The Dataflow Interchange Format: Towards Co-design of DSP-oriented Dataflow Models and Transformations

Shuvra S. Bhattacharyya

Maryland DSPCAD Research Group
http://www.ece.umd.edu/DSPCAD/home/dspcad.htm

Department of Electrical and Computer Engineering, and Institute for Advanced Computer Studies
University of Maryland, College Park, 20742, USA.

Outline

• Introduction to the dataflow interchange format (DIF) project, dataflow transformations, and DICE
• Application case study: high energy physics
• Wrapup
The Dataflow Interchange Format (DIF)

- DIF captures coarse grain dataflow applications formally [4]
- To formally describe applications, the DIF Language (TDL) is
  - Designed to capture a variety of dataflow models
  - Can be used in conjunction with functionally simulatable actor descriptions
- To facilitate design, the DIF Package (TDP) provides:
  - Scheduler, simulator, analyzers

Other benefits to beginning with a formal description:
- Bounded memory and deadlock detection
- Buffer and communication minimization:
- Parallel, Multirate loop, or Quasi-static scheduling
- Heterogeneous task mapping and co-synthesis
- Probabilistic design, Data partitioning, Vectorization,
The DIF Language: Sketch

```plaintext
[dataflowModel] graphID { 
  basedon { 
    graphID; 
  } 

[topology] { 
  nodes = ndID, ...; 
  edges = edgeID(srcNdID, snkNdID), ...; 
} 

[ builtinAttr ] { 
  elementID = value; 
  elementID = id; 
  elementID = id1, id2, ...; 
} 

[attribute] usrDefAttr { 
  elementID = value; 
  elementID = id; 
  elementID = id1, id2, ...; 
} 

[refinement] { 
  ... 
} 
```
Evolution of Dataflow Models of Computation for DSP: Examples

- Kahn process networks [Kahn 1974]
- Synchronous dataflow, [Lee 1987]
  - Static multirate behavior
  - SPW (Cadence), National Instruments LabVIEW, and others.
- Well behaved stream flow graphs [1992]
  - Schemas for bounded dynamics
- Boolean/integer dataflow [Buck 1994]
  - Turing complete models
- Multidimensional synchronous dataflow [Lee 1992]
  - Image and video processing
- Scalable synchronous dataflow [Ritz 1993]
  - Block processing
  - COSSAP (Synopsys)
- CAL [Eker 2003]
  - Actor-based dataflow language
- Cyclo-static dataflow [Bilsen 1996]
  - Phased behavior
  - Eonic Virtuoso Synchro, Synopsys El Greco and Cocentric,
  Angeles System Canvas
- Bounded dynamic dataflow
  - Bounded dynamic data transfer [Pankert 1994]
- The processing graph method [Stevens, 1997]
  - Reconfigurable dynamic dataflow
  - U. S. Naval Research Lab, MCCI Autocoding Toolset
- Stream-based functions [Kienhuis 2001]
- Parameterized dataflow [Bhattacharya 2001]
  - Reconfigurable static dataflow
  - Meta-modeling for more general dataflow graph reconfiguration
- Reactive process networks [Geilen 2004]
- Blocked dataflow [Ko 2005]
  - Image and video through parameterized processing
- Windowed synchronous dataflow [Keinert 2006]
- Parameterized stream-based functions [Nikolov 2008]
- Enable-involve dataflow [Plishker 2008]
- Variable rate dataflow [Wiggers 2008]
DIF Project Components

• Core components
  – The DIF language (TDL)
  – The DIF package (TDP)
  – Enable-invoke dataflow (EIDF) and functional DIF
  – DIFML: XML dialect

• Plug-ins
  – DIF-to-C: Software synthesis for SDF
  – TDIF and TDIFSyn
  – The dataflow schedule graph (DSG)

• Interfaces to ADS, OpenDF, LabVIEW, Ptolemy II, …
High Level Dataflow Transformations

• A well designed dataflow representation exposes opportunities for high level algorithm and architecture transformations.
• High level of abstraction $\rightarrow$ high implementation impact
• Dataflow representation is suitable both for behavior-level modeling, structural modeling, and mixed behavior-structure modeling
  – Transformations can be applied to all three types of representations to focus subsequent steps of the design flow on more favorable solutions
• Complementary to advances in
  – C compiler technology (intra-actor functionality)
  – Object oriented methods (library management, application service management)
  – HDL synthesis (intra-actor functionality)
Representative Dataflow Analyses and Optimizations

• Bounded memory and deadlock detection: consistency
• Buffer minimization: minimize communication cost
• Multirate loop scheduling: optimize code/data trade-off
• Parallel scheduling and pipeline configuration
• Heterogeneous task mapping and co-synthesis
• Quasi-static scheduling: minimize run-time overhead
• Probabilistic design: adapt system resources and exploit slack
• Data partitioning: exploit parallel data memories
• Vectorization: improve context switching, pipelining
• Synchronization optimization: self-timed implementation
• Clustering of actors into atomic scheduling units
Formal Model Detection
(Core Functional Dataflow [3])

- Divide actors into a set of modes
  - Each mode has a fixed consumption and production behavior
- Write the enabling conditions for each mode
- Write the computation associated with each mode
  - Including next mode to enable and then invoke
- For example, consider a standard Switch:

  Production & consumption behavior of switch modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Consumes</th>
<th>Produces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>1 0</td>
<td>0 0</td>
</tr>
<tr>
<td>True</td>
<td>0 1</td>
<td>1 0</td>
</tr>
<tr>
<td>False</td>
<td>0 1</td>
<td>0 1</td>
</tr>
</tbody>
</table>
Practical Model Detection on Units

• Deterministic – Does the output repeat?

- Input Sequence 1: 00101110
- Input Sequence 2: 00101110
- Output Sequence 1: 1101101000
- Output Sequence 2: 1101101000

• Statefulness – Does the output just reorder?

- Input Sequence 1: 00101110
- Input Sequence 2: 00101011
- Output Sequence 1: 1101101000
- Output Sequence 2: 1100001101

• Dataflow model – Does input & output behavior repeat?

- Input Sequence 1: 00101110
- Input Sequence 2: consumes 2
- Consumes 2
- Produces 4
- Produces 3
- Output Sequence 1: 1101101000
- Output Sequence 2: produces 4
- Produces 3
DICE: DSPCAD Integrative Command-Line Environment [2]

What it is…
- a framework for managing cross-platform testing
- language independent
- an open source resource

What it does not do
- provide code synthesis or debugging tools
- provide simulation capabilities
- transcode between platforms or languages
Outline

• Introduction to the dataflow interchange format (DIF) project, dataflow transformations, and DICE
• Application case study: high energy physics
• Wrapup
Case Study: Compact Muon Solenoid Trigger

• Complex:
  – 9300 magnets
  – Protons travel at 99.99% times the speed of light
  – 7 TeV beam collisions

• Performance Oriented:
  – 6 collision detectors
  – 600 million proton collisions per second

• International Collaboration:
  – 2000 Scientists
  – 155 Institutes
  – 37 Countries
CMS Trigger Background

- Large Hadron Collider (LHC)
  - CERN: Switzerland/France
  - Event rate of 1GHz
  - Trigger Selectivity: ratio of trigger rate to event rate (e.g., $10^{-11}$)
- Compact Muon Solenoid
  - General purpose particle physics detector for the LHC
  - CMS Trigger: Multi-Level Filtering: Level 1 (FPGA) $\rightarrow$ High Level Trigger (software) $\rightarrow$ Tape storage

Goals: Efficient, Agile Design

• The upgraded Calorimeter Trigger will require new algorithms
• Modern field programmable gate arrays (FPGAs) provide efficient platforms
• Implement Calorimeter Trigger using
  – A unified design platform
  – Unified design and test methodologies
  – Techniques that facilitate future upgrades
• Start by implementing a baseline design for the new algorithms
Solution: Novel Implementations and a Unified Cross-Platform Management System

- Collaboration with University of Wisconsin [1]
- Novel FPGA designs
  - Reexamination of physics algorithms for FPGAs
  - Structured analysis of resource usage
- Cross-platform design management
  - Novel, light weight development framework
  - Cross-platform unit testing
  - Dataflow model detection
  - Enhanced auto-documentation

Automatically generated application graph
Impact: Performance and Cost

- Novel FPGA implementations for over a dozen modules in the CMS detector
  - Improve performance
  - Cut implementation costs by reducing the number of FPGAs required for the upgrade

- New design process
  - Bugs found earlier in design process saves time and money
  - Automated documentation facilitates fast collaborative design process
Processing Detectors

- 56x72 sized grids
- With millions of events a second, storing all of the data would result in GigaBytes per second
- Instead, store only events that trigger certain conditions
- L1 trigger finds image features that represent certain particles from a series of:
  - Thresholding
  - Filtering
  - Sorting
- Must complete in nanoseconds to process every sample period
Triggering Application Graph
Written by application designers and then re-implemented by hardware engineers → Cross-platform verification is a problem
Test directory structure
Model based testbench creation

Text files (sample input provided by user)

ECAL 1-4
File reader 1

FG 1-4
File reader 2

HCAL 1-4
File reader 3

Cluster Computation Actor

File writer 1
38b output

Comparator from larger test framework

Expected 38b output
Results

<table>
<thead>
<tr>
<th>Actor</th>
<th>Inputs</th>
<th>Outputs</th>
<th>Deterministic</th>
<th>Model Detected</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster Thresh</td>
<td>12</td>
<td>12</td>
<td>Yes</td>
<td>HSDF</td>
<td>No</td>
</tr>
<tr>
<td>Cluster Compute</td>
<td>12</td>
<td>6</td>
<td>Yes</td>
<td>HSDF</td>
<td>No</td>
</tr>
<tr>
<td>Overlap Filter</td>
<td>8</td>
<td>4</td>
<td>Yes</td>
<td>SDF</td>
<td>No</td>
</tr>
<tr>
<td>Jet Reconstruction</td>
<td>1</td>
<td>2</td>
<td>Yes</td>
<td>SDF</td>
<td>No</td>
</tr>
</tbody>
</table>

\[(H)SDF = (homogeneous) synchronous dataflow\]
Outline

• Introduction to the dataflow interchange format (DIF) project, dataflow transformations, and DICE
• Application case study: high energy physics
• → Wrapup
Summary

• The dataflow interchange format (DIF) project
  – The DIF Language (TDL)
  – The DIF Package (TDP)
  – Plug-ins for simulation and synthesis
• The DSPCAD Integrative Command Line Environment (DICE)
• Application case study: high-energy physics
• Other ongoing application thrusts in the DIF project include: embedded speech processing, software-defined radio, wireless sensor networks, image registration, radio astronomy instrumentation
• Co-design of dataflow-based representations and transformations
Portions of the work presented here have been sponsored by DARPA (through MCCI), and NSF (ECCS0823989 and CNS0720596).

For more details on these projects, and associated publications:  
http://www.ece.umd.edu/DSPCAD/home/dspcad.htm.
To Probe Further …

(Available from: http://www.ece.umd.edu/DSPCAD/papers/contents.html)


