\theta})} \]

Implementing a Wireless Receiver on an FPGA

Build a Baseline QPSK Model
- Create Simulink executable model to explore design choices and determine baseline performance
- Use model to generate wireless test signal

Elaborate and Prototype the Design
- Designing the carrier and timing recovery loops for a QPSK receiver
- Verify the design with the wireless test signal

Implement on Target Hardware
- Convert the QPSK design to an implementation level model
- Generate HDL code and integrate with target hardware
Algorithm Development Process

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

Requirements

Research & Design

- Design
- Test
- Elaborate

Implementation

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

Embedded
- C
- VHDL / Verilog
- Structured Text
Solution: C and HDL Code Generation

- Design, execute, and verify algorithms in MATLAB
- Automatically generate C or HDL code
- Deploy generated code on hardware
Targeting HDL with an SDR Platform – Workflow Demo
Model-Based Design flow using Simulink
from Algorithm to FPGA Implementation

MATLAB® and Simulink®
Algorithm and System Design

HDL Coder
RTL Creation

HDL Verifier
HDL Co-Simulation

RTL

Implement Design
Synthesis
Map
Place & Route

Verification
Functional Simulation
Static Timing Analysis
Timing Simulation

HDL Verifier
FPGA in the Loop

Back Annotation

Design
Algorithm Development
MATLAB
Simulink
Stateflow
Thank You!