A. JAMES CLARK
SCHOOL OF ENGINEERING

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.

Ptolemy Miniconference, University of California, Berkeley,
Feb. 16, 2011 [Version: Feb. 13, 2011]

UMIACS
University of Maryland Institute for Advanced Computer Studies

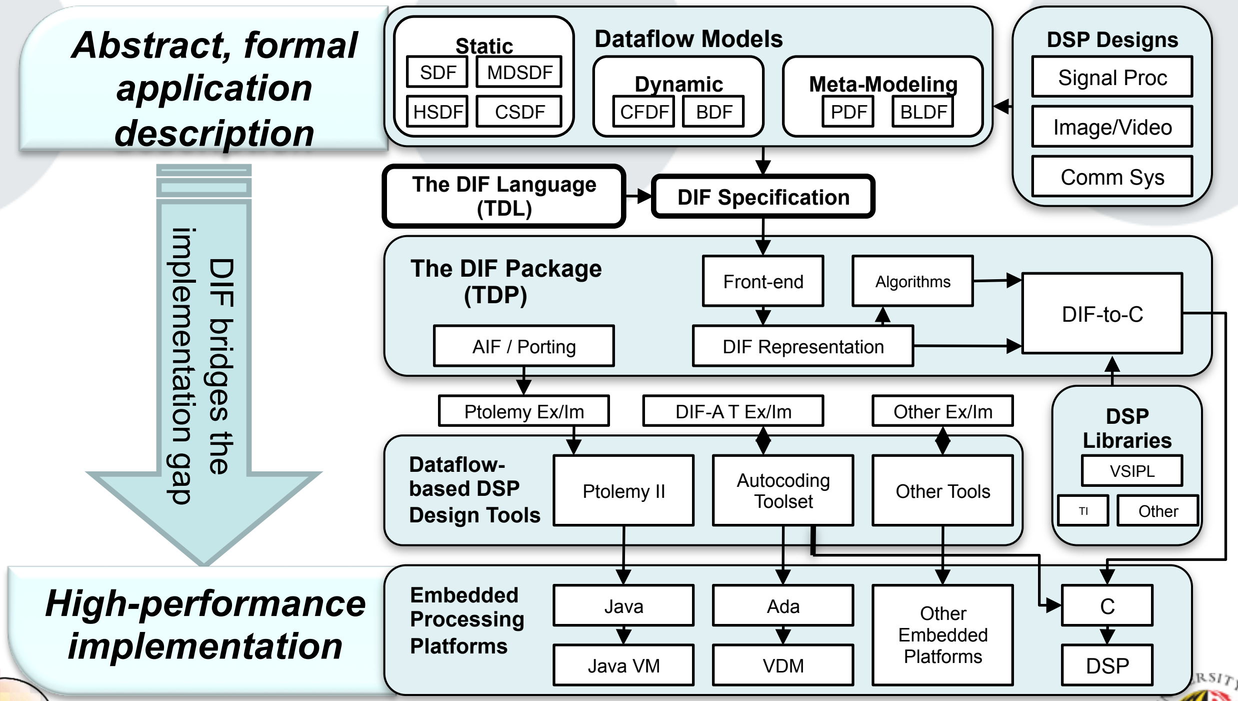
DEPARTMENT OF
ELECTRICAL & COMPUTER ENGINEERING

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

```
[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

- Computation Graphs and Marked Graphs [Karp 1966, Reiter 1968]
- 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-invoke dataflow [Plishker 2008]
- Variable rate dataflow [Wiggers 2008]

A. JAMES CLARK
SCHOOL OF ENGINEERING

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 → 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)

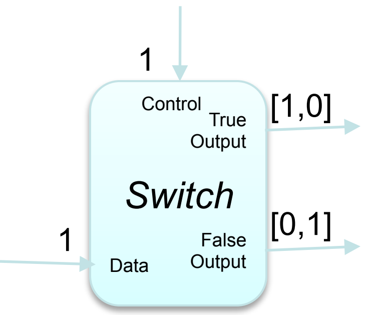
Representative Dataflow Analyses and Optimizations

- 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:

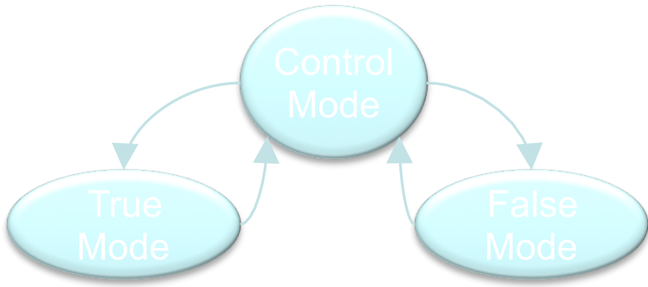
Switch Actor



Production & consumption behavior of switch modes

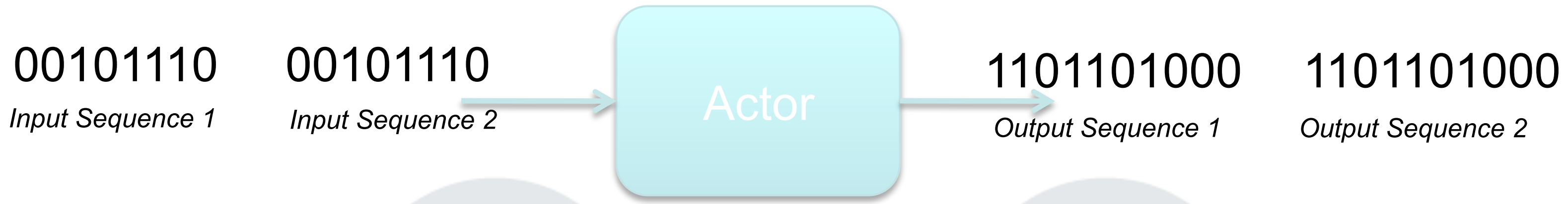
Mode	Consumes		Produces	
	Control	Data	True	False
Control	1	0	0	0
True	0	1	1	0
False	0	1	0	1

Mode transition diagram between switch modes

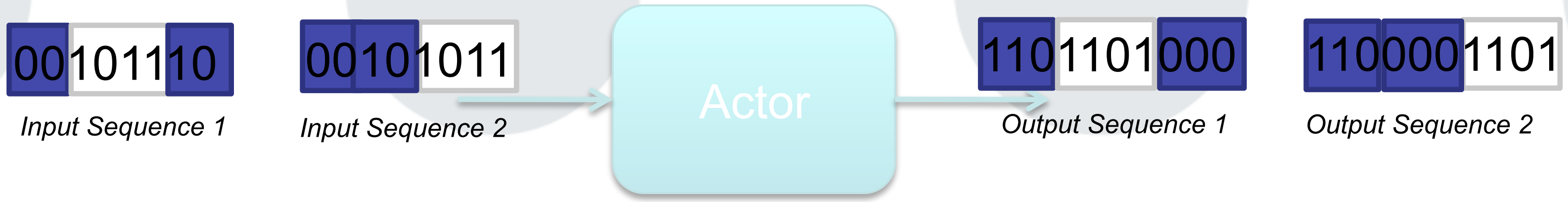


Practical Model Detection on Units

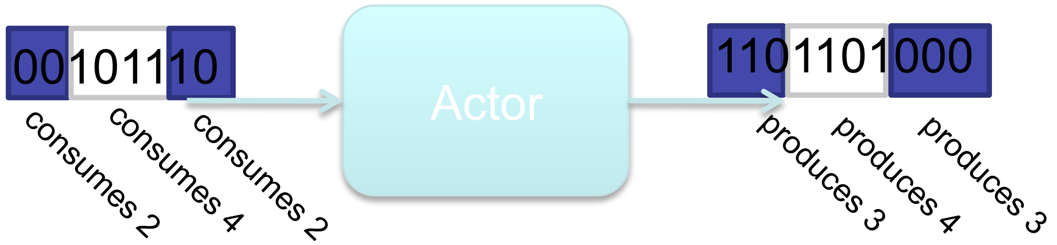
- Deterministic – Does the output repeat?



- Statefulness – Does the output just reorder?



- Dataflow model – Does input & output behavior repeat?



A. JAMES CLARK
SCHOOL OF ENGINEERING

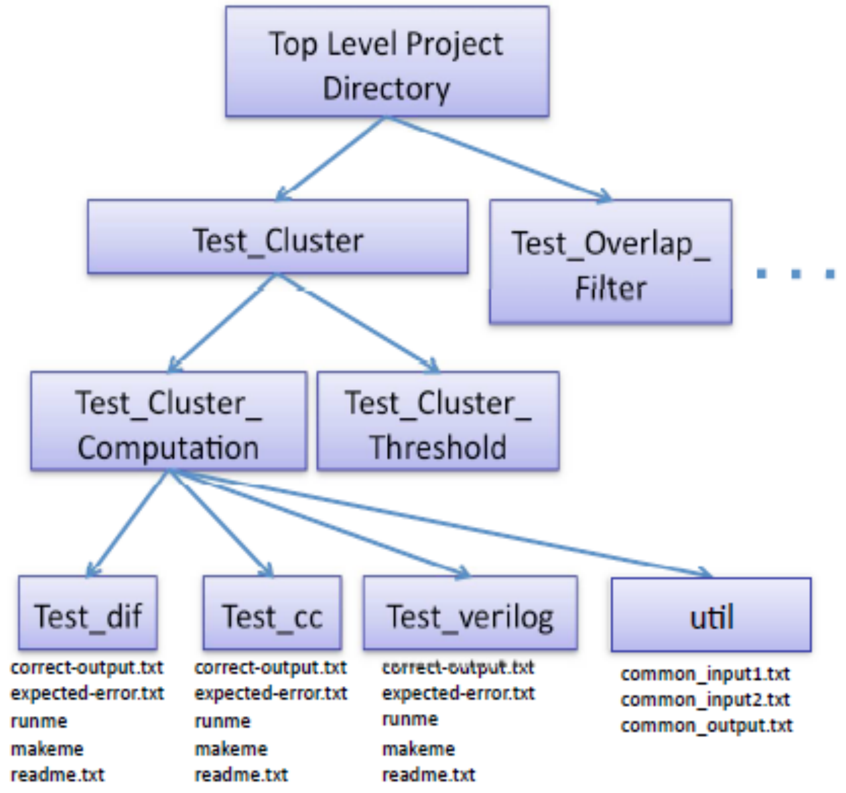
DICE: DSPCAD Integrative Command-Line Environment [2]

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



A. JAMES CLARK
SCHOOL OF ENGINEERING

Outline

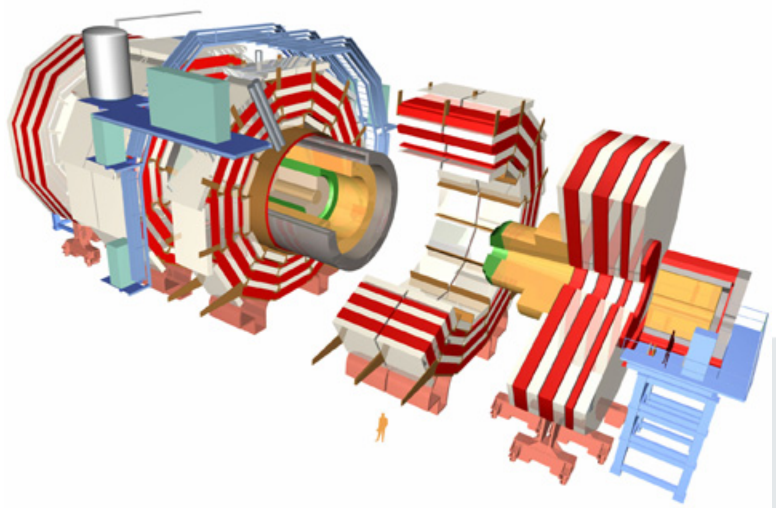
- Introduction to the dataflow interchange format (DIF) project, dataflow transformations, and DICE
- → Application case study: high energy physics
- Wrapup

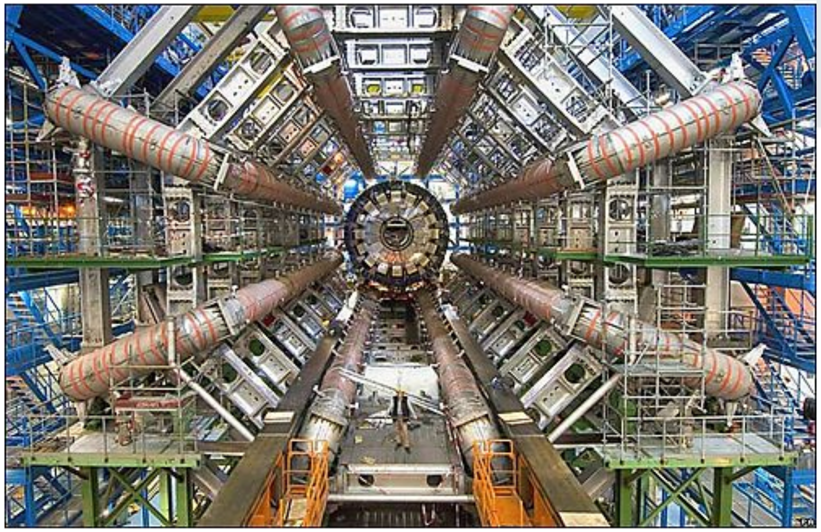
UMIACS
University of Maryland Institute for Advanced Computer Studies

DEPARTMENT OF
ELECTRICAL &
COMPUTER ENGINEERING

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: Switzlerland/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) → High Level Trigger (software) → Tape storage

Source: http://en.wikipedia.org/wiki/Trigger_%28particle_physics%29, and http://en.wikipedia.org/wiki/Compact_Muon_Solenoid#Layer_2_.E2.80.93_The_Electromagnetic_Calorimeter, Dec. 1, 2010.

A. JAMES CLARK
SCHOOL OF ENGINEERING

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

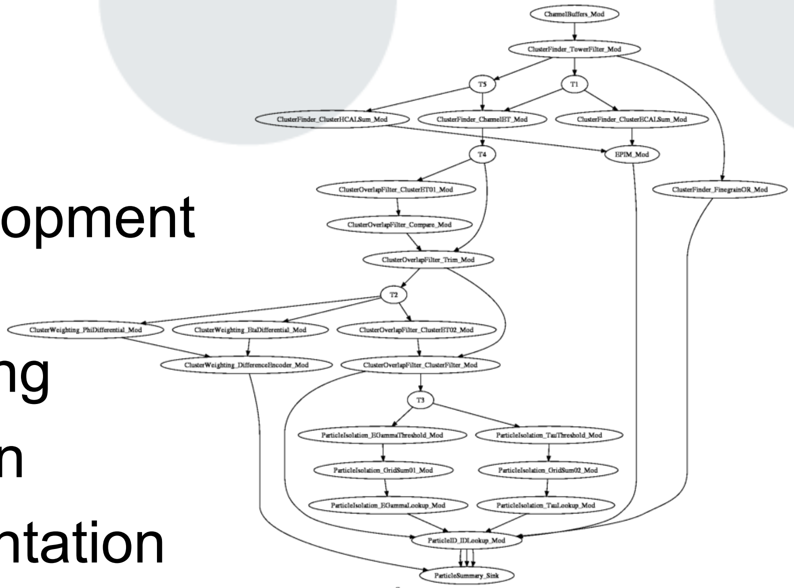
UMIACS
University of Maryland Institute for Advanced Computer Studies

DEPARTMENT OF
ELECTRICAL &
COMPUTER ENGINEERING

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


ChannelBuffers_Mod
ClusterFinder_TowerFilter_Mod
T5
T1
ClusterFinder_ClusterHCALSum_Mod
ClusterFinder_ChannelET_Mod
ClusterFinder_ClusterECALSum_Mod
T4
EPIM_Mod
ClusterOverlapFilter_ClusterET01_Mod
ClusterFinder_FinegrainOR_Mod
ClusterOverlapFilter_Compare_Mod
ClusterOverlapFilter_Trim_Mod
T2
ClusterWeighting_PhiDifferential_Mod
ClusterWeighting_EtaDifferential_Mod
ClusterOverlapFilter_ClusterET02_Mod
ClusterWeighting_DifferenceEncoder_Mod
ClusterOverlapFilter_ClusterFilter_Mod
T3
ParticleIsolation_EGammaThreshold_Mod
ParticleIsolation_TauThreshold_Mod
ParticleIsolation_GridSum01_Mod
ParticleIsolation_GridSum02_Mod
ParticleIsolation_EGammaLookup_Mod
ParticleIsolation_TauLookup_Mod
ParticleID_IDLookup_Mod
ParticleSummary_Sink

c

A. JAMES CLARK
SCHOOL OF ENGINEERING

Impact: Performance and Cost

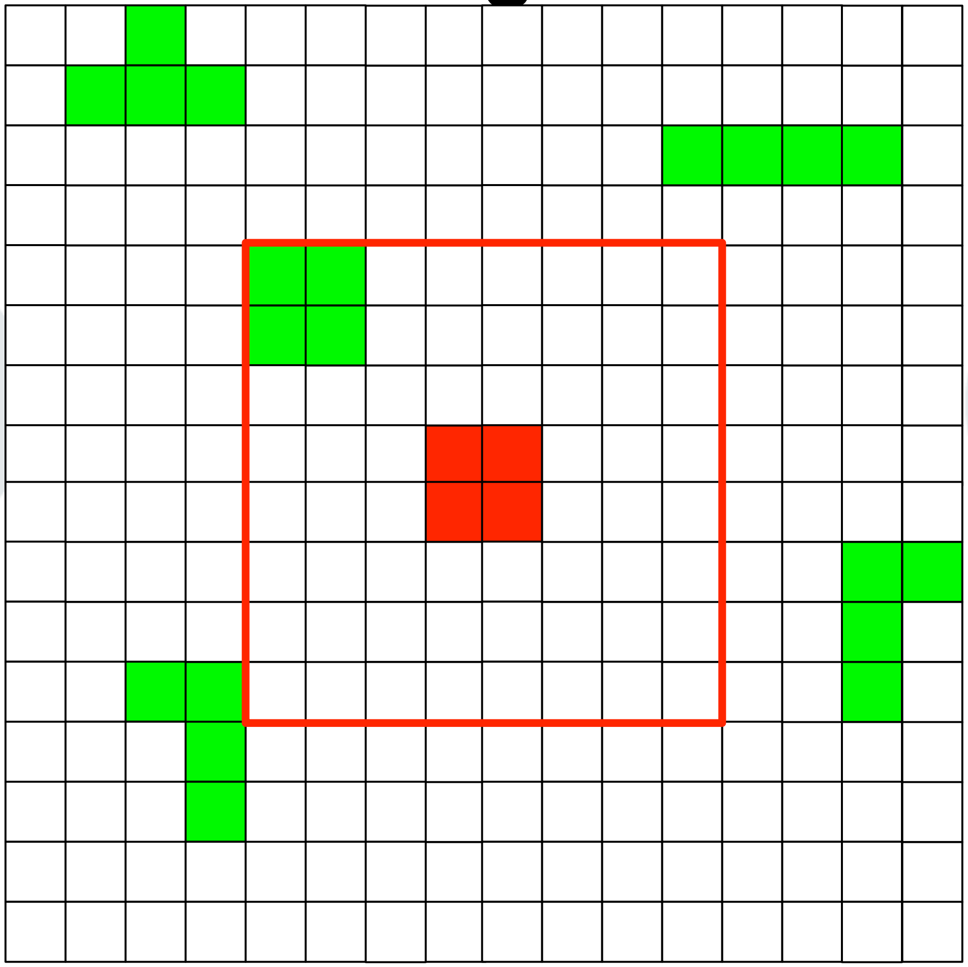
- Novel FPGA implementations for over a dozen modules in the CMS detector
  - Improve performance
  - Cut implementation costs by reducing the number of FPGAs required for the upgrade
- New design process
  - Bugs found earlier in design process saves time and money
  - Automated documentation facilitates fast collaborative design process

UMIACS
University of Maryland Institute for Advanced Computer Studies

DEPARTMENT OF
ELECTRICAL &
COMPUTER ENGINEERING

A. JAMES CLARK
SCHOOL OF ENGINEERING

Processing Detectors

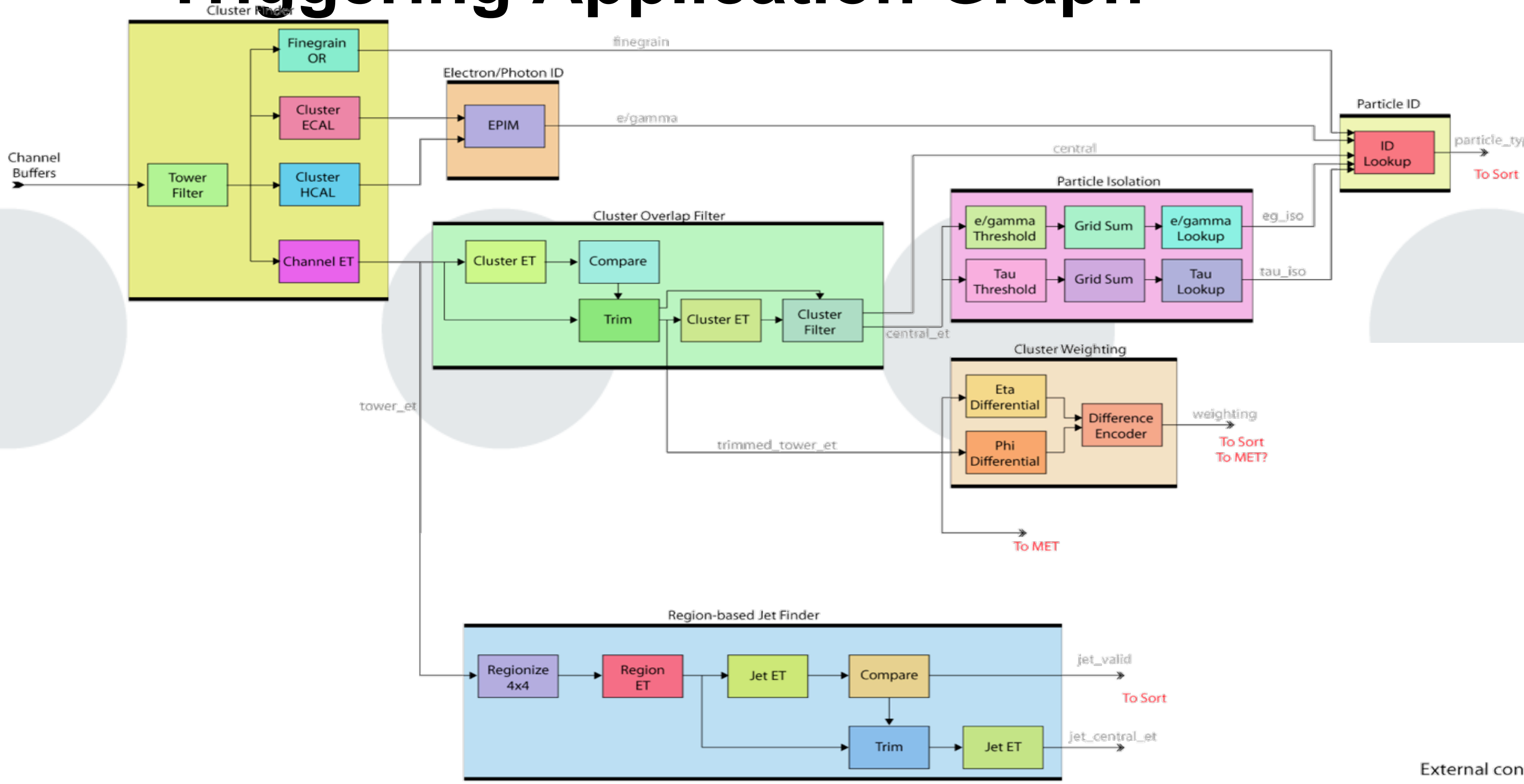


- 56x72 sized grids
- With millions of events a second, storing all of the data would result in GigaBytes per second
- Instead, store **only** events that trigger certain conditions
- L1 trigger finds image features that represent certain particles from a series of:
  - Thresholding
  - Filtering
  - Sorting
- Must complete in nanoseconds to process every sample period

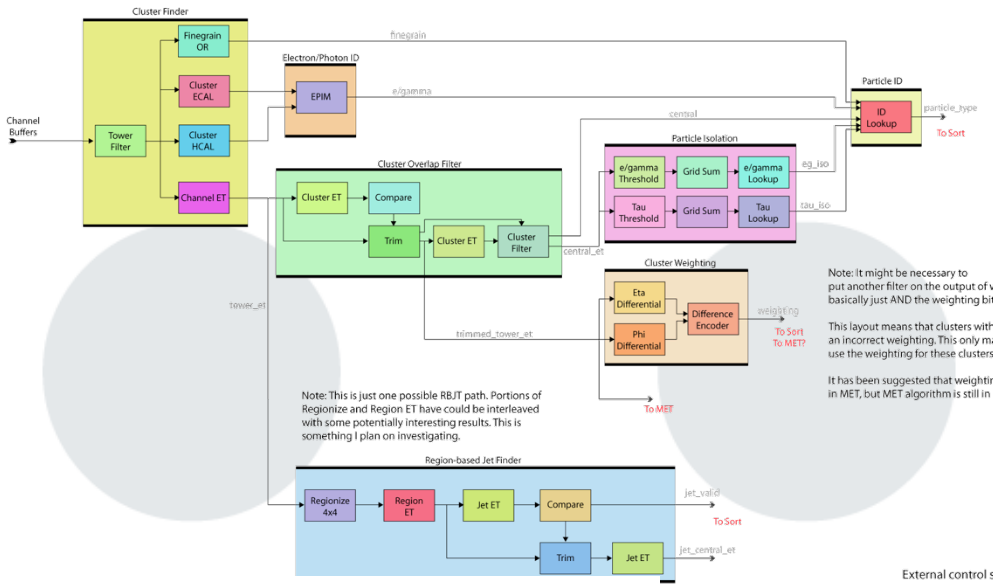
UMIACS
University of Maryland Institute for Advanced Computer Studies

DEPARTMENT OF
ELECTRICAL & COMPUTER ENGINEERING

Triggering Application Graph



Triggering Application Graph



Note: It might be necessary to
put another filter on the output of
basically just AND the weighting bi

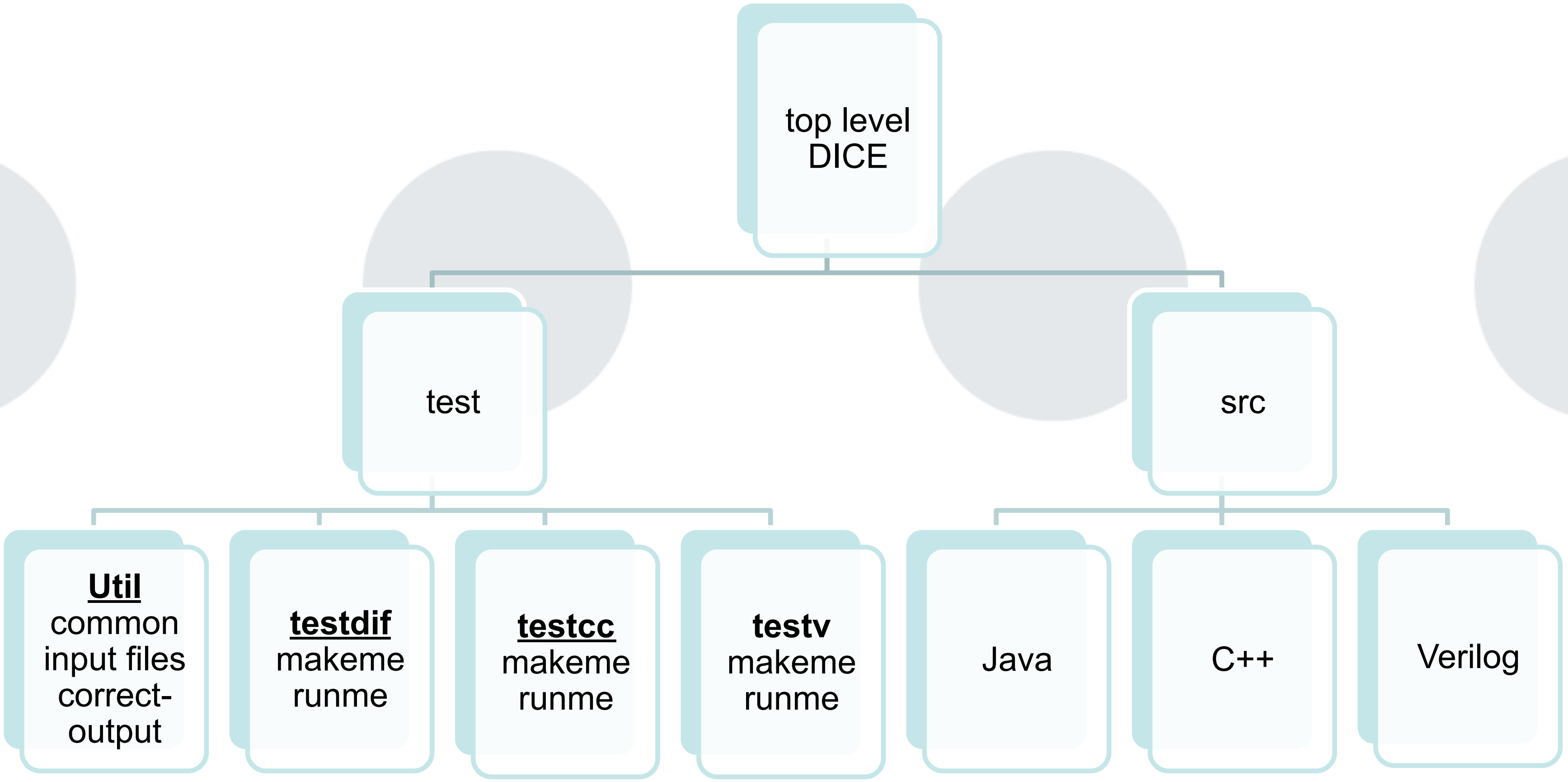
This layout means that clusters wit
an incorrect weighting. This only m
use the weighting for these clusters

It has been suggested that weightin
in MET, but MET algorithm is still in

Note: This is just one possible RBJT path. Portions of Regionize and Region ET have could be interleaved with some potentially interesting results. This is something I plan on investigating.

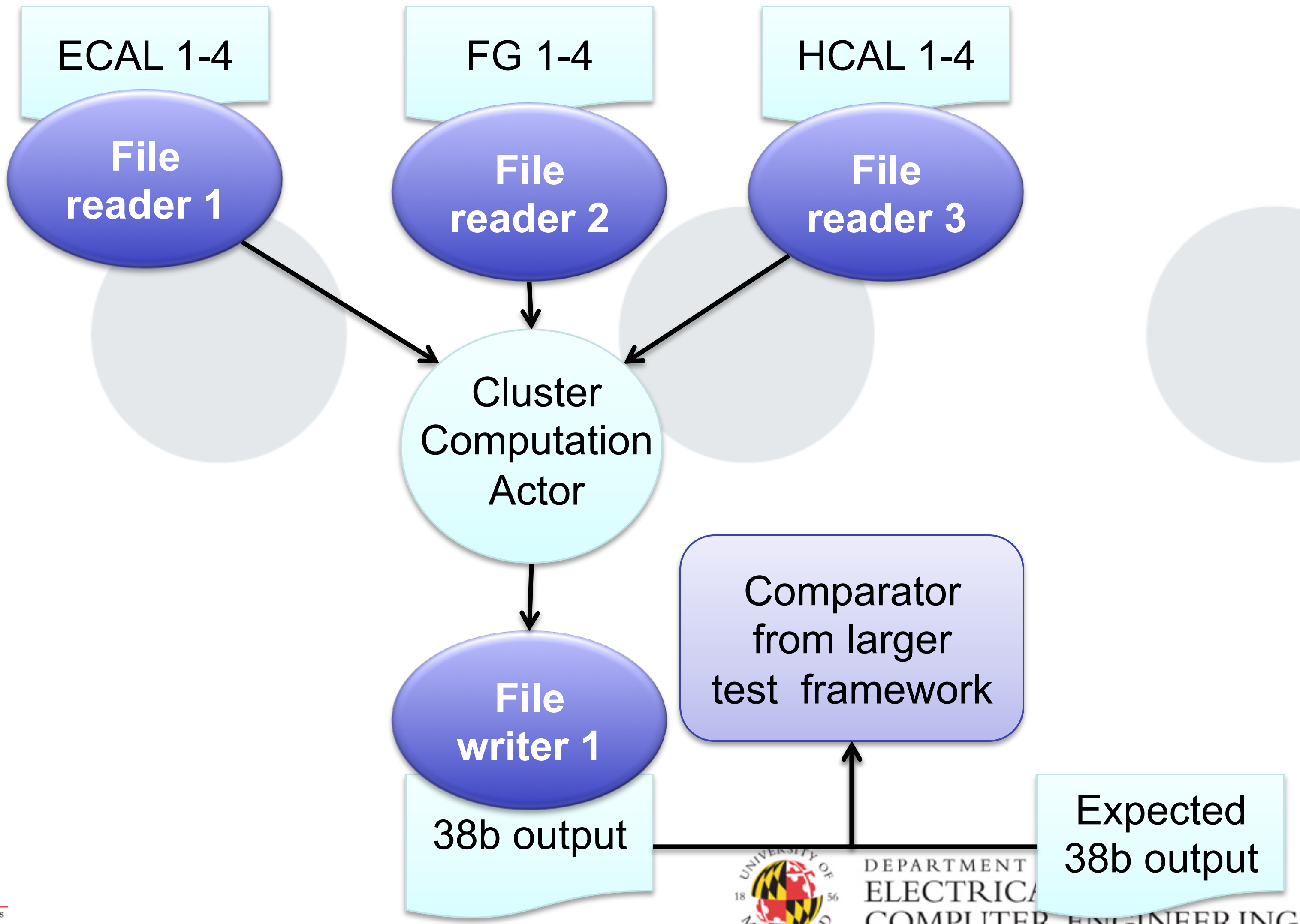
Written by application designers and then re-implemented by hardware engineers → Cross-platform verification is a problem

Test directory structure

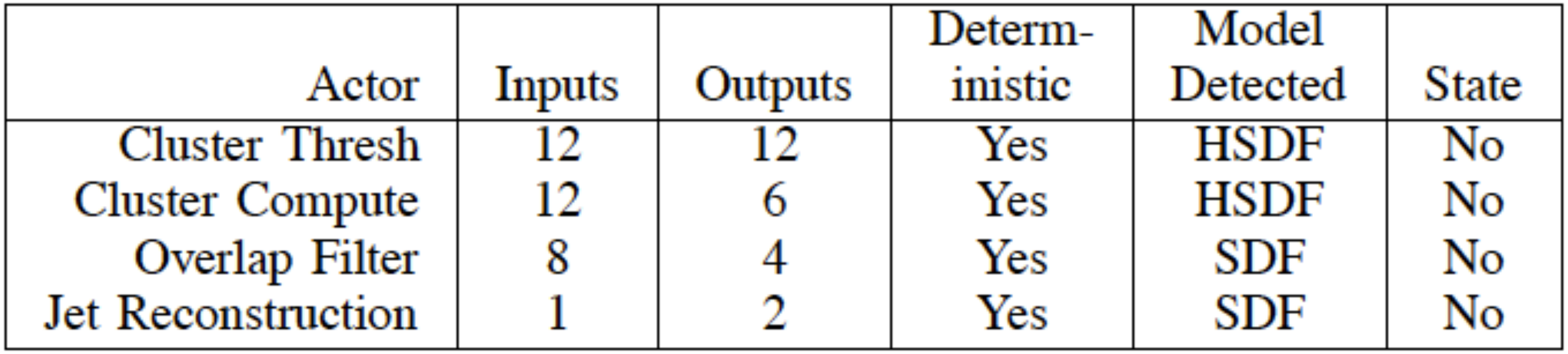


Model based testbench creation

Text files (sample input provided by user)



Results



Actor	Inputs	Outputs	Determ-inistic	Model Detected	State
Cluster Thresh	12	12	Yes	HSDF	No
Cluster Compute	12	6	Yes	HSDF	No
Overlap Filter	8	4	Yes	SDF	No
Jet Reconstruction	1	2	Yes	SDF	No

(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

A. JAMES CLARK
SCHOOL OF ENGINEERING

Acknowledgements

- 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.**

UMIACS
University of Maryland Institute for Advanced Computer Studies

DEPARTMENT OF
ELECTRICAL &
COMPUTER ENGINEERING

To Probe Further …

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

- [1] W. Plishker, C. Shen, S. S. Bhattacharyya, G. Zaki, S. Kedilaya, N. Sane, K. Sudusinghe, T. Gregerson, J. Liu, and M. Schulte. Model-based DSP implementation on FPGAs. In *Proceedings of the International Symposium on Rapid System Prototyping*, Fairfax, Virginia, June 2010. Invited paper.
- [2] S. S. Bhattacharyya, S. Kedilaya, W. Plishker, N. Sane, C. Shen, and G. Zaki. The DSPCAD integrative command line environment: Introduction to DICE version 1. Technical Report UMIACS-TR-2009-13, Institute for Advanced Computer Studies, University of Maryland at College Park, August 2009.
- [3] W. Plishker, N. Sane, M. Kiemb, K. Anand, and S. S. Bhattacharyya. Functional DIF for rapid prototyping. In *Proceedings of the International Symposium on Rapid System Prototyping*, pages 17-23, Monterey, California, June 2008.
- [4] C. Hsu, F. Keceli, M. Ko, S. Shahparnia, and S. S. Bhattacharyya. DIF: An interchange format for dataflow-based design tools. In *Proceedings of the International Workshop on Systems, Architectures, Modeling, and Simulation*, pages 423-432, Samos, Greece, July 2004.