Introduction to High-Level Synthesis
Required Reading

The ZYNQ Book

- Chapter 14: Spotlight on High-Level Synthesis
- Chapter 15: Vivado HLS: A Closer Look

S. Neuendorffer and F. Martinez-Vallina, Building Zynq Accelerators with Vivado High Level Synthesis, FPGA 2013 Tutorial
Recommended Reading


Behavioral Synthesis
Need for High-Level Design

• Higher level of abstraction
• Modeling complex designs
• Reduce design efforts
• Fast turnaround time
• Technology independence
• Ease of HW/SW partitioning
Platform Mapping
SW/HW Partitioning

Program

Software
(executed in the microprocessor system)

Hardware
(exeucuted in the reconfigurable processor system)
SW/HW Partitioning & Coding
Traditional Approach

- Specification
- SW/HW Partitioning
  - SW Coding
    - SW Compilation
    - SW Profiling
  - HW Coding
    - HW Compilation
    - HW Profiling
SW/HW Partitioning & Coding

New Approach

Specification

SW/HW Coding

SW/HW Partitioning

SW Compilation

SW Profiling

HW Compilation

HW Profiling
Advantages of Behavioral Synthesis

- Easy to model higher level of complexities
- Smaller in size source compared to RTL code
- Generates RTL much faster than manual method
- Multi-cycle functionality
- Loops
- Memory Access
Short History of High-Level Synthesis

Generation 1 (1980s-early 1990s): research period

Generation 2 (mid 1990s-early 2000s):
• Commercial tools from Synopsys, Cadence, Mentor Graphics, etc.
• Input languages: behavioral HDLs  Target: ASIC

Outcome: Commercial failure

Generation 3 (from early 2000s):
• Domain oriented commercial tools: in particular for DSP
• Input languages: C, C++, C-like languages (Impulse C, Handel C, etc.), Matlab + Simulink, Bluespec
• Target: FPGA, ASIC, or both

Outcome: First success stories
Hardware-Oriented High-Level Languages

- **C-Based System level languages**
  - **Commercial**
    - Handel C -- Celoxica Ltd.
    - Impulse C -- Impulse Accelerated Technologies
    - Carte C – SRC Computers
    - SystemC -- The Open SystemC Initiative
  - **Research**
    - Streams-C -- Los Alamos National Laboratory
    - SA-C -- Colorado State University, University of California, Riverside, Khoral Research, Inc.
    - SpecC – University of California, Irvine and SpecC Technology Open Consortium
Other High-Level Design Flows

- Matlab-based
  - AccelChip DSP Synthesis -- AccelChip
  - System Generator for DSP -- Xilinx
- GUI Data-Flow based
  - Corefire -- Annapolis Microsystems
- Java-based
  - Commercial
    - Forge -- Xilinx
  - Research
    - JHDL – Brigham Young University
Handel-C Overview

- High-level language based on ISO/ANSI-C for the implementation of algorithms in hardware
- Allows software engineers to design hardware without retraining
- Clean extensions for hardware design including flexible data widths, parallelism and communications
- Well defined timing model
  - Each statement takes a single clock cycle
- Includes extended operators for bit manipulation, and high-level mathematical macros (including floating point)
Handel-C/ANSI-C Comparisons

ANSI-C

- Preprocessors i.e. #define
- Pointers
- Structures
- ANSI-C Constructs for, while, if, switch
- Bitwise logical operators
- Logical operators
- Arithmetic operators
- Functions
- ANSI-C Standard Library
- Recursion
- Floating Point

HANDEL-C

- Handel-C Standard Library
- Parallelism
- Arbitrary width variables
- Enhanced bit manipulation
- RAM, ROM
- Signals
- Interfaces

Handel-C Standard Library

HANDEL-C
Handel-C Design Flow

Executable Specification

Handel-C

Synthesis

VHDL

Place & Route

EDIF

EDIF
Different Levels of C/C++ Synthesis Abstraction

Untimed C Domain
(Non-implementation-specific)

More abstract, less implementation-specific

Timed C Domain
(Implementation-specific)

Augmented C/C++

RTL Domain
(Implementation-specific)

Verilog and VHDL

Less abstract, more implementation-specific

Pure C/C++
Pure Untimed C/C++ Design Flow

Pure C/C++ → Pure C/C++ Synthesis → Verilog / VHDL RTL → RTL Synthesis → Gate-level netlist

- Non-implementation-specific
- Easy to create
- Fast to simulate
- Easy to modify

User interaction and guidance

ASIC target → Auto-generated, implementation-specific

FPGA target

Verilog / VHDL RTL → RTL Synthesis → LUT/CLB-level netlist

The Design Warrior’s Guide to FPGAs
Devices, Tools, and Flows. ISBN 0750676043
Copyright © 2004 Mentor Graphics Corp. (www.mentor.com)
Catapult C

(2004-2011)

Calypto Design Systems

(2011-2015)

(2015-present)
Catapult C

- Catapult C automatically converts un-timed C/C++ descriptions into synthesizable RTL.
SystemC -based design-flow alternatives

Implementation specific, relatively slow to simulate, relatively difficult to modify

Alternative SystemC flows
Reconfigurable Supercomputers
What is a Reconfigurable Computer?

Microprocessor system

\[
\begin{align*}
&\mu P \\
&\mu P \\
&\mu P \\
&\mu P \\
&\mu P \\
&I/O
\end{align*}
\]

Reconfigurable system

\[
\begin{align*}
&FPGA \\
&FPGA \\
&FPGA \\
&FPGA \\
&FPGA \\
&I/O
\end{align*}
\]

I/O Interface

Reconfigurable system

I/O Interface
<table>
<thead>
<tr>
<th>Machine</th>
<th>Released</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRC 6 from SRC Computers</td>
<td>2002</td>
</tr>
<tr>
<td>Cray XD1 from Cray</td>
<td>2005</td>
</tr>
<tr>
<td>SGI Altix from SGI</td>
<td>2005</td>
</tr>
<tr>
<td>SRC 7 from SRC Computers, Inc,</td>
<td>2006</td>
</tr>
</tbody>
</table>
Pros and cons of reconfigurable computers

+ can be programmed using high-level programming languages, such as C, by mathematicians & scientist themselves
+ facilitates hardware/software co-design
+ shortens development time, encourages experimentation and complex optimizations
+ allows sharing costs among users of various applications

- high entry cost (~$100,000)
- hardware aware programming
- limited portability
- limited availability of libraries
- limited maturity of tools
SRC Programming Model

Microprocessor

main.c

function_1()

function_2()

FPGA

function_1

macro_1(a, b, c)

macro_2(b, d)

macro_2(c, e)

function_2

macro_3(s, t)

macro_1(n, b)

macro_4(t, k)

ANSI C

MAP C
(subset of ANSI C)

Libraries of macros

macro_1

macro_2

macro_3

macro_4

……………………

VHDL

I/O

a

Macro_1

b

Macro_2

c

d

e

Macro_2

I/O
SRC Compilation Process

**Application sources**
- .c or .f files
  - μP Compiler
    - .o files
  - MAP Compiler
    - .o files
- .mc or .mf files
  - MAP Compiler
    - .o files
- .v files
  - Logic synthesis
    - .vhd or .v files
  - HDL sources
  - Netlists
    - .ngo files
  - Place & Route
    - .bin files
- Configuration bitstreams
- Application executable

**Macro sources**
Library Development - SRC

- **μP system**
  - LLL (ASM)
  - HLL (C, Fortran)
  - HLL (C, Fortran)

- **FPGA system**
  - HDL (VHDL, Verilog)
  - HLL (C, Fortran)
  - HLL (C, Fortran)

Library Developer

Application Programmer
SRC Programming Environment

+ very easy to learn and use
+ standard ANSI C
+ hides implementation details
+ very well integrated environment
+ mature

- subset of C
- legacy C code requires rewriting
- C limitations in describing HW (parallelism, data types)
- closed environment, limited portability of code to HW platforms other than SRC
Application Development for Reconfigurable Computers

Program Entry

Platform mapping

Compilation

Debugging & Verification

Execution
Ideal Program Entry

Function

Program Entry
Actual Program Entry

- Preferred Architectures
- Use of FPGA Resources (multipliers, µP cores)
- Sequence of Run-time Reconfigurations
- SW/HW Interface
- Function
- SW/HW Partitioning
- FPGA Mapping
- Data Transfers & Synchronization
- Use of Internal and External Memories
AutoESL Design Technologies, Inc. (25 employees)

Flagship product:

AutoPilot, translating \texttt{C/C++/System C} to \texttt{VHDL} or \texttt{Verilog}

- Acquired by the biggest FPGA company, Xilinx Inc., in 2011
- AutoPilot integrated into the primary Xilinx toolset, Vivado, as \texttt{Vivado HLS}, released in 2012

“High-Level Synthesis for the Masses”
Vivado HLS

High Level Language
C, C++, System C

Vivado HLS

Hardware Description Language
VHDL or Verilog
HLS-Based Development and Benchmarking Flow

Reference Implementation in C

Manual Modifications (pragmas, tweaks)

HLS-ready C code

High-Level Synthesis

HDL Code

Physical Implementation

FPGA Tools

Netlist

Post Place & Route Results

Functional Verification

Timing Verification

Test Vectors
LegUp – Academic Tool for HLS

- Open-source HLS Tool
  - Developed at the University of Toronto
  - Faculty supervisors: Jason H. Anderson and Stephen Brown
  - FPL Community Award 2014
- High-Level Synthesis from C to Verilog
- Targets Altera FPGAs (extension to Xilinx relatively simple)
- Two flows
  - Pure Hardware
  - Hardware/Software Hybrid
    = Tiger MIPS + hardware accelerator(s) + Avalon bus + shared on-chip and off-chip memory
Cryptol – New Language for Cryptology

- Domain specific language for cryptology: Cryptol
  - High-level programming language similar to Haskell
  - Developed by Galois Inc. based in Portland, USA
- High-Level Synthesis from Cryptol to efficient Software and Hardware
Levels of Abstraction in FPGA Design

- High Level
- Behavioural
- RTL
- Structural

C/C++/SystemC design entry

HDL design entry

Source: The Zynq Book
High-Level Synthesis vs. Logic Synthesis

Source: The Zynq Book
Algorithm and Interface Synthesis

Source: The Zynq Book
Vivado HLS Design Flow

Source: The Zynq Book
Design Trade-offs Explored Using HLS

Source: The Zynq Book
C Functional Verification and C/RTL Cosimulation in Vivado HLS

Source: The Zynq Book
Vivado HLS

- Starts at C
  - C
  - C++
  - SystemC

- Produces RTL
  - Verilog
  - VHDL
  - SystemC

- Automates Flow
  - Verification
  - Implementation
Vivado HLS Scheduling and Binding

Source Files
(C, C++, SystemC)

User Directives

Technology Library

Scheduling

HLS

Binding

RTL Files
(Verilog / VHDL / SystemC)

Source: The Zynq Book
**Vivado HLS**

**Scheduling and Binding**

**Scheduling** – translation of the RTL statements interpreted from the C code into a set of operations, each with an associated duration in terms of clock cycles. Affected by the clock frequency, uncertainty, target technology, and user directives.

**Binding** - associating the scheduled operations with the physical resources of the target device.
Three Possible Outcomes from HLS
Average of 10 numbers

Source: The Zynq Book
Vivado HLS Synthesis Process

Source: The Zynq Book