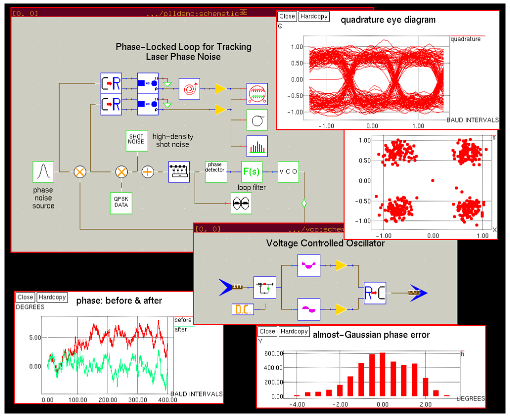
The Ptolemy Project

Shuvra Bhattacharyya
Joseph T. Buck
Wan-Teh Chang
Brian L. Evans
Steve X. Gu
Sangjin Hong
Christopher Hylands
Asawaree Kalavade
Alan Kamas
Allen Lao
Bilung Lee
Edward A. Lee
David G. Messerschmitt
Praveen K. Murthy
Thomas M. Parks
José Luis Pino
Farhana Shiekh
S. Sriram
Juergen Teich
Warren W. Tsai
Patrick J. Warner
Michael C. Williamson

System-Level Design of Signal Processing Systems

Ptolemy Research

- Design complexity management.
- Visual, algorithm-level system design.
- Formal methods for dataflow systems.
- Programming language semantics.
- Software and hardware synthesis.
- Parallel architectures, partitioning, and scheduling.

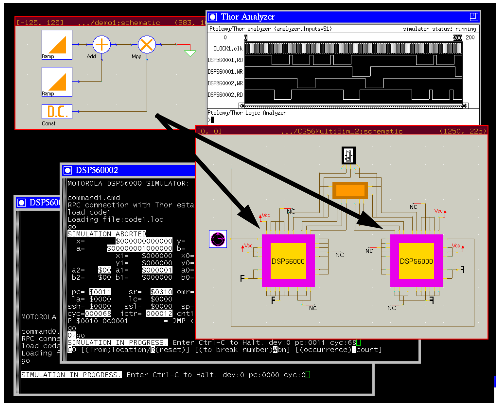


This highly multidisciplinary project addresses system-level design and implementation of signal processing systems.

Implementation of Signal Processing Systems

Hardware/ Software Synthesis

- Design of heterogeneous embedded systems.
- Design of real-time applications for workstations.
- Synthesis of software from dataflow graphs.
- System-level hardware design.
- Cosimulation of hardware/ software systems.
- Hardware/software codesign.



```
MOTOROLA DSP56000 SIMULATOR:

command1.cmd
RPC connection with Thor esta
load code1
Loading file:code1.lod
go
SIMULATION ABORTED
  x=      $000000000000  y=
  a=     $00000001000000  b=
          x1=   $000000  x0=
          y1=   $000000  y0=
 a2=   $00 a1=  $000001  a0=
 b2=   $00 b1=  $000000  b0=

 pc= $0011    sr=  $0310  omr=
 la= $0000    lc=  $0000
ssh= $0000   ssl=  $0000  sp=
cyc=000068  ictr= 000012  cnt1
P:$0010 0c0001         = JMP <
go
0>go
SIMULATION IN PROGRESS. Enter Ctrl-C to Halt. dev:0 pc:0011 cyc:68
GO [(from)location/R(reset)] [(to break number)#bn] [(occurrence):count]
```

```
MOTOROLA

command0.
RPC conne
load code
Loading f
go

SIMULATION IN PROGRESS. Enter Ctrl-C to Halt. dev:0 pc:0000 cyc:0
```

The design philosophy in Ptolemy is heterogeneous, allowing for effective use of specialized design tools within a general system-level design environment.

Annotation for this Page of Slides

The Ptolemy Project

Ptolemy is the work of many people at UC Berkeley including professors, research staff, and graduate students. This is a list of the people currently working on Ptolemy. The figure is a portrait of Claudius Ptolemaus.

System-Level Design of Signal Processing Systems

These are the main areas of research within the Ptolemy project. The figure is a phased-locked loop system for tracking laser phase noise as implemented in Ptolemy. The various graphs are an eye diagram, a scatter plot of the output samples, a graph of the phase, and a bar chart of the phase error.

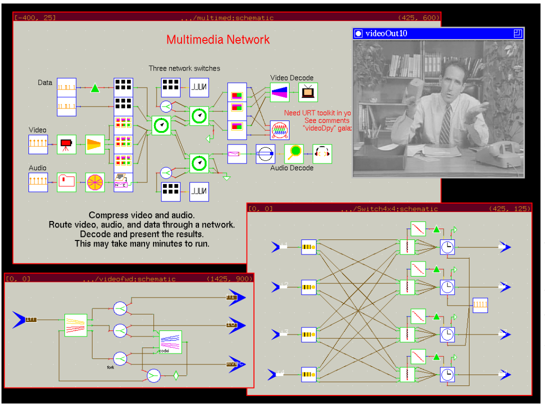
Implementation of Signal Processing Systems

Because Ptolemy is a heterogeneous design environment, specialized tools for both hardware and software design may be used to build a system level design. The figure is an example of hardware/software codesign with programmable DSP chips and custom hardware.

Heterogeneous Modeling and Design

Key Principles

- Extensible, object-oriented kernel.
- No model of computation is implemented in the kernel.
- Models of computation are implemented in modular *domains*.
- A domain interacts with another without knowing its semantics.



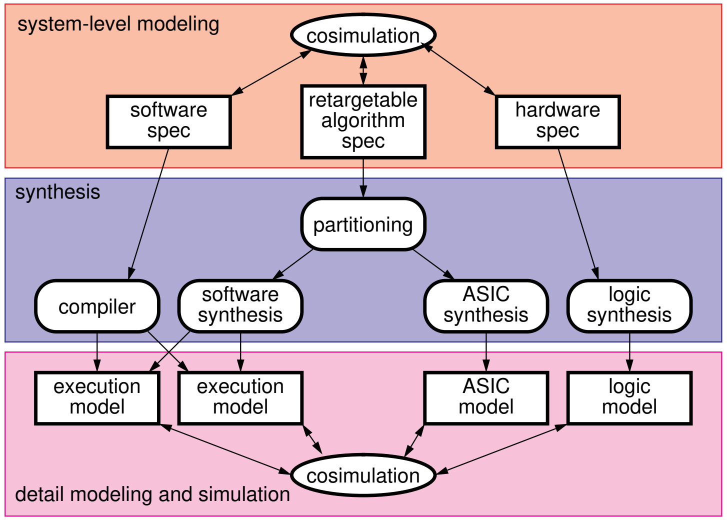
Need URT toolkit in yo
See comments
"videoDpy" gala:

Compress video and audio.
Route video, audio, and data through a network.
Decode and present the results.
This may take many minutes to run.

Application designed by Paul Haskell

Multiple models of computation may be used in the same system. Here, dataflow is used for signal processing, while a timed discrete-event system models a communication network.

Heterogeneity in System-Level Design



system-level modeling

cosimulation

software spec

retargetable algorithm spec

hardware spec

synthesis

partitioning

compiler

software synthesis

ASIC synthesis

logic synthesis

execution model

execution model

ASIC model

logic model

cosimulation

detail modeling and simulation

Ptolemy as a Tool and as a Laboratory

Ptolemy is
- **Extensible**
- **Publicly available**
- **An open architecture**
- **Object-oriented**

Allows for experiments with:
- **Models of computation**
- **Domain-specific tools**
- **Design methodology**
- **Software synthesis**
- **Hardware synthesis**
- **Cosimulation**
- **Codesign**

Rationale for heterogeneity: specialized models are
- **More useful to the system-level designer**
- **More amenable to hardware and software synthesis.**

Annotation for this Page of Slides

Heterogeneous Modeling and Design

Ptolemy allows multiple models of computation to be used in the same system. This allows the an appropriate (and possibly different) model to be chosen for each part of the system.
The figure is a multimedia network implemented in Ptolemy. A video image is compressed and sent across a packet network. The packets are then received and the video is uncompressed and displayed. Dataflow is used to perform the compression, while a discrete event system models the packet network.

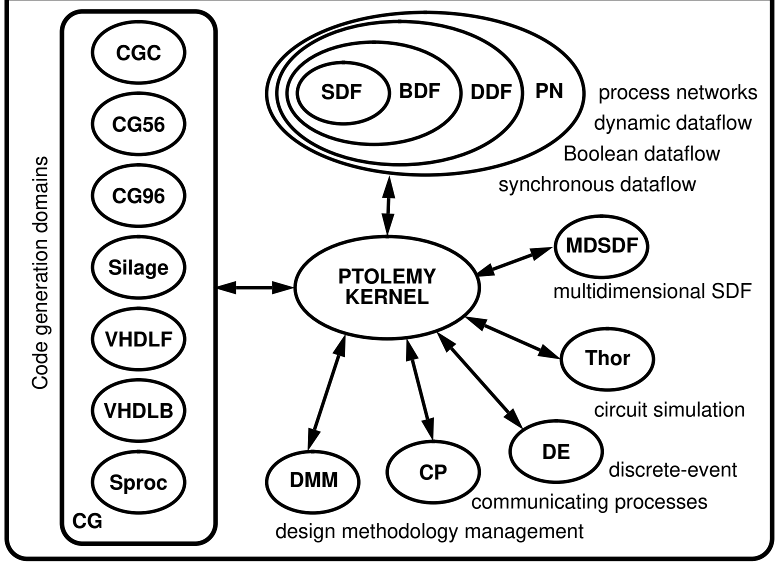
Heterogeneity in System-Level Design

Different tools and methods are needed at each stage of the design. A Heterogeneous design allows one to use appropriate tools for each area of each stage of design. For example, tools which are useful for system level modeling would not necessarily be good for detail modeling and simulation.

Ptolemy as a Tool and as a Laboratory

Ptolemy a tool used by industry and universities to build and test systems. But Ptolemy is also a laboratory for experimenting with new tools and design methodologies.

Domains in Ptolemy



Code generation domains

CGC

CG56

CG96

Silage

VHDLF

VHDLB

Sproc

CG

SDF BDF DDF PN

process networks

dynamic dataflow

Boolean dataflow

synchronous dataflow

PTOLEMY KERNEL

MDSDF

multidimensional SDF

Thor

circuit simulation

DE

discrete-event

CP

communicating processes

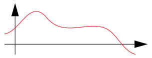
DMM

design methodology management

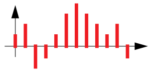
Major Activities

- Formal methods
  - **Dataflow** (process networks, synchronous, Boolean, multidimensional, ...)
  - **Control** (hierarchical FSMs, Esterel, synchronous languages, ...)
  - **Partitioning and scheduling of dataflow graphs** (optimize IPC, memory, ...)
  - **Programming languages** (higher-order functions, polymorphism, ...)
- Algorithm-level design methodology
  - **Mixing models of computation** (discrete-event, FSMs, dataflow, imperative, ...)
  - **Animation and visualization** (Tcl/Tk, Matlab, xv, ...)
  - **Mixing domain-specific tools** (filter design, Matlab, Mathematica, ...)
  - **Visual programming** (dataflow, FSMs, regularity, recursive, functional, ...)
- System-level design methodology
  - **Synthesis of embedded software** (high-level, assembly, ...)
  - **Design complexity management** (data, tool, flow, methodology, ...)
  - **Hardware/software codesign** (DesignMaker, GCLP partitioning, ...)
  - **Architecture design and performance modeling** (OT principle, VHDL, ...)

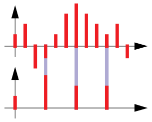
Modeling Discrete-Event Systems



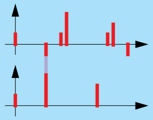
continuous time



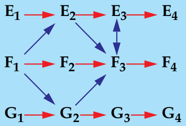
discrete time



multirate discrete time



totally-ordered discrete events



partially-ordered discrete events

Annotation for this Page of Slides

Domains in Ptolemy

Each domain in Ptolemy has its own model of computation. This slide shows the relationships of the various domains in Ptolemy to each other and how they are all linked to the Ptolemy kernel. Note that some domains may be thought of as supersets of other domains (as BDF is a superset of SDF).

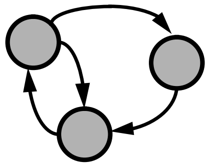
Major Activities

This is a list of the major activities within the Ptolemy group.

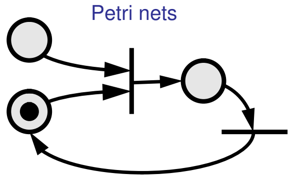
Modeling Discrete-Event Systems

There are many different models for discrete-event systems. This slide is a quick overview of some common discrete event models.

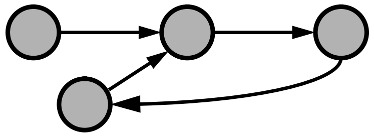
Some Discrete-Event Models



finite state machines



Petri nets



process algebras
(e.g. SCCS, CSP, Kahn process networks)

. . .
- temporal logic systems
- hybrid systems

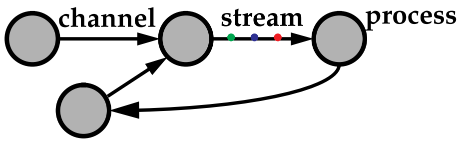
. . .

Kahn Process Networks — An Underlying Formalism

Network of concurrent sequential processes

- processes communicate via one-way FIFO channels.
- channels have unbounded capacity.
- each channel has one producer and one consumer.
- writes to channels are non-blocking.
- reads from channels are blocking.

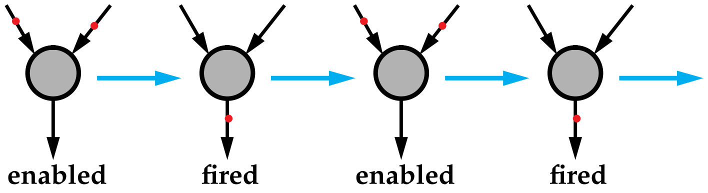
A process in the Kahn model is a mapping from one or more input sequences to one or more output sequences.



Dataflow and Kahn Process Networks

Dataflow:

- A *signal* is a sequence of *tokens*.
- An *actor* maps input tokens onto output tokens.
- A set of *firing rules* specify when an actor can fire.
- A firing *consumes* input tokens and *produces* output tokens.
- A sequence of firings is a *dataflow process*.



Annotation for this Page of Slides

Some Discrete-Event Models

This slide is a very brief overview of various discrete event models: finite state machines, Petri nets, process algebras, temporal logic systems, and hybrid systems. The Kahn process network model will be covered in the next slide.

Kahn Process Networks — An Underlying Formalism

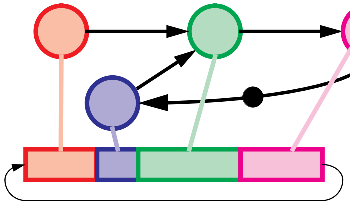
The main elements of the Kahn process networks model are listed here. More details on this model may be found in the original papers:
G. Kahn, “The Semantics of a Simple Language for Parallel Programming,” Proc. of the IFIP Congress 74, North-Holland Publishing Co., 1974.
G. Kahn and D. B. MacQueen, “Coroutines and Networks of Parallel Processes,” Information Processing 77, B. Gilchrist, editor, North-Holland Publishing Co., 1977.

Dataflow and Kahn Process Networks

This slide shows the basics of dataflow. The figure illustrates the firing sequence of a dataflow model. When tokens are available on the input arcs, the actor is enabled. The actor is then fired, producing an output token on its output arch.

Scheduling Dataflow Graphs

- A dataflow graph specifies a partial order on a (typically) *infinite number* of firings.
- Scheduling is much more manageable if we can reduce this to a *finite number* of firings.
- A periodic schedule for a single processor specifies a *total ordering* for a finite sequence of firings.
- The finite sequence of firings must return the graph to its original state to form a *complete cycle*.



Major Contributions

- **Compile-time scheduling of *synchronous dataflow* graphs with optimized partitioning and memory utilization.**
- **Specification of the *Boolean dataflow (BDF) model*, which is Turing complete.**
- **Proof that the existence of a finite complete cycle and a bounded memory implementation for BDF is *undecidable*.**
- ***Heuristics* for constructing finite complete cycles and bounded memory schedules most of the time.**
- ***Multidimensional* generalization to dataflow models.**
- ***Process network* model generalization to dataflow.**
- ***Visual programming* formulation and use of *higher-order functions*.**

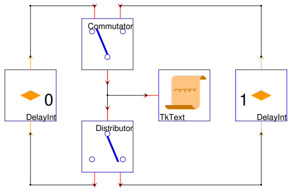
DOMAIN DEVELOPED BY THOMAS PARKS

The Process Network Domain

Generalization of dataflow

- Stars are written as concurrent sequential processes.
- Stars block on reads from inputs if data is lacking.
- A multitasking O/S is used to manage context.
- Real-time behavior is more flexible (e.g. handling periodic timing constraints).
- The order of reads and writes can be dynamic.

Produce a stream of 0's and 1's.
(from figure 2 in Kahn74)



Annotation for this Page of Slides

Scheduling Dataflow Graphs

Because dataflow graphs take infinite streams of data as their inputs, they have infinite firings of the actors. Schedules are more manageable if they can be reduced to a finite number of firings that are repeated over and over again.

Major Contributions

This slide lists the major contributions the project has made to solving scheduling issues.

The Process Network Domain

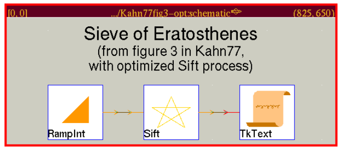
The stars in the Process Network domain run within concurrent sequential processes. This model has the attributes listed here.

Writing Stars as Processes

DOMAIN DEVELOPED BY THOMAS PARKS

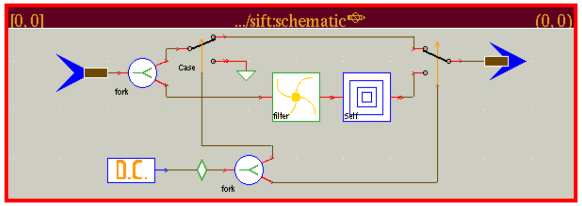
Go method for the sift star in PN:

```
go {
        StringList prime((input%0).print());
        output%0 = input%0;     // Emit a discovered prime.
        DataFlowStar* star = splice(«Filter», «input», «output», &input);
        star->setState(«prime», prime);
        star->initState();
        newProcess(star);
}
```



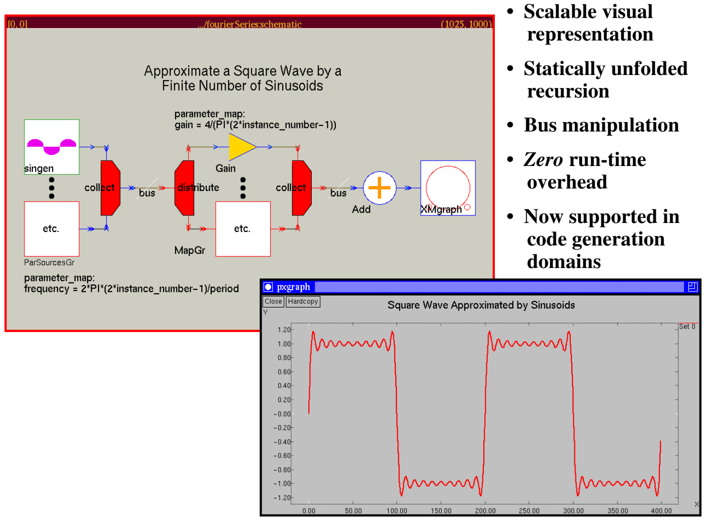
Sieve of Eratosthenes
(from figure 3 in Kahn77,
with optimized Sift process)

DDF version of the sift:



Higher-Order Functions

- Scalable visual representation
- Statically unfolded recursion
- Bus manipulation
- *Zero* run-time overhead
- Now supported in code generation domains



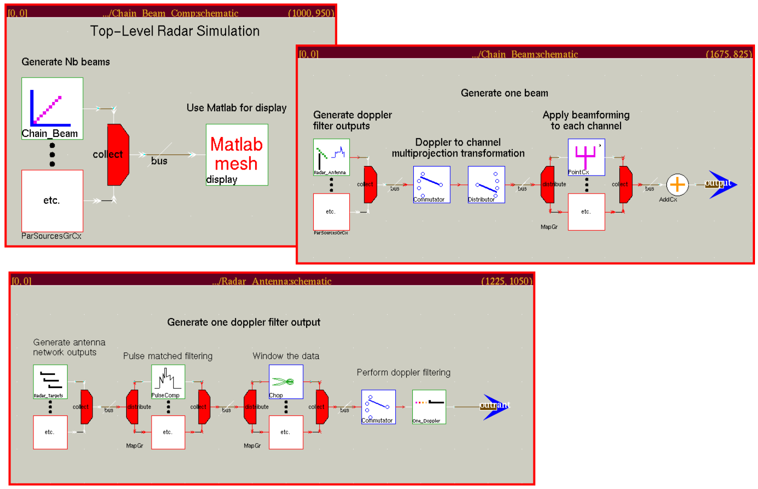
Approximate a Square Wave by a
Finite Number of Sinusoids

parameter_map:
gain = 4/(PI*(2*instance_number-1))

parameter_map:
frequency = 2*PI*(2*instance_number-1)/period

Square Wave Approximated by Sinusoids

A Major Application Using HOFs — Radar



Top–Level Radar Simulation

Generate Nb beams

Use Matlab for display

Generate one beam

Generate doppler filter outputs

Doppler to channel multiprojection transformation

Apply beamforming to each channel

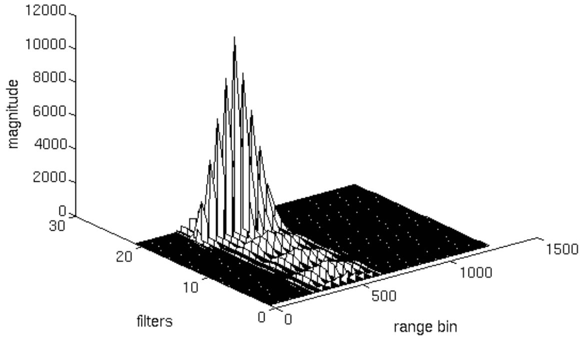
Generate one doppler filter output

Generate antenna network outputs

Pulse matched filtering

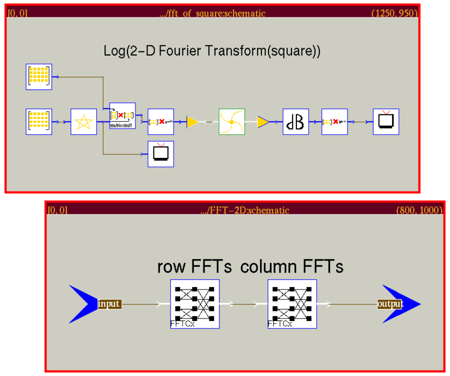
Window the data

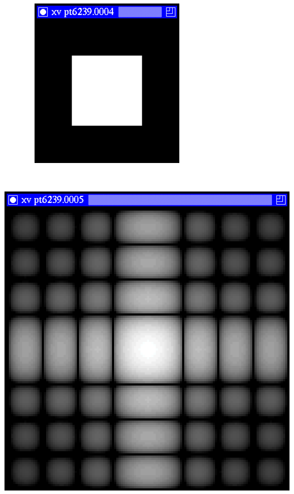
Perform doppler filtering


Pulse compression + Doppler flitering

DOMAIN DEVELOPED BY MICHAEL CHEN

Multidimensional Dataflow



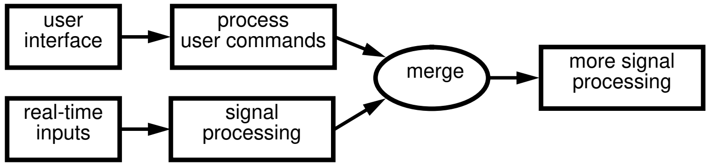


Generalize streams to multidimensional partial orderings for representing multidimensional operations.

Extending Dataflow to Model Nondeterminism

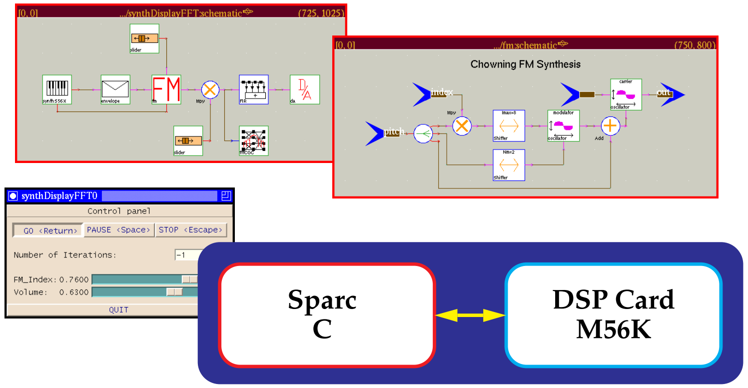
- **The Kahn model maintains determinism by not allowing processes to test an input for the presence of data.**
- **Determinism is usually, but not always, desirable.**

Example scenario where nondeterminism is needed:


user interface
process user commands
real-time inputs
signal processing
merge
more signal processing

APPLICATION DEVELOPED BY JOSÉ LUIS PINO

Heterogeneous Real-Time Prototyping



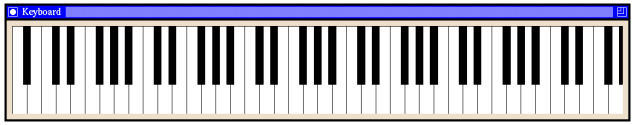
Number of Iterations: -1

FM_Index: 0.7600

Volume: 0.6300

Sparc
C

DSP Card
M56K



Annotation for this Page of Slides

Multidimensional Dataflow

In Ptolemy, streams can be generalized to multidimensional partial ordering. In the figure, Multidimensional Dataflow is used to perform a 2-D Fourier Transform. The input to the system is the white square. The output transform is shown in the bottom right of the slide.

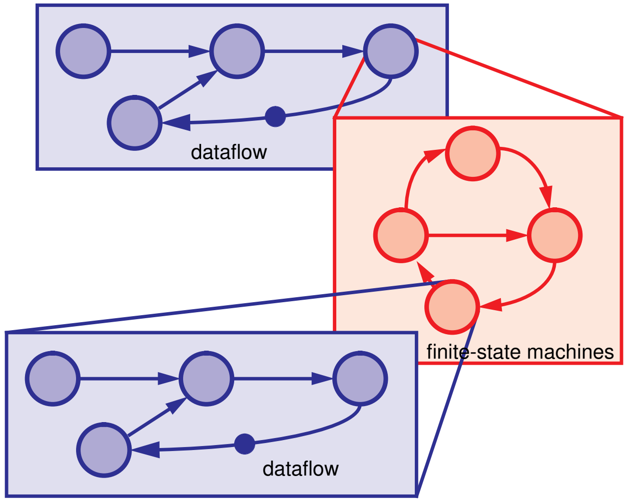
Extending Dataflow to Model Nondeterminism

The Kahn model does not allow non-determinism. However, there are times when non-determinism is desired.

Heterogeneous Real-Time Prototyping

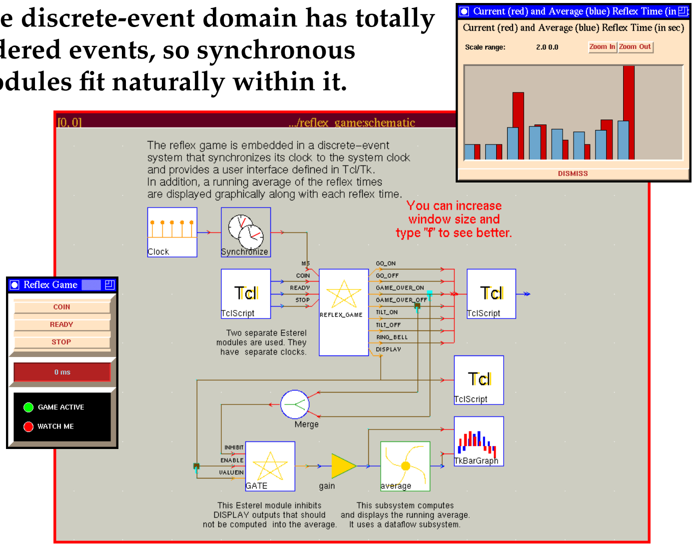
This system generates musical tones using the Chowning FM synthesis method. The system runs on two processors. The user interface is handled on a Sparc workstation, while the real time requirements of the FM synthesis are handled by a Motorola DSP card. Ptolemy generates the C code that creates the piano keyboard user interface on the Sparc, and Ptolemy generates DSP assembly code to perform the Chowning FM synthesis on the DSP card. This example uses non-determinate, non-blocking communication to isolate the real-time signal processing from the user interface.

Mixing Control-Oriented Models with Dataflow



Mixing Esterel with Ptolemy Discrete-Event Domain

The discrete-event domain has totally ordered events, so synchronous modules fit naturally within it.



The reflex game is embedded in a discrete–event system that synchronizes its clock to the system clock and provides a user interface defined in Tcl/Tk. In addition, a running average of the reflex times are displayed graphically along with each reflex time.

You can increase window size and type "f" to see better.

Two separate Esterel modules are used. They have separate clocks.

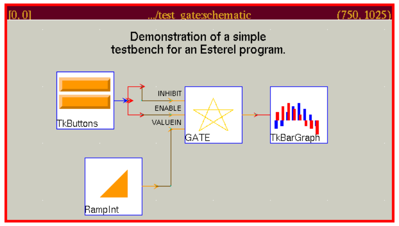
This Esterel module inhibits DISPLAY outputs that should not be computed into the average.

This subsystem computes and displays the running average. It uses a dataflow subsystem.

Mixing Esterel with Dataflow

Synchronous dataflow using TRUE/FALSE encoding of pure signals, is one option.

Compile-time analysis via the balance equations allows for multiple clocks with certain restrictions.



Demonstration of a simple testbench for an Esterel program.

Avoiding the TRUE/FALSE encoding, however, leads to some subtle issues:

- Maintaining determinacy when it is desired.
- Supporting nondeterminacy when it is desired.

Annotation for this Page of Slides

Mixing Control-Oriented Models with Dataflow

One of the research areas in Ptolemy is the combination of Finite State Machines with Dataflow graphs.

Mixing Esterel with Ptolemy Discrete-Event Domain

The figure is a Reflex Game system written in Esterel and incorporated into the Discrete-Event domain in Ptolemy. It demonstrates that Esterel can be integrated into the DE domain.

Mixing Esterel with Dataflow

There are several considerations when combining Esterel with Dataflow. The figure is a simple test bench for Esterel programs with dataflow.

Complexity Management in System-level Design

Need a systematic mechanism to define and execute design methodologies in a flexible and configurable way.

Sources of Complexity at the “system” level:

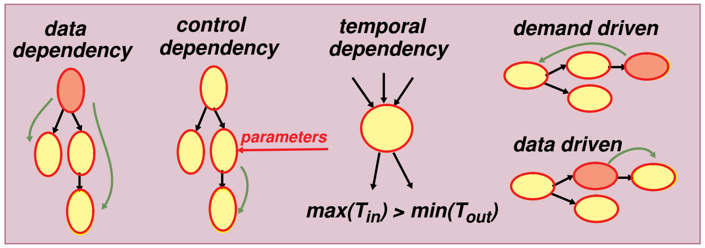
- **Design Abstraction (Data management, Tool Management)**
  - **behavioral level specification, a range of data types and tools are involved.**
- **Dynamic Flow Execution (Flow Management)**
  - **complex dependencies and run-time flow determination.**
- **Design Options (Methodology Management)**
  - **possible design options increase, no pre-defined methodology.**

Design Space Exploration is the key issue.

Design flow is an integral part of the design itself.

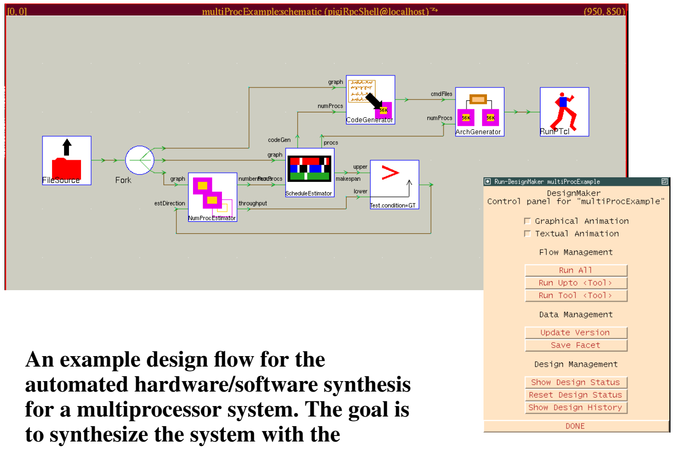
Our Solution to Complexity Management

- **Design Methodology Management (DMM):** **DMM Domain in Ptolemy, graphical representation of design flow.**
- **Task Management:** **Stars encapsulate tools.**
- **Data Management:** **Portholes communicate via file names.**
- **Flow Management:** **complex dependencies, automatic flow execution.**
- **Execution Environment:** **Target manages design, Tcl/Tk based GUI.**



DOMAIN DEVELOPED BY ASAWAREE KALAVADE

Example Design Flow: Multiprocessor Synthesis



An example design flow for the automated hardware/software synthesis for a multiprocessor system. The goal is to synthesize the system with the minimum number of processors such that required throughput is met.

APPLICATION DEVELOPED BY ASAWAREE KALAVADE

Annotation for this Page of Slides

Complexity Management in System-level Design

This slide lists the sources of complexity in system-level design.

Our Solution to Complexity Management

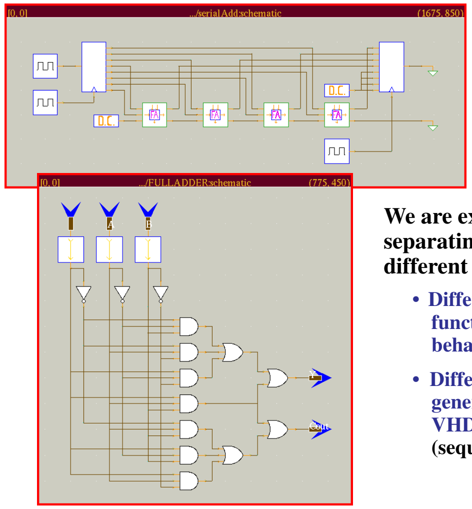
A design methodology management (DMM) domain has been developed in Ptolemy to manage the complexity of the system-level design process. In this framework, the design flow is specified graphically, where the nodes represent tools and arcs represent dependencies. Features of the DMM domain, including various dependencies and scheduling mechanisms, are listed here.

Example Design Flow: Multiprocessor Synthesis

The figure shows an example design flow specified in the DMM domain. Source specifies the application for which the system is to be synthesized. The Estimator-Scheduler-Comparator loop is iteratively repeated until it converges. The Estimator estimates the number of processors required to implement the system with the desired latency. The Scheduler computes the actual makespan using this estimated number of processors. The Comparator compares the makespan with desired latency to provide feedback to the estimator. Once the number of processors is fixed, the Code Generator generates assembly code for the processors, and the Architecture Generator synthesizes a netlist for the system. The combined hardware/software system is simulated by the RunPTcl block.

The lower figure shows the control panel for the DMM domain, with options for flow, data, and methodology management.

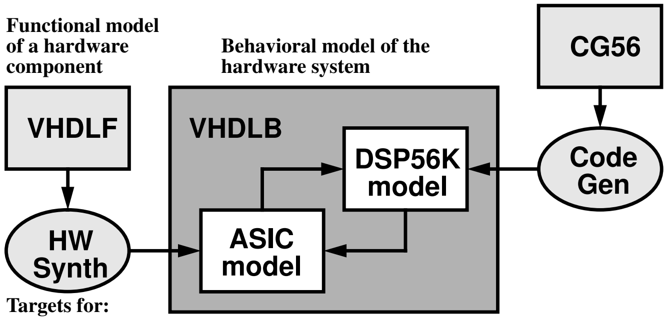
VHDL Code Generation



We are experimenting with separating conceptually different uses of VHDL:

- **Different Domains are used for functional modeling and behavioral modeling.**
- **Different Targets are used for generating different styles of VHDL from the same source (sequential, synthesizable,...).**

Using VHDL with Multiple Design Styles



Embedded software models (dataflow, FSMs)

Functional model of a hardware component

Behavioral model of the hardware system

CG56

VHDLF

VHDLB

DSP56K model

Code Gen

HW Synth

ASIC model

Targets for:
- fast simulation
- high-level synthesis
- logic synthesis

Activities during the Next Year

- **Software infrastructure**
  - **Generalize Wormhole mechanism with dynamic switching.**
  - **Redesign the code generation mechanism for better retargeting.**
  - **Improve support for scripted runs.**
  - **Evaluate and install CGC fixed-point capability.**
  - **Implement visual recursion.**
  - **Generalize type checking and automatic type conversion.**
  - **Introduce an enumerated parameter type.**
  - **Enhance interactive graphics.**
- **Heterogeneous design**
  - **Design of a hierarchical finite-state machine domain for control.**
  - **Embed controllers designed using Esterel.**
  - **Model system-level performance for heterogeneous hardware.**
  - **Complete the design methodology management domain.**

Annotation for this Page of Slides

VHDL Code Generation

The top figure is a serial adder in Ptolemy VHDL. The lower figure is one of the full adders used to implement the serial adder.

Using VHDL with Multiple Design Styles

We are experimenting with separating conceptually different uses of VHDL. In the figure, VHDLF stands for VHDL for functional descriptions while VHDLB stands for VHDL for behavioral modeling.

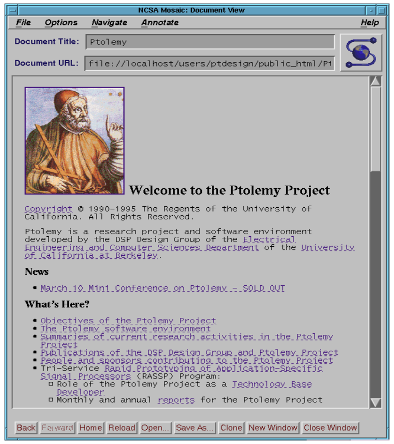
Activities during the Next Year

This is a list of the planned activities of the Ptolemy group during the next year.

Activities During the Next Year (cont.)

- **Models of computation**
  - **Explore interacting semantics of synchronous languages and dataflow.**
  - **Generalize multidimensional dataflow and explore synthesis issues.**
  - **Apply Boolean dataflow technology to VHDL-based synthesis.**
  - **Develop a process network domain that supports nondeterminacy.**
  - **Add enhancements from the Navy's PGM to dynamic dataflow domain.**
  - **Implement and understand cyclo-static dataflow.**
- **Applications**
  - **InfoPad**
  - **Radar**
  - **Image processing**
  - **Communications**
- **Algorithm-level design**
  - **Interface to Mathematica for symbolic computation.**
  - **Integrate system rewriting in Mathematica with DMM domain.**

World Wide Web Server



Document Title: Ptolemy

Document URL: file://localhost/users/ptdesign/public_html/Pt

Welcome to the Ptolemy Project

Copyright © 1990-1995 The Regents of the University of California. All Rights Reserved.

Ptolemy is a research project and software environment developed by the DSP Design Group of the Electrical Engineering and Computer Sciences Department of the University of California at Berkeley.

News

- March 10 Mini Conference on Ptolemy - SOLD OUT

What's Here?

- Objectives of the Ptolemy Project
- The Ptolemy software environment
- Summaries of current research activities in the Ptolemy Project
- Publications of the DSP Design Group and Ptolemy Project
- People and sponsors contributing to the Ptolemy Project
- Tri-Service Rapid Prototyping of Application-Specific Signal Processors (RASSP) Program:
  - Role of the Ptolemy Project as a Technology Base Developer
  - Monthly and annual reports for the Ptolemy Project

Included in the server

- RASSP-specific entry point.
- Complete distribution of version 0.5.1, including all source code.
- Distribution of Ptiny Ptolemy, a small, easy-to-install demonstration version.
- An evolving quick tour of Ptolemy with animations of simple demos.
- *The Almagest*, a four-volume manual for Ptolemy, in PostScript form.
- User's manual in hypertext form.
- Publications listing from the Ptolemy group, with many of the papers in PostScript and/or hypertext form.
- Keyword searching for publications.
- Directory of project participants and sponsors.
- Copy of the FAQ and information about mailing lists and newsgroups.

http://ptolemy.eecs.berkeley.edu

Annotation for this Page of Slides

Activities During the Next Year (cont.)

This slide continues the list of planned activities for the Ptolemy project.

World Wide Web Server

Ptolemy now has a world wide web page. It is available through the RASSP pages and is also available directly by pointing a web browser (such as Mosaic) to
http://ptolemy.eecs.berkeley.edu
The figure is a screen dump of the Ptolemy WWW home page.