Efficient System-Level DSP Design Flow with WiMAX DUC and DDC Case Study
Agenda

- FPGA in digital signal processing (DSP) applications overview
- System-level design tools for Altera® FPGAs
  - Overview
  - Introduction to DSP Builder
  - DSP Builder design flow walkthrough
- Case study: WiMAX digital upconverter (DUC) and digital downconverter (DDC) design
FPGA in DSP Applications

- Automotive
- Communications
- Consumer
- Industrial
- Test and Measurement
- Military
- Broadcast
- Medical
DSP Design Flow in FPGA

Area of Innovation

Traditional Focus of FPGA Tools
Altera System-Level Design Tools

IP Integration

Software Development

DSP Algorithm Development

System Integration
What is DSP Builder?

- Interface between Quartus II and MATLAB/Simulink
- Library add-on to Simulink
- Altera blockset
  - Library of fixed-point Simulink functions
  - Uses double precision
- Altera DSP IP
  - OpenCore Plus
- Signal Compiler utility
  - Converts between Simulink and Altera domain
- Hardware debug
  - Hardware in the loop (HIL)/SignalTap® logic analyzer
Traditional System Design Tool Flow

Development
System-Level Simulation of Algorithm Model
Algorithm Modeling (C/C++, M, MDL)
MATLAB/Simulink

Implementation
RTL Implementation
RTL Simulation
Synthesis Place & Route Simulation (VHDL/Verilog)
Precision, Synplify Quartus II, ModelSim

Verification
System-Level Verification of Hardware Implementation
System-Level Verification (POF)
Hardware

System Algorithm Design and FPGA Design Separated
DSP Builder - Simulink Design Flow

**Development**
- System-Level Simulation of Algorithm Model
- MATLAB/Simulink

**Implementation**
- RTL Implementation
- RTL Simulation
- LeonardoSpectrum Precision, Synplify Precision, Quartus II, ModelSim

**Verification**
- System-Level Verification of Hardware Implementation
- Synthesis, Place & Route
- Simulation
- System-Level Verification (POF)

*System Algorithm Design and FPGA Design Integrated*
DSP Builder Features

- Automatic generation of VHDL design from a MATLAB/Simulink representation
- Automatic generation of VHDL testbench
  - Captures stimulus from Simulink, writes testbench
- HDL import
  - Reads in design: Verilog or VHDL, or Quartus II project
  - Creates Simulink simulation model
- SignalTap embedded logic analyzer
  - Captures internal data and it into MATLAB
- HIL testing
  - Pass vectors to/from board
DSP Builder Benefits

For hardware engineer
- Extends RTL analysis and debug capabilities to system-level tool
  - Access to MATLAB data formatting
  - Access to a large library of Simulink models
- Speeds up simulation run time
- Enables IP evaluation at system level

For system-level engineer
- Allows rapid prototyping with minimal PLD expertise
- Provides easy access to hardware evaluation
- Extends floating-point to fixed-point system analysis
DSP Builder Design Flow

Matlab/Simulink Domain
(C+ System Analysis)

VHDL Domain
(Implementation/Simulation)
Design Flow Overview

1) Create design in Simulink using Altera libraries
2) Simulate in Simulink
3) Add SignalCompiler to model
4) Create HDL code and generate testbench
5) Perform RTL simulation
6) Synthesize HDL code and place and route
7) Program device
8) Verify hardware: SignalTap logic analyzer/HIL
Altera DSP Builder Libraries

- AltLab
- Arithmetic
- Boards
- Complex type
- Gate and control
- I/O and bus
- Rate change
- SOPC Builder links
- State machine functions
- Storage
- MegaCore functions
Step 1: Create Design in Simulink Using Altera Libraries

- Drag and drop library blocks into Simulink design and parameterize each block
Parameterization of IP MegaCore Functions
Step 2: Simulate in Simulink

[Image of Simulink simulation software interface with graphs and charts displaying waveforms and power spectral density.]
Step 3: Add SignalCompiler to Model to Generate HDL Code

- Stratix and Stratix II
- Stratix GX
- Cyclone & Cyclone II
- ACEX® 1K
- Mercury™
- FLEX® 10K and FLEX 6000
- Development Boards
- APEX™ 20K/E/C
- APEX II

- LeonardoSpectrum™
- Synplify
- Precision
- Quartus II

- Speed
- Area
- Balanced
- Fast fit – no timing optimization
- Use current Quartus II project

Testbench generation

Message window
Step 4: Create HDL Code and Generate Testbench

FilteringLab.vhd

FilteringLab.mdl

Enable “Generate Stimuli for VHDL Testbench” Button

FilteringLab.vhd
HDL Code Generation

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_signed.all;

library dspbuilder;
use dspbuilder.dspbuilderblock.all;
library ipx:
use ipx.ipx_components.all;

Entity FilteringLab is
Port(
  clock : in std_logic;
  scil : in std_logic_vector(1 downto 0);
  scun : in std_logic_vector(2 downto 0);
  clk_outl : out std_logic;
  olEDls : out std_logic;
  ofr_data : out std_logic_vector(11 downto 0);
  ofl : out std_logic_vector(11 downto 0);
);

end FilteringLab;

architecture dspbuilder of FilteringLab is

signal Sate : std_logic_vector(11 downto 0);
signal scil : std_logic;

begin

-- Using FLL to drive pin Y3 (J34 clock source)
component boardS12S_pill PORT(
  inclk : in std_logic;
  oclk : out std_logic;
);
end component;

signal ad : std_logic_vector(7 downto 0);
signal al : std_logic;
signal aw : std_logic;
signal ax : std_logic_vector(2 downto 0);
signal ay : std_logic;
signal a2 : std_logic_vector(2 downto 0);
signal a3 : std_logic_vector(2 downto 0);
signal a4 : std_logic_vector(2 downto 0);
signal a5 : std_logic_vector(2 downto 0);
signal a6 : std_logic_vector(2 downto 0);
signal a7 : std_logic_vector(2 downto 0);
signal a8 : std_logic_vector(2 downto 0);
signal a9 : std_logic_vector(2 downto 0);
signal a10 : std_logic_vector(2 downto 0);
signal a11 : std_logic_vector(2 downto 0);
signal a12 : std_logic_vector(2 downto 0);
signal a13 : std_logic_vector(2 downto 0);
signal a14 : std_logic_vector(2 downto 0);
signal a15 : std_logic_vector(2 downto 0);
signal a16 : std_logic_vector(2 downto 0);
signal a17 : std_logic_vector(2 downto 0);
signal a18 : std_logic_vector(2 downto 0);
signal ExtExtract4 : std_logic_vector(7 downto 0);

-- SubSystem Hierarchy - Simlink Block "LowPass_Serial_Filter" component LowPass_Serial_Filter port:

  clock : in std_logic;
  scil : in std_logic;
  lInputData : in std_logic_vector(11 downto 0);
  lStarts : in std_logic;
  oflResults : out std_logic_vector(17 downto 0);
);

end component;

-- SubSystem Hierarchy - Simlink Block "SineWave_Generator" component SineWave_Generator port:

  clock : in std_logic;
  scil : in std_logic;
  lStarts : in std_logic;
  osin_833_33kHz : out std_logic_vector(12 downto 0);
  osin_00_33kHz : out std_logic_vector(12 downto 0);
);

end component;

begin

assert (<=0) report alversion severity Note;

-- Output - I/O assignment from Simlink Block "LEAD"
olEDls <= a19;
ofr_data <= Sate_data;

-- Output - I/O assignment from Simlink Block "trir_result"
ofl_results <= a16;
olT <= scil;

-- Input - I/O assignment from Simlink Block "isW3s"
aw <= a15;

-- Bit Extraction - Simlink Block "ExtractInt"
ExtExtract4 <= aw;
aw1 <= ExtExtract4(1);
scil <= aw1 or scil;
scil <= aw1 or scil;

-- Simlink Block "VCC"
aw <= '1';

-- Simlink Block "GND"
aw <= '0';

-- Simlink Block "GND"
DSP Builder Report File

- Lists all converted blocks
  - Port widths
  - Sampling frequencies
  - Warnings and messages
Step 5: Perform RTL Simulation (ModelSim)

1) Set working directory (**File** => **Change Directory**)

2) Run TCL file (**Tools** => **Execute Macro**...)

```
ModelSim ALTERA 5.6a - Custom Altera Version
```
Perform Verification

<table>
<thead>
<tr>
<th>wave - default</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
</tr>
<tr>
<td>------</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>clock</td>
</tr>
<tr>
<td>reset</td>
</tr>
<tr>
<td>/tb_fir32/stts</td>
</tr>
<tr>
<td>/tb_fir32/linput</td>
</tr>
<tr>
<td>/tb_fir32/binput</td>
</tr>
<tr>
<td>/tb_fir32/sfruits</td>
</tr>
</tbody>
</table>

ModelSim vs. Simulink
Step 6: Synthesize HDL and Place and Route

- Leonardo Spectrum
- Synplify
- Quartus II

• Synthesis
  - Leonardo Spectrum
  - Synplify
  - Quartus II

• Quartus II Fitter
  - Synthesis
Step 7: Program Device

Download Design to DSP Development Kits
Stratix II DSP Development Board

- VGA Connector
- 12-Bit, 125-MHz A/D
- 14-Bit, 165-MHz D/A
- 16-Bit Audio Codec
- 16-Mbyte SDR SDRAM
- 9-Pin RS232 Connector
- 10/100 Ethernet
- Texas Instruments DSP Connector on Underside of Board

Available with EP2S60 or EP2S180 Device
Cyclone II DSP Development Board

- Power Supply Connector
- Stereo Codec
- VGA Connector
- 14-Bit 165-MHz DAC
- 12-Bit, 125-MHz ADC
- 256-Mbyte DDR2 DIMM
- TI EMIF Connector on Underside of Board
- Expansion Prototype Connector
- 1-Mbyte Synchronous SRAM

Now Available with EP2C70 Device
Step 8: SignalTap II Logic Analyzer

- Embedded logic analyzer (ELA)
  - Downloads into device with design
  - Captures state of internal nodes
  - Uses JTAG for communication
SignalTap II Logic Analyzer

Imported Data

Imported Plot

Analysis of Imported Data
Hardware in Loop (HIL)
Design Flow Review

1) Create design in Simulink using Altera libraries
2) Simulate in Simulink
3) Add SignalCompiler to model
4) Create HDL code and generate testbench
5) Perform RTL simulation
6) Synthesize HDL code and place and route
7) Program device
8) Verify hardware: SignalTap logic analyzer HIL
Altera DSP Development Kits

- Development Kit
- 30-Day Evaluation Version
- System Reference Designs
WiMAX DUC and DDC Design Case Study
Base Station Architecture Overview

PLD Applications

- PA
- LNA

A/D, D/A, Digital I/F, DUC/DDC, Glue Logic, RF Card

Baseband Processing (W-CDMA/CDMA2000/WiMAX), Glue Logic, Channel Card


Host CPU, Control Logic, Glue Logic, Control Card

Clock Generator, GPS Receiver, Timing Card, Switch Interface, Switch Card

IP/ATM Interface, Glue Logic, BSC/RNC Interface
Reference Design Overview

- **DUC/DDC**
  - Provides the link between digital baseband and analog RF front end of generic transceiver
  - High throughput signal processing required makes FPGA ideal platform
WiMAX DUC and DDC Designs

- Compliant to the draft WiMAX standard (IEEE 802.16)
- Multi-channel filter design for low cost
- Support for multiple transmit and receive antenna configurations
- Easily modifiable to support scalable channel bandwidths
- Uses DSP Builder methodology
- Backed up by DSP Builder-ready, highly parameterizable IP MegaCore functions
DUC and DDC High-Level Block Diagrams

From Baseband Modem

FIR → CIC → NCO → Σ → CFR → DPD → D/A

Q → FIR → CIC → NCO → CIC → FIR → Resampler

A/D

DUC and DDC: Wireless, Military, Medical, Broadcast

Crest-Factor Reduction (CFR) and Digital Predistortion (DPD): Wireless

© 2006 Altera Corporation
DSP Builder Implementation: IP MegaCore Library

IP Can Be Added to the Library Separately
DSP Builder Implementation: Digital Intermediate Frequency (IF) Library

- **Adapters**
  - Provide input/output interface to finite impulse response (FIR) filter

- **Multichannel**
  - Frame format converter
  - Decimation
  - Interpolation
  - Multiplexer
  - Demultiplexer

- **Rounding**
DUC With 2 Antennas Design Architecture

Timeshare DUC Hardware Between Antennas
DSP Builder Implementation: DUC Example Design With 2 Antennas

Use FIR Compiler IP

Use Numerically Controlled Oscillator (NCO) Compiler IP
Simulation

Cascaded Filter Output

After Upconversion
Convert to VHDL: SignalCompiler
DDC With 4 Antennas Design Architecture

oversample

oversample

oversample

oversample

NCO

FIR ↓4

FIR ↓4

FIR ↓4

FIR ↓4

FIR ↓2

FIR

I_1

Q_1

I_2

Q_2

I_3

Q_3

I_4

Q_4

91.392 MHz

182.784 MHz

45.696 MHz

182.784 MHz

91.392 MHz

11.424 MHz

91.392 MHz

182.784 MHz

45.696 MHz

182.784 MHz

91.392 MHz

11.424 MHz
DSP Builder Implementation: DDC Example Design With 4 Antennas

- **Simulate button**
- **SignalCompiler (convert to HDL)**
- **Analyze output data**

- **Input stimulus loaded as part of model Initialization function**
2xRx DDC Architecture

Remove Redundant Filter Chains

- oversample
- oversample
- oversample
- oversample

Modify Number of Channels

Change Input/Output Frame Formats

- I_1
- Q_1
- I_2
- Q_2
- I_3
- Q_3
- I_4

Change Input/Output Frame Formats

30 minutes!
# DUC and DDC Synthesis Results

<table>
<thead>
<tr>
<th>ALUTs</th>
<th>M512</th>
<th>M4K</th>
<th>MRAM</th>
<th>Multipliers 18x18</th>
<th>$f_{\text{max}}$ MHz</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DUC Time Multiplexed IQ Design</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,113</td>
<td>21</td>
<td>23</td>
<td>0</td>
<td>30</td>
<td>281</td>
</tr>
<tr>
<td><strong>DUC 2 Antenna Design</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4,229</td>
<td>21</td>
<td>56</td>
<td>0</td>
<td>55</td>
<td>193</td>
</tr>
<tr>
<td><strong>DDC Time Multiplexed IQ Design</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2,488</td>
<td>19</td>
<td>22</td>
<td>0</td>
<td>25</td>
<td>293</td>
</tr>
<tr>
<td><strong>DDC 4 Antenna Design</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10,753</td>
<td>67</td>
<td>69</td>
<td>0</td>
<td>74</td>
<td>201</td>
</tr>
</tbody>
</table>

**Highly Optimized and Cost Efficient Designs!**

*More Information at www.altera.com*
Summary

- DSP Builder tool improves productivity
  - System-level DSP design and FPGA design integrated into one platform: Simulink

- WiMAX DUC and DDC application example
  - DSP Builder-based IQ time multiplexed and multi-antenna designs
    - Use FIR compiler and NCO compiler IP
    - Design methodology significantly reduces the development time for different standards
    - Highly optimized and cost-efficient designs
Thank You
Q & A