DSP Design Using MATLAB and Simulink with Xilinx Targeted Design Platform

MathWorks and Xilinx joint Seminar

Daniele Bagni
XILINX DSP Specialist for EMEA
(daniele.bagni@xilinx.com)

15 Sept. 2011
Outlines

- Xilinx corporate
- Virtex-6 / Spartan-6 family overview
- 7-series: a new family
- Digital Signal Processing on FPGAs
- System Generator for DSP overview
- High Level Synthesis from C: AutoESL
- Demos with HW-SW Co-Simulation (HW in the Loop)
- Conclusion
Outlines

- **Xilinx corporate**
  - Virtex-6 / Spartan-6 family overview
  - 7-series: a new family
  - Digital Signal Processing on FPGAs
  - System Generator for DSP overview
  - High Level Synthesis from C: AutoESL
  - Demos with HW-SW Co-Simulation (HW in the Loop)
  - Conclusion
Xilinx at a Glance

- **Worldwide leader in programmable solutions**
  - Founded in 1984
  - $2.3B in revenues in FY ’10
  - ~3,100 employees worldwide
    - 1,300 in San Jose
  - 20,000+ customers worldwide
  - Pioneer of the fabless model
  - Inventor of the FPGA

- **50% PLD market segment share**
  - Larger than all competitors combined

- **Diversified customers and markets**

- **Excellent financial scorecard**
Xilinx Serves a Wide Range of Markets

<table>
<thead>
<tr>
<th>Category</th>
<th>Markets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communications</td>
<td>Infrastructure, Wireless</td>
</tr>
<tr>
<td>Automotive</td>
<td>Infotainment, Instrumentation</td>
</tr>
<tr>
<td>Aerospace and Defense</td>
<td>Avionics, Space</td>
</tr>
<tr>
<td>Consumer</td>
<td>Displays, Handhelds</td>
</tr>
<tr>
<td>Industrial Scientific and Medical</td>
<td>Video imaging, Test and measurement</td>
</tr>
</tbody>
</table>
Outlines

- Xilinx corporate
- **Virtex-6 / Spartan-6 family overview**
  - 7-series: a new family
  - Digital Signal Processing on FPGAs
  - System Generator for DSP overview
  - High Level Synthesis from C: AutoESL
  - Demos with HW-SW Co-Simulation (HW in the Loop)
- Conclusion
Virtex-6 / Spartan-6 family overview
Customers Requested

- **Lower power**
  - The world is going green

- **Higher system performance**
  - Standards are getting faster

- **Lower system cost**
  - The market is getting more competitive

- **Ease-of-Use / Ease-of-Design**
  - Faster time-to-market, shorter product lifetime
Virtex-6 and Spartan-6 FPGA
Efficient Hard IP Blocks

- **More efficient than soft solution**
  - Higher performance
  - Lower power
  - Smaller size / lower cost

- **Carefully chosen benefits**
  - Memory controller, system monitor, TEMAC, PCIe, FIFO controller

- **Carefully designed to maintain flexibility**
  - Customizable through user-defined parameters

- **Documented, verified, and guaranteed performance**
  - Lower risk and shorter design time

FPGAs are becoming Systems-On-a-Chip
## Spartan-6 and Virtex-6 Overview

<table>
<thead>
<tr>
<th></th>
<th>Spartan-6</th>
<th>Virtex-6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Cells</td>
<td>4K → 150K</td>
<td>75K → 760K</td>
</tr>
<tr>
<td>LUT6</td>
<td>2.5K → 92K</td>
<td>47K → 474K</td>
</tr>
<tr>
<td>FF</td>
<td>5K → 184K</td>
<td>93K → 948K</td>
</tr>
<tr>
<td>BRAM (kbits)</td>
<td>216 Kb → 4.8 Mb</td>
<td>5.5 Mb → 38.3 Mb</td>
</tr>
<tr>
<td>DSP48</td>
<td>8 → 180</td>
<td>288 → 2016</td>
</tr>
<tr>
<td>DSP48 FMax</td>
<td>283 MHz</td>
<td>600MHz</td>
</tr>
<tr>
<td>Processing Performance</td>
<td>51 GMAC</td>
<td>1210 GMAC</td>
</tr>
</tbody>
</table>
Outlines

- Xilinx corporate
- Virtex-6 / Spartan-6 family overview
  - 7-series: a new family
    - Digital Signal Processing on FPGAs
    - System Generator for DSP overview
    - High Level Synthesis from C: AutoESL
    - Demos with HW-SW Co-Simulation (HW in the Loop)
- Conclusion
7-series family overview
Xilinx 7 Series
Previous Generation Comparison

**ARTIX™7**
- Lowest Power and Cost

**KINTEX™7**
- Industry’s Best Price / Performance “New Class of FPGA”

**VIRTEX™7**
- Industry’s Highest System Performance and Capacity

**Compared to Spartan-6**
- 2.4x larger
- 30% more performance
- 35% lower cost
- 50% less power
- 50% smaller footprint

**Compared to Virtex-6**
- Comparable performance
- 50% lower cost
- 50% less power

**Compared to Spartan-6**
- 3.3x larger
- Over 2x performance with 4x transceiver speed
- Better Price / Performance

**Compared to Virtex-6**
- 2.5x larger (2M LCs)
- 50% lower power
- 2x line rate (28Gbps with 2.8Tbps serial bandwidth)
## 7 Series

**Breakthrough Power, Performance & Productivity**

<table>
<thead>
<tr>
<th>Maximum Capability</th>
<th>ARTIX&lt;sup&gt;7&lt;/sup&gt;</th>
<th>KINTEX&lt;sup&gt;7&lt;/sup&gt;</th>
<th>VIRTEX&lt;sup&gt;7&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lowest Power and Cost</td>
<td>8K – 350K</td>
<td>70K – 480K</td>
<td>285K – 2,000K</td>
</tr>
<tr>
<td>Logic Cell Range</td>
<td>18 Mb</td>
<td>34 Mb</td>
<td>85 Mb</td>
</tr>
<tr>
<td>Block RAM</td>
<td>700</td>
<td>1,920</td>
<td>5,280</td>
</tr>
<tr>
<td>DSP Slices</td>
<td>375 GMACS</td>
<td>1,225 GMACs</td>
<td>3,368 GMACs</td>
</tr>
<tr>
<td>Peak DSP Perf.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Transceivers</td>
<td>4</td>
<td>32</td>
<td>96</td>
</tr>
<tr>
<td>Transceiver Performance</td>
<td>6.6 Gbps</td>
<td>12.5 Gbps</td>
<td>13.1 Gbps</td>
</tr>
<tr>
<td>Memory Performance</td>
<td>1,066 Mbps</td>
<td>2,133 Mbps</td>
<td>2,133 Mbps</td>
</tr>
<tr>
<td>I/O Pins</td>
<td>450</td>
<td>500</td>
<td>1,200</td>
</tr>
<tr>
<td>I/O Voltages</td>
<td>3.3V and below</td>
<td>3.3V and below</td>
<td>3.3V and below</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1.8V and below</td>
<td>1.8V and below</td>
</tr>
</tbody>
</table>
Zynq-7000 EPP Family Highlights

- Complete ARM Processing System
  - Dual ARM® Cortex™-A9, Processor Centric
  - Integrated Memory Controllers & Peripherals
  - Fully autonomous to the Programmable Logic

- Tightly Integrated Programmable Logic
  - Extends Processing System
  - Scalable density and performance
  - Over 3000 Internal Interconnects

- Flexible Array of I/O
  - Wide Range of external Multi Standard I/O
  - High Performance integrated serial tranceivers
  - Analog-to-Digital Converter inputs
Zynq-7000 ARM Processing System

**Processor Core Complex**
- Dual ARM® Cortex™-A9 MPCore™ with NEON™ extensions
- Single / Double Precision Floating Point support
- Up to 800 MHz Operation

**High BW Memory**
- Internal
  - L1 Cache – 32KB/32KB (per Core)
  - L2 Cache – 512KB Unified
- On-Chip Memory of 256KB
- Integrated Memory Controllers (DDR2, DDR3, LPDDR2, 2xQSPI, NOR, NAND Flash)

**Integrated Memory Mapped Peripherals**
- 8 DMA Channels
- 2x USB 2.0 (OTG) w/DMA
- 2x Tri-mode Gigabit Ethernet w/DMA
- 2x SD/SDIO w/DMA, 2x UART, 2x CAN 2.0B, 2x I2C, 2x SPI, 32b GPIO

**Open Standard Interconnect Enabled by AXI**
- High Bandwidth Interconnect between Processing System and Programmable Logic
- ACP port for enhanced Hardware Acceleration and cache coherency for additional Soft processors

Processing System Ready to Program
Tightly Integrated Programmable Logic

Built with State-of-the-art 7 Series Programmable Logic

- 28K-235K logic cells
- 430K-3.5M equivalent ASIC gates

Note: ASIC equivalent gates based on analysis over broad range of designs

Over 3000 internal Interconnects

- Up to 100Gb of BW
- Memory-mapped interface

Integrated ADCs

- Dual multi channel 12-bit A/D converter
- Up to 1Msps

Enables Massive Parallel Processing

- Up to 760 DSP blocks delivering over 480GMACs

Scalable Density and Performance
Outlines

- Xilinx corporate
- Virtex-6 / Spartan-6 family overview
- 7-series: a new family
- **Digital Signal Processing on FPGAs**
  - System Generator for DSP overview
  - High Level Synthesis from C: AutoESL
  - Demos with HW-SW Co-Simulation (HW in the Loop)
- Conclusion
DSP on FPGAs
Delivering DSP Performance through Parallelism

Standard DSP Processor – Sequential (Generic DSP)

- Data In
- Coefficients
- Single-MAC Unit
- Data Out

200 clock cycles needed

1.2 GHz

\[
\frac{200 \text{ clock cycles}}{1 \text{ clock cycle}} = 6 \text{ MSPS}
\]

FPGA - Fully Parallel Implementation (Virtex-6/7-Series FPGA)

- Data In
- Reg
- X
- C0
- C1
- C2
- C3
- \ldots
- C199
- Reg
- Single-MAC Unit
- Data Out

200 operations in 1 clock cycle

600 MHz

\[
\frac{1 \text{ clock cycle}}{1 \text{ clock cycle}} = 600 \text{ MSPS}
\]
Bridging The DSP Performance Gap

DSP Performance
- 3D Medical Imaging
- Wireless Base Stations
- HD Audio/Video Broadcast
- Radar & Sonar
- HD Video Surveillance
- Mobile Software Defined Radio
- MIMO

DSP Cost / Performance
- Portable Ultrasound
- Pico/Femto Base Stations
- Consumer Video
- HD Video Surveillance
- Mobile Software Defined Radio
- Automotive Driver Assist

Source: Forward Concepts
## Delivering DSP Performance through DSP48 slice

<table>
<thead>
<tr>
<th></th>
<th><strong>DSP48E1</strong></th>
<th><strong>DSP48A1</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Optimized for</strong></td>
<td>Performance</td>
<td>Cost / Performance</td>
</tr>
<tr>
<td><strong>Clock Speed</strong></td>
<td>600 MHz</td>
<td>278 MHz</td>
</tr>
<tr>
<td><strong>Pre-Adder</strong></td>
<td>25 bits</td>
<td>18 bits</td>
</tr>
<tr>
<td><strong>Multiplier</strong></td>
<td>25x18</td>
<td>18x18</td>
</tr>
<tr>
<td><strong>ALU Functions</strong></td>
<td>Post Add</td>
<td></td>
</tr>
<tr>
<td><strong>Pattern Matching</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How the New Pre-Adder is Used

Example: 8-tap Even Symmetric Systolic FIR

Using the pre-adder, it reduces the usage of DSP48 slices from 8 down to 4!

Copyright 2009 Xilinx
Virtex-6 DSP48E1 Block Diagram

*These signals are dedicated routing paths internal to the DSP48E1 column. They are not accessible via fabric routing resources.
Outlines

- Xilinx corporate
- Virtex-6 / Spartan-6 family overview
- 7-series: a new family

**Digital Signal Processing on FPGAs**
- System Generator for DSP overview
- High Level Synthesis from C: AutoESL
- Demos with HW-SW Co-Simulation (HW in the Loop)
- Conclusion
System Generator for DSP
DSP Design Flow Process

Algorithm Development, Simulation & Modeling

Translation & Code Gen to HDL

HDL Simulation

Synthesis

Implementation & Verification

MathWorks

MATLAB® Simulink®

Simulink HDL Coder

ModelSim® (Mentor)

Synplify Pro® (Synopsys)

Xilinx

ISE®

XST®

ISE Design Suite

Third-party
Algorithm capture, exploration, simulation, and implementation environment based on Simulink.

- Implementation leverages optimized Xilinx IP
- Automatic generation of fixed-point RTL
  - Includes saturation and rounding logic
- Custom RTL integration (hand-written or automatically generated by Simulink HDL Coder for example)
- Automated verification flows
Automated Verification

HDL test benches can automatically be generated using Simulink test vectors.

System Generator leverages the power of the Simulink algorithmic verification environment.
SysGen usage

Gateway In block
- Double precision input data is quantized into Fixed Point representation
- After netlist generation, just an input port

Gateway Out blocks
- Convert the fixed point representation into Simulink floating point
- Used to define the output data ports of the HDL design.

Allows you to define what kind of generation you want
- Netlist or bitstream
- Hardware in the loop
- Export as hardware peripheral for an embedded processor
- Timing and Power analysis

Bit True, Cycle True Models
- developed by the people that made the IP!
Design optimization using SysGen blocks
Examples

- **DSP Macro**
  - Select operations to use
  - Choose which register stage to implement
  - Specify dedicated routing

- **BRAM**
  - Specify type: Distributed RAM or BRAM
  - Indicate depth, latency
  - Specify bitwidth on the different ports
  - Provide reset and enable ports
  - Select Write mode: read after write, read before write, no read on write

- **FFT**
  - Choose architecture: pipeline streaming IO, radix 2/4 burst IO
  - Bitwidth
  - BRAM usage
Outlines

- Xilinx corporate
- Virtex-6 / Spartan-6 family overview
- 7-series: a new family
- Digital Signal Processing on FPGAs
- System Generator for DSP overview
- **High Level Synthesis from C: AutoESL**
- Demos with HW-SW Co-Simulation (HW in the Loop)
- Conclusion
Introduction to Xilinx
High Level Synthesis (HLS)

Daniele Bagni (daniele.bagni@xilinx.com)
EMEA DSP Specialist FAE
The future: AutoESL…
High Level Synthesis from C/C++

Accepts C/C++, SystemC
Accepts user constraints & implementation directives

AutoESL C-to-RTL High Level Synthesis (HLS)

RTL output in Verilog, VHDL and SystemC
Automatic re-use of the C-level testbench
Automated RTL Synthesis
BDTI Certification

- Two BDTI Benchmarks Conducted:
  - Video Motion Analysis Application
  - Wireless Receiver Baseband Application

- Benchmark Results:
  - “Comparable resource utilization to hand-coded RTL”
  - “40x better performance than a mainstream DSP”
  - “tools required a similar level of effort as required for DSP”

Results for the BDTI High-Level Synthesis Tool Certification Program © 2010 BDTI. For more info and results see www.BDTI.com.
HLS vs. “C to DSP” Design Flow
Ease of use, Quality of Results

BDTI Case Study

BDTI Optical Flow Workload. © 2010 BDTI. Used with Permission

- Ease of use and results compared to TI DaVinci and CCS
- Tracks Pixel motion across multiple video frames
- Initial results achieved in AutoESL with minor code edits

<table>
<thead>
<tr>
<th>Metric</th>
<th>C to DSP Flow</th>
<th>AutoESL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Final Performance Achieved</td>
<td>5.1 fps</td>
<td>185 fps</td>
</tr>
<tr>
<td>Cost ($) / FPS</td>
<td>$4.25</td>
<td>$.14</td>
</tr>
</tbody>
</table>
HLS vs. “RTL to FPGA” Design Flow

Quality of Results

BDTI Case Study

RTL Created by experience hardware designer
- Used 2 optimized CoreGen IP blocks
- Both designs met performance

<table>
<thead>
<tr>
<th>Metric</th>
<th>Hand Coded RTL</th>
<th>AutoESL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance</td>
<td>23.4 Gops @75 MHz</td>
<td>23.4 Gops @75 MHz</td>
</tr>
<tr>
<td>FPGA Utilization (Spartan3A DSP 3400)</td>
<td>5.9%</td>
<td>5.6%</td>
</tr>
</tbody>
</table>

BDTI DQPSK Workload. © 2010 BDTI. Used with Permission
HLS Value Proposition

- Simulate C/C++/SystemC instead of RTL: 10000x faster
- Design and verify in C instead of RTL: 4-5x faster
  - Correctness and verification is 80% of the work in RTL
- Correctness is based on C, performance on compiler directives (or C preprocessor #pragmas)
- One design can reach several performance points, portable over generations of FPGAs
- These tools are for embedded algorithm designers and for existing RTL designers
- However: need to understand the tool and WHY the directives work the way they do.
Improved Productivity with C-Based Hardware Verification

- Significant productivity gains achieved by migrating functional verification to C/C++
  - 2 to 3 orders of magnitude faster than RTL for large designs
  - RTL verification becomes final check
    - Verified against C/C++ Test harness

**Optical flow Video Example**

<table>
<thead>
<tr>
<th>Input</th>
<th>C Simulation Time</th>
<th>RTL Simulation Time</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 frames of video data</td>
<td>10 seconds</td>
<td>~2 days*</td>
<td>~12,000X</td>
</tr>
</tbody>
</table>

* RTL Simulations performed using ModelSim
Design Variations with Directives

- **AutoESL directives are used to modify the design implementation from it’s default**

The same hardware is used for each iteration of the loop:
- Small area
- Long latency
- Long Throughput

Different hardware is used for each iteration of the loop:
- Higher area
- Short latency
- Better Throughput

Different iterations are executed concurrently:
- Higher area
- Short latency
- Best throughput
Arbitrary Precision Integers

- **C and C++ have standard types created on the 8-bit boundary**
  - char (8-bit), short (16-bit), int (32-bit), long long (64-bit)
    - Also provides stdint.h (for C), and stdint.h and cstdint (for C++)
    - Types: int8_t, uint16_t, uint32_t, int_64_t etc.
  - They result in hardware which is not bit-accurate and can give sub-standard QoR

- **AutoESL provides bit-accurate types in both C and C++**
  - Allow any arbitrary bit-width to be specified
  - Will simulate with bit-accuracy

```c
#include autopilot_tech.h
void foo_top (...) {
    int1 var1;    // 1-bit
    uint1 var1u;   // 1-bit unsigned
    int2 var2;    // 2-bit
    ...
    int1024 var1024; // 1024-bit
    uint1024 var1024u; // 1024-bit unsigned
    ...
}
```

```c
#include ap_int.h
void foo_top (...) {
    ap_int<1> var1; // 1-bit
    ap_uint<1> var1u; // 1-bit unsigned
    ap_int<2> var2; // 2-bit
    ...
    ap_int<1024> var1024; // 1024-bit
    ap_int<1024> var1024u; // 1024-bit unsigned
    ...
}
```
Outlines

- Xilinx corporate
- Virtex-6 / Spartan-6 family overview
- 7-series: a new family
- Digital Signal Processing on FPGAs
- System Generator for DSP overview
- High Level Synthesis from C: AutoESL
- Demos with HW-SW Co-Simulation (HW in the Loop)
- Conclusion
Demos with HW-SW Co-Simulation (HW in the Loop)
Demos with HW-SW Co-Simulation (HW in the Loop)

- HW-SW Co-Simulation Basics Using System Generator for DSP: how it works
- Demo on ML605 board: Edge Detection on images
- Demo on ML605: frame-based HW-SW Co-Simulation
Demos with HW-SW Co-Simulation (HW in the Loop)

- HW-SW Co-Simulation Basics Using System Generator for DSP: how it works
  - Demo on ML605 board: Edge Detection on images
  - Demo on ML605: frame-based HW-SW Co-Simulation
HW-SW Co-Simulation Basics Using System Generator for DSP: how it works
Virtex-6 FPGA DSP Kit

- Xilinx ML605 Development Board
  - Dual FMC Daughter Card slots
  - Virtex-6 LX240T Device
    - 768 DSP48E1 Slices
    - Designs can migrate to SXT family
- One year entitlement to ISE Design Suite: System Edition
  - Includes System Generator for DSP
- DSP Reference design
  - RTL
  - Simulink
- Documentation
  - Getting Started Guided
  - Design Tutorials
  - Board schematics

http://www.em.avnet.com/v6dspkit
HW Co-Simulation Using System Generator

Simulink test bench running on host computer

HW running on target board

SW / HW interfaces automatically handled by System Generator
HW Co-Simulation Environment

System Generator Software

Connection Options:
• JTAG
• Ethernet

• Verification environment without expensive emulators
• Flexible connectivity between PC and target board

Xilinx or customer board
Hardware Co-Simulation

Advantages

- Accelerate simulation up to 1000x
- Powerful Simulink® verification environment
- No hardware knowledge required
- All above advantages can also be applied to RTL designs

<table>
<thead>
<tr>
<th>Design</th>
<th>Simulation Time (Seconds)</th>
<th>Increase</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Software</td>
<td>HW Co-Sim</td>
</tr>
<tr>
<td>Beamformer</td>
<td>113</td>
<td>2.5</td>
</tr>
<tr>
<td>OFDM BER Test</td>
<td>742</td>
<td>.75</td>
</tr>
<tr>
<td>DUC CFR</td>
<td>731</td>
<td>23</td>
</tr>
<tr>
<td>Color Space Converter</td>
<td>277</td>
<td>4</td>
</tr>
<tr>
<td>Video Scalar</td>
<td>10422</td>
<td>92</td>
</tr>
</tbody>
</table>
Basic steps in the process

- Build your design
- Create a testbench
- Choose your target HW
- Compile & run
Create a Design in System Generator

- Create design in System Generator
- Rich library of Xilinx FPGA components in Simulink
Compile a System Generator Design

Start with a model that is ready to be compiled for hardware co-simulation.

1. Double click here

Select an appropriate compilation target from the System Generator block dialog box.

2. Double click here
Compilation (2)

Select Clock Frequency option

Press the Generate button.

Press the Generate button.
The compilation creates a new library containing a parameterized run-time co-simulation block.

Compilation creates both the HW co-sim design and the testbench.

Add the co-simulation run-time block to a System Generator model.
HW co-sim output can be captured in multiple ways

System Generator builds and operates all the interfaces
Supported Boards

- System Generator automatically supports many Xilinx development boards

- Custom boards can be added via a setup wizard
  - Only requires JTAG access to the target FPGA
  - SBDBuilder inside System Generator configures target
Choosing an Interface

- **JTAG (parallel/USB)**
  - Support for any board with a Xilinx FPGA, JTAG header, and clock source
  - Burst-transfer support
    - 1 Mbps down to the board
    - 0.5 Mbps back from the board

- **Ethernet**
  - Network-based
  - Point-to-point
Ethernet HW-SW Co-Simulation

- **Two flavors**
  - Network-based
    - Remote access
    - 10/100/1000 Base-T
    - Ethernet-based configuration
  - Point-to-point
    - Requires a direct connection between host PC and FPGA
    - 10/100/1000 Base-T
    - Ethernet or JTAG-based (that is, Platform USB or PC4) configuration
Demos with HW-SW Co-Simulation (HW in the Loop)

- HW-SW Co-Simulation Basics Using System Generator for DSP: how it works

- Demo on ML605 board: Edge Detection on images

- Demo on ML605: frame-based HW-SW Co-Simulation
Demo on ML605 board: Edge Detection on images
3x3 Sobel scheme: top level

- The Simulink model combines Simulink subsystems with a Xilinx System Generator for DSP synthetisable subsystem
3x3 Sobel scheme: SysGen level -1
3x3 Sobel scheme: SysGen level -2
x directional filter subsystem
SysGen Timing Analyzer and ISE Reports
3x3 Sobel scheme: Simulink

Top & -1 levels
Define algorithm
Generate HDL code with Simulink HDL Coder
Insert this block in a black box
Run simulation with HDL code simulation
Run simulation with hardware Co-Simulation
Generating VHDL with HDL Composer
SysGen VHDL Black Boxing
HW-SW Co-Simulation on ML605: the model

Sobel Edge Detection HW SW CO-SIM on ML605 board

Reference Filter

Input Image → Edges

Point-to-point Ethernet

sobel3x3_BlackBox hwcosim (Xilinx Point-to-point Ethernet Hardware Co-simulation)

<table>
<thead>
<tr>
<th>Basic</th>
<th>Ethernet</th>
<th>Configuration</th>
<th>Shared Memories</th>
<th>Software</th>
</tr>
</thead>
</table>

Clocking

Clock source:
- Single stepped
- Free running

Has combinational path

Bitstream file: `\hwswcosim_ML605\sobel3x3_blackbox_cw.bit`
HW-SW Co-Simulation on ML605: the results

- The fixed point reference Simulink results perfectly match the HW model generated results for both X, Y and whole output filtered data.
Demos with HW-SW Co-Simulation (HW in the Loop)

- HW-SW Co-Simulation Basics Using System Generator for DSP: how it works
- Demo on ML605 board: Edge Detection on images
- Demo on ML605: frame-based HW-SW Co-Simulation
Demo on ML605: frame-based HW-SW Co-Simulation
HW-SW Co-Simulation Methodologies

- **Sample-based HW-SW Co-simulation**
  - Scalar data type transfers only

- **Frame-based HW-SW Co-simulation**
  - Vector, Frame, and Matrix data types transfers
Frame-Based Acceleration Advantages

- Vector and frame data types improve simulation performance

- Lockable Shared Memory
- Shared FIFO
- Shared Memory Read/Write
- Shared Memory Read
- Shared Memory Write
Basic steps in the process

1. Create testbench with input and output buffers
2. Create a subsystem
3. Generate a hardware co-sim block
4. Replace subsystem with the hardware Co-sim block
5. Convert a testbench from Sample-based to Frame-based
Create Test Bench with Input and Output Buffers

1. Build testbench and input/output buffers

2. Create a subsystem
Generate HW Co-sim Block

Generate a hardware co-sim block for the hardware_cosim subsystem

Replace hardware_cosim subsystem with the hardware co-sim block

- At Step.3: Software Simulink® simulation can be performed at this point
- At Step.4: hardware co-simulation can be performed at this step but it’s still using single word, data transfers (scalar data type)
Add all required blocks

- Add Simulink input and output data conversion blocks
- Add Simulink buffer and unbuffer blocks
- Replace To/FROM FIFO blocks with Shared Memory Read/Write blocks
Data Flows

- Buffer
- Transfer
- Process
- Write
- Unbuffer
DEMO: 5x5 2D FIR filter
Outlines

- Xilinx corporate
- Virtex-6 / Spartan-6 family overview
- 7-series: a new family
- Digital Signal Processing on FPGAs
- System Generator for DSP overview
- High Level Synthesis from C: AutoESL
- Demos with HW-SW Co-Simulation (HW in the Loop)

- Conclusion
Conclusion
Summary

- **XILINX Virtex-6, Spartan-6 and 7-series** are optimized for performance, low power consumption and ease of use.

- **XILINX FPGAs** are best choice for High performance DSP
  - Up to 5280 DSP48
  - Up to 3.4 TMACC in a single chip

- **System Generator for DSP** is the XILINX reference tool for DSP development
  - Based on MATLAB/Simulink
  - HDL code insertion (Black Box)
  - HW-SW CO-SIM with any FPGA board (JTAG)

- **AutoESL** is the newest Xilinx tool for High Level Synthesis directly from C/C++/SystemC