System-level Synthesis of Dataflow Applications for FPGA-based Distributed Platforms

Hugo A. Andrade, Kaushik Ravindran, Alejandro Asenjo, Casey Weltzin
NI Berkeley, NI Austin
National Instruments Corporation

Tenth Biennial Ptolemy Miniconference
Berkeley, California, November 7th, 2013
Modern FPGA-based Distributed Heterogeneous Platforms

LabVIEW
Desktop, RT, FPGA, uP, DSP Design, IP Builder

PCI EXPRESS

PXI Systems  CompactRIO  Single-Board RIO

Zynq

Cortex™-A15 MPCore

Intel inside®

Modern FPGA-based Distributed Heterogeneous Platforms

LabVIEW
Desktop, RT, FPGA, uP, DSP Design, IP Builder

PCI EXPRESS

PXI Systems  CompactRIO  Single-Board RIO

Zynq

Cortex™-A15 MPCore

Intel inside®
High Level Synthesis

• A method for *increasing* (FPGA or system) *design productivity* while still providing an *efficient implementation*
  – Broadens design productivity to domain experts
    • They can use a high-level language such as ANSI C/C++, Python, MATLAB, LabVIEW
  – Design space exploration: optimizing among performance, footprint, or other constraints
  – Enables targeting more complex platforms from higher level program
HLS Techniques Covered

- Directive-based high-level synthesis
- Synthesis from analyzable domain specific languages
- System-level synthesis to heterogeneous targets

Applicable to textual and graphical hardware programming environments, e.g. VHDL/Verilog or LabVIEW FPGA
Y-Chart Disciplined Design Methodology

Application Logic

Analysis and Mapping

Performance Evaluation

Deployment


Models of Computation and Platform-Based Design

Dataflow

C / HDL Code

Textual Math

Simulation

Statechart

LabVIEW

Desktop, RT, FPGA, uP

Personal Computers

PXI Systems

CompactRIO

Single-Board RIO

Custom Design

Ref: Models of Computations, Prof. Edward Lee, Platform Based Design, Prof. Alberto Sangiovanni-Vincentelli
Models of Computation and Platform-Based Design

Dataflow

C / HDL Code

Textual Math

Simulation

Statechart

LabVIEW

Desktop, RT, FPGA, uP

Personal Computers

PXI Systems

CompactRIO

Single-Board RIO

Processor

FPGA

Custom I/O

I/O

I/O

I/O

Ref: Models of Computation, Prof. Edward Lee, Platform Based Design, Prof. Alberto Sangiovanni-Vincentelli
Base Programming Language: G

Timing

Built-in I/O

Math and Analysis

Concurrency

User Interface
Base Programming Language: G
DIRECTIVE-BASED HIGH-LEVEL SYNTHESIS
Focus on describing algorithm in high-level programming language
Add separate directives

Platform model
Known single target FPGA
Explore mapping based on directive sets
User in the loop

Code generation
Configured reusable IP
Current Challenges Programming in G

FIR filter in regular LabVIEW:

G implementation after manual unrolling and pipelining:
LabVIEW FPGA IP Builder User Flow

- Create Algorithm VI
- Specify Directives
- Generate performance estimation
- Generate Design
- Use within top-level LV FPGA VI

• Use dataflow programming
• Limited functions palette
LabVIEW FPGA IP Builder User Flow

1. Create Algorithm VI
2. Specify Directives
3. Generate performance estimation
4. Generate Design
5. Use within top-level LV FPGA VI
LabVIEW FPGA IP Builder User Flow

1. Create Algorithm VI
2. Specify Directives
3. Generate performance estimation
4. Generate Design
5. Use within top-level LV FPGA VI
LabVIEW FPGA IP Builder User Flow

1. Create Algorithm VI
2. Specify Directives
3. Generate performance estimation
4. Generate Design
5. Use within top-level LV FPGA VI

Choose VI and Directives
Generate HDL
Create IP VI
LabVIEW FPGA IP Builder User Flow

- Create Algorithm VI
- Specify Directives
- Generate performance estimation
- Generate Design
- Use within top-level LV FPGA VI

- Integrate into Single-Cycle Timed Loop
- Add I/O, DMA FIFOs, Host Communication, etc.
Exploration Example - Directives

Throughput = (ClockRate * SamplesPerIteration) / InitiationInterval,
## Comparison

<table>
<thead>
<tr>
<th>Example Set #</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Directive Settings</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>40</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>Initiation Interval (Cycles)</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td><strong>Estimated Performance</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clock Rate (MHz)</td>
<td>42</td>
<td>250</td>
<td>250</td>
</tr>
<tr>
<td>Initiation Interval (Cycles)</td>
<td>62</td>
<td>77</td>
<td>1</td>
</tr>
<tr>
<td>Latency (cycles)</td>
<td>61</td>
<td>76</td>
<td>6</td>
</tr>
<tr>
<td>Throughput (MS/s)</td>
<td>0.68</td>
<td>3.25</td>
<td>250</td>
</tr>
<tr>
<td><strong>Estimated Resource Utilization</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Registers</td>
<td>0.80%</td>
<td>0.90%</td>
<td>6.30%</td>
</tr>
<tr>
<td>LUTs</td>
<td>0.80%</td>
<td>0.80%</td>
<td>2.70%</td>
</tr>
<tr>
<td>Multipliers</td>
<td>1</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>Block RAMs</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Estimate Time (s)</td>
<td>28</td>
<td>28</td>
<td>30</td>
</tr>
</tbody>
</table>
SYNTHESES FROM ANALYZABLE DOMAIN SPECIFIC LANGUAGES
• **Domain specific application language**
  - E.g. streaming applications in communications domain
• **Prequalified IP Component Library**

• Platform model
  - Known single target FPGA
• Mapping based on domain specific intent
  - Dataflow optimizations
  - Hardware specific optimizations
• Code generation
  - Configured reusable IP
- **Domain specific application language**
  - E.g. streaming applications in communications domain
- **Prequalified IP Component Library**

**Platform**
- Platform model
  - Known single target FPGA
- Mapping based on domain specific intent
  - Dataflow optimizations
  - Hardware specific optimizations

**Deployment**
- Code generation
  - Configured reusable IP

**RF Communications Applications**
- **Research**
  - Network Optimization
  - Interference Alignment
  - Single Ant Diversity
  - Cognitive Radio
- **Surveillance**
  - Signal INT
  - Communications INT
  - Electronic INT
  - Spectrum Sniffer
- **Advanced Test**
  - RF Channel Emulation
  - BTS Emulation
  - Protocol Analysis
  - Next Gen Test
Streaming Model of the OFDM Transmitter

- **Nt = \{1,2,4\}**
  - Compile time - # transmitters
- **Nu = \{72, 180, 300, 600, 900, 1200\}**
  - Initialization time - Bandwidth
- **CP mode = \{‘Normal’, ‘Extended’\}**
  - Run time, To overcome Inter-symbol-interference, Can be applied at symbol boundary
- **CP Vector**
  - Selection based on CP mode, Elements must be applied at symbol boundary

Challenge: How to express a domain expert’s algorithm specification in a model that is viable for analysis and implementation?
MoCs for Streaming Applications

<table>
<thead>
<tr>
<th></th>
<th>Expressive</th>
<th>Analyzable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kahn Process Networks</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Boolean Dataflow</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SHIM</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Heterochronous Dataflow</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Static Dataflow</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Homogeneous Dataflow</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Area of focus for DSP Designer

<table>
<thead>
<tr>
<th>Feature</th>
<th>Expressive</th>
<th>Analyzable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Bounded data rates?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Deadlock and boundedness decidable?</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Static scheduling?</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Key trade-off: Analyzability vs. Expressability

Application Specification in LabVIEW DSP Design Module
Application Specification in LabVIEW DSP Design Module

Sub Diagrams  Xilinx CoreGen Blocks  Consumption/Production Rates  Data Ports
Tool Flow
Tool Flow

- Actor Definition
- Performance Models
- and Timing Library
- IP Modeling and Integration

Models of Computation

Analysis and Optimization
Back End

Simulation and Verification

• Code Generation and Implementation
Analysis and Optimization Features

• Core dataflow optimizations
  – Model validation
    • Deadlock detection and boundedness check
  – Throughput and latency computation
  – Buffer size optimization (under throughput constraints)
  – Schedule computation

• Hardware specific optimizations
  – Resource constrained schedule computation
  – Retiming and fusion
  – Rate matching
  – IP interface synthesis

Analysis and Optimization Results

Auto buffer sizing to minimize resources

Calculated firing counts and timing data

Throughput constraints
Analysis and Optimization Results

Calculated Schedule View
High-Level Model to FPGA blocks
Extending SDF with Access Patterns

- **SDF**
  - Optimal throughput = 1 sample/cycle
  - Buffer Size for optimal throughput = 16

- **SDF-AP**
  - Optimal throughput = 1 sample/cycle
  - Buffer Size for optimal throughput = 2

- **SDF-AP Model of Computation**
  - Access Patterns viewed only by the tool, not the user
  - Formal syntax and operational semantics
  - Definitions of key model properties
    - Executability, boundedness and throughout
  - Case studies to evaluate these algorithms
  - Executability for SDF -AP model is decidable
  - Boundedness for SDF -AP model is decidable
SYSTEM-LEVEL SYNTHESIS TO HETEROGENEOUS TARGETS
Approach

- Application language
  - System Specifications & Constraints
  - Target agnostic
- Prequalified IP Component Library

- Platform model
  - General element library
  - Platform constraints

- General mapping model
  - Mapping constraints
  - Mapping objectives
  - Optimization toolbox
  - Platform selection

- System-level simulation, verification, validation

- Target code generation
  - Performance results
  - Cost results
Modern FPGA-based Distributed Heterogeneous Platforms
On any edge, max bandwidth supported is 800 MB/s for unidirectional traffic, and 750 for bidirectional traffic for 128 byte packets. The bandwidth can be shared linearly among competing streams.
**Analysis and Mapping**

**Problem Inputs**

- **Actor**: \( v_0, v_1, v_2, v_3, v_4 \)
- **II**: 1, 1000, 500, 1000, 1
- **Area**: 100, 1000, 2000, 1500, 100

**Max BW**: 800MB/s

**Optimization problem**

- **Enforce**
  - Bandwidth constraint per link/switch
  - Area constraint per target
  - Link bound constraint per link
  - Affinity, grouping and exclusion

- **Optimize**
  - Throughput
  - Area
  - Latency

**Solver and heuristic methods applied, and generate valid mappings**

**Results**:

- Optimal mappings for OCT and OFDM models on 2- and 4- FPGAs in 1-3 mins
- Exact solver methods intractable for problems with over 50 tasks
- Heuristics within 20% of optimal; runtime of 1s for problems over 100 tasks
Experimental Results

• Platform and mapping model enabled experiments
• Static problem formulation (SMT solvers)
• Intuitive optimal results, sometimes even better than manual
• Reasonable scalability (alternate heuristics)
Summary

• Reviewed current trends in HLS
  – Viable method for increasing (FPGA or system) design productivity while still providing an efficient implementation

• Techniques Covered (Y-chart Methodology)
  – Directive-based synthesis
  – Synthesis from analyzable domain specific application models
  – System-level synthesis to heterogeneous targets

• These techniques can be combined hierarchically
Thank you

Questions? Comments?
Acknowledgements

This work was done in collaboration with:
Arkadeb Ghosal, Rhishikesh Limaye, Douglas Kim, Jeff Correll, Jacob Kornerup, Ian Wong, Guoqiang Wang, Guang Yang, Amal Ekbal, Mike Trimborn, Ankita Prasad, Trung N Tran, and Randy Allen.