OpenFPGA CoreLib Core Library Interoperability Effort

M. Wirthlin¹, D. Poznanovic², P. Sundararajan³, A. Coppola⁴, D. Pellerin⁵, W. Najjar⁶, R. Bruce⁷, M. Babst⁸, O. Pritchard⁹, P. Palazzari¹⁰, G. Kuzmanov¹¹

¹Brigham Young University, ²SRC Computers, ³Xilinx Corporation, ⁴OptNgn Software, ⁵Impulse, ⁶UC Riverside, ⁷Nallatech, ⁸DSPLogic, ⁹Altera Corporation, ¹⁰Ylichron, ¹¹TU Delft
Overview

• Background and Motivation
• OpenFPGA CoreLib Project Goals
• Related Hardware Circuit Reuse Efforts
• XML & IP-XACT
• Example
• Status and Future Work
### FPGA Design Methods

<table>
<thead>
<tr>
<th>RTL</th>
<th>HLL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Register Transfer Level</strong></td>
<td>High-Level Languages</td>
</tr>
<tr>
<td>VHDL, Verilog, etc.</td>
<td>C, C++, Java, SystemC, etc.</td>
</tr>
<tr>
<td><strong>“Low-Level” Circuit Design</strong></td>
<td>Graphical Programming</td>
</tr>
<tr>
<td>Focus on implementation</td>
<td></td>
</tr>
<tr>
<td>Time consuming</td>
<td><strong>“High-Level” Algorithm Design</strong></td>
</tr>
<tr>
<td>High quality circuits</td>
<td>Focus on algorithm</td>
</tr>
<tr>
<td></td>
<td>Implementation details ignored</td>
</tr>
<tr>
<td></td>
<td>Possibly lower quality circuits</td>
</tr>
<tr>
<td><strong>High Performance</strong></td>
<td><strong>High Productivity</strong></td>
</tr>
<tr>
<td>Meet timing constraints</td>
<td>Easy to modify algorithm</td>
</tr>
<tr>
<td>Meet resource constraints</td>
<td>Fewer Implementation details</td>
</tr>
<tr>
<td>Utilized specialized resources</td>
<td>Less design time</td>
</tr>
</tbody>
</table>
High Performance “Cores”

• High-performance circuit “cores” needed for high-performance
  – Perform performance critical functions (FFT, etc.)
  – Hand crafted, efficient implementations
  – Manage a complex resources (MGT, memories, etc.)

• Cores often described in “low-level” language
  – VHDL, EDIF, raw bitstreams, etc.
  – Provide high-performance, hand crafted circuit “functions”
High Performance “Cores”

• High-performance circuit “cores” needed for high-performance
  – Perform performance critical functions (FFT, etc.)
  – Hand crafted, efficient implementations
  – Manage a complex resources (MGT, memories, etc.)

• Cores often described in “low-level” language
  – VHDL, EDIF, raw bitstreams, etc.
  – Provide high-performance, hand crafted circuit “functions”

• Challenge:
  – How do we integrate high-performance cores into high-level languages?
  – How do we reuse high-performance cores with more than one high-level language?
Project Goals

1. Develop a standard for circuit cores
2. Develop a standard for circuit libraries
3. Encourage the use of these standards
4. Create a synergistic environment
5. Create Libraries
Hardware Reuse

• Reuse of hardware circuits is very important for hardware engineers
  – Reduce design cost of large single chip systems
  – Amortize cost of circuit cores over multiple chips
  – Purchase high-quality circuits from “IP” specialists

• Hardware reuse is more difficult than software reuse
  – Complex interfaces
    • Signal timing, signal types, communication protocol
  – Physical implications
    • Circuit area, timing, power, placement, etc.
  – Challenging verification
    • How do you know it works?
Hardware Reuse Efforts

opencores.org
- Open source repository of circuit cores
- Cores developed for wishbone bus interface

OPC-IP
- Specifies a common standard for core interface
- Provides tools/infrastructure for integrating cores

SPIRIT Consortium
- Develops specs for describing circuits for re-use
- Created IP-XACT v1.2 spec (used in this work)

Virtual Socket Interface Alliance (VSI)
- Provide standards, docs, and methods for SoC design
- Facilitate circuit protection and transfer

Si2
- Facilitate interoperability of EDA tools
- Provides common libraries and interoperability tools
Reuse with Existing HLL Tools

- Techniques used for importing external cores
  - Function call interface
  - New language semantics or PRAGMA statements
  - Overloading standard operators
  - New graphical library elements
  - Custom instructions

- Example

```c
double a, b;
b = sqrt(a);
```

```vhdl
entity dpfp_sqrt_64 is
  clk, ce : in std_logic;
  load : in std_logic;
  a, out : std_logic_vector(63 downto 0)
end dpfp_sqrt_64;
architecture rtl of dpfp_sqrt_64 is
  -- lots of cool design tricks
  .
  .
end rtl;
```
Reuse with Existing HLL Tools

• Techniques used for importing external cores
  – Function call interface
  – New language semantics or PRAGMA statements
  – Overloading standard operators
  – New graphical library elements
  – Custom instructions

• Commercial
  – Impulse-C
  – Dime-C
  – Carte
  – Reconfigurable Computing Toolbox
  – C2H

• Research
  – ROCCC
  – Trident
  – DWB
  – CHiMPS
Reuse Challenges

• Each tool has a custom method for importing core
• Importing external core for new tool requires extra work
• Difficult to use same core in multiple tools
• Difficult to reuse libraries among tools
Reuse Goal

• Each core and library conforms to a standard
• Each tool recognizes this standard
• Easy to use same core in multiple tools
• Easy to reuse libraries among tools
IP Interface Standard Requirements

• Structural Information
  – Signals (name, bit width, properties)
  – Signal types (higher level type information)

• Timing Interface
  – Signal arrival times
  – Signal response times

• Control interface
  – Interface protocol specification
  – Handshaking requirements

• Parameterization

• Estimation interface

• External tool/generator interface
XML Circuit Meta Description

• Exploit XML infrastructure to describe cores and libraries
  – Define custom XML schema for describing cores
  – Create XML descriptions of reusable cores
    • Define all details of complicated circuit interface
    • Provide multiple views of core
  – Package cores into reusable libraries

• Benefits
  – Many tools available for manipulating/viewing XML
  – Several HLL tools already use XML to describe cores
  – Existing techniques for publishing cores with XML (IP-XACT)
IP-XACT

- XML meta description of reusable IP
  - XML schema defining tags for specifying reusable IP
  - Defines interface, configuration, and generation of IP
  - Used primarily for defining bus-based IP in SoC design
  - Current standard intended for RTL-level IP
- Created by the Spirit Consortium
  - Non-profit organization with over 60 company members
  - Led primarily by ARM and Mentor Graphics
  - Being considered as an IEEE Standard (P1685)
- Compatible and complementary with other IP-reuse activities
  - Opencores, OCP-IP, etc.
Example – UART on AMBA Bus

-->

Raw XML

RSSI 2007
Example – UART on AMBA Bus

Eclipse IP-XACT Plug-In View

Several tools available for manipulating and creating IP-XACT meta descriptions
IP-XACT Extensions

• IP-XACT standard not sufficient for CoreLib effort
  – Limited to bus-based cores and HDL data types
  – Limited support for custom interfaces
  – IP-XACT supports a variety of extensions for new functionality

• Develop extensions to IP-XACT for CoreLib
  – Higher level data types (floating point, fixed point, etc.)
  – Temporal interface (timing and protocol)
  – Specifying behavior (arithmetic functions)
  – Advanced memory architectures
Square Root Example

double a,b;

b = sqrt(a);

b = my_fast_sqrt(a);

b = vendor_small_sqrt(a);
Square Root Example

\[
x = [3.2; 1.0; 2; 5];
y = [1.1, -2.3, 4, .4];
a = \text{det}(x \times y);
\]

\[
b = \sqrt{a};
\]

\[
b = \text{my\_sqrt}(a);
\]

\[
b = \text{vendor\_sqrt}(a);
\]
Status

• Completed survey of previous work
• Gathered information about tools
• Proposed framework around IP-XACT
• Experimenting with IP-XACT
• Gathering freely available cores
Future Work

• Develop sample IP-XACT wrappers for cores
• Propose and demonstrate extensions
• Solicit feedback from vendors
• Encourage use of standards
CoreLib Wiki

https://isl.ncsa.uiuc.edu/twiki/bin/view/OpenFPGA/CoreLib

- Related Circuit Reuse Activities
- Tools and Compilers
- Examples
- HLL Compiler Requirements
- Interface Standards