Algorithm Analysis and Mapping Environment for Adaptive Computing Systems

Cory Myers, Eric Pauer, Ken Smith and Paul Fiore

Sanders, A Lockheed Martin Company

Statement of the Problem

Reconfigurable computing technology offers leap ahead performance, e.g. 10X ops per watt and/or ops per cubic inch, over general purpose programmable solutions without the need to develop custom hardware. However, today generation of a working implementation requires hardware design expertise and generation of a good implementation requires many slow iterations between an algorithm designer and a hardware developer.

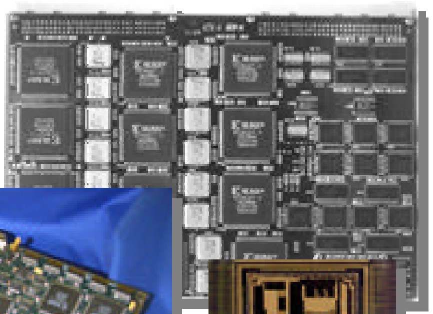
Adaptive Computing Performance Gain

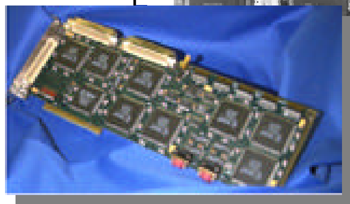
	CHAMP	TMS320 C80
Image Size	256 x 256	256 x 256
Implementation Time	44 Days	28 Days
Frame Rate	305 frames/sec	12 frames/sec
Latency	68 μsec	82,000 μsec
Processing Load	4.7 Bops	0.2 Bops
Utilization	73%	Unknown
Gates	510k	N/A

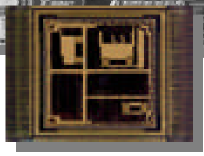
(Operation count increase by 70% if memory loads and stores are counted)

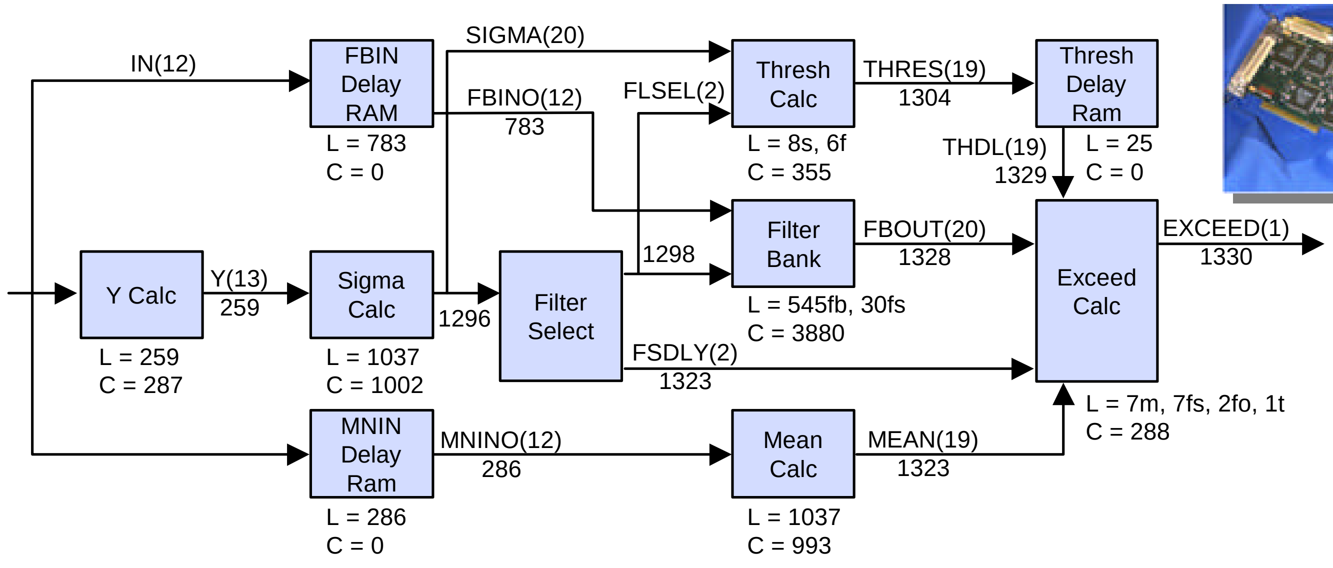
Greater than 10X performance
Design time measured in weeks

Reconfigurable Architectures


CHAMP

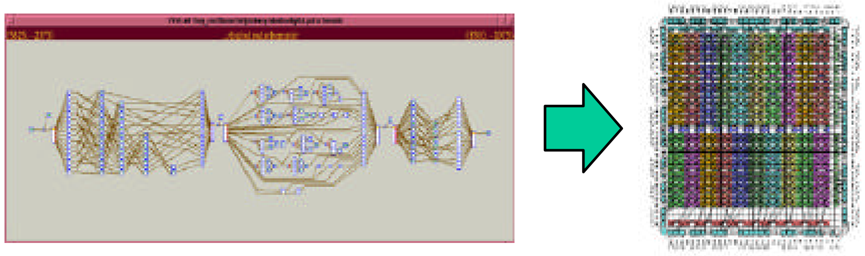

RCP


CHAMP 2 MCM



Analysis:
Bit Widths
Latency (L)
Cells Used (C)

Algorithm Analysis and Mapping Environment for Adaptive Computing Systems



Direct mapping of algorithm to adaptive computing system implementation.

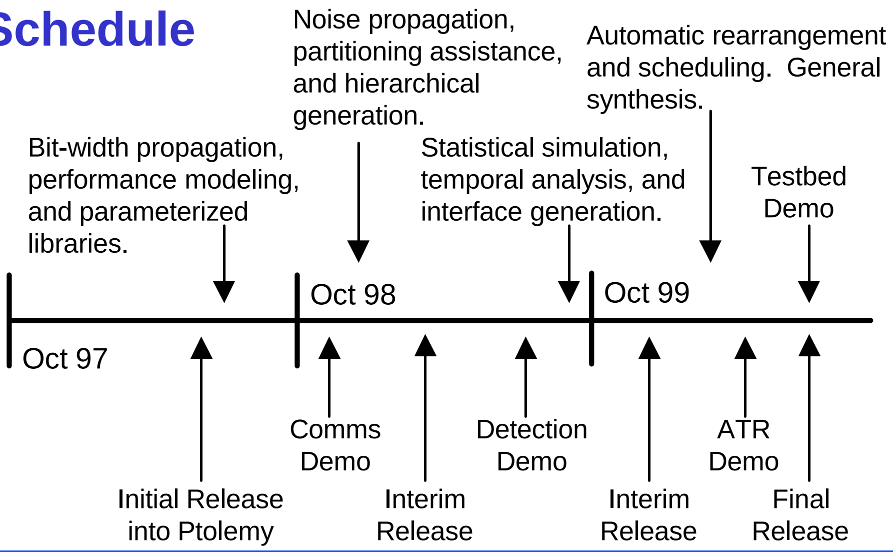
New Ideas

- **Support algorithm analysis at the bit level**
  - Combine analytical, symbolic, and simulation methods
  - Provide feedback on implementation implications
- **Leverage structure of signal processing domain**
  - Coarse-grain dataflow enhances partitioning
  - Scheduling tools used for implementation of sequencers
- **Integrate optimized generators for low level functions, high level functions, and interfaces**

Impact

- **Demonstrate an order of magnitude reduction in development time for mapping military signal processing algorithms to adaptive computing systems**
- **Bit-level analysis and implementation of algorithms reduces space and power of computing by a factor of from two to ten**
- **Algorithm analysis and mapping capabilities enable adaptive computing systems to be accessible directly to algorithm developers**

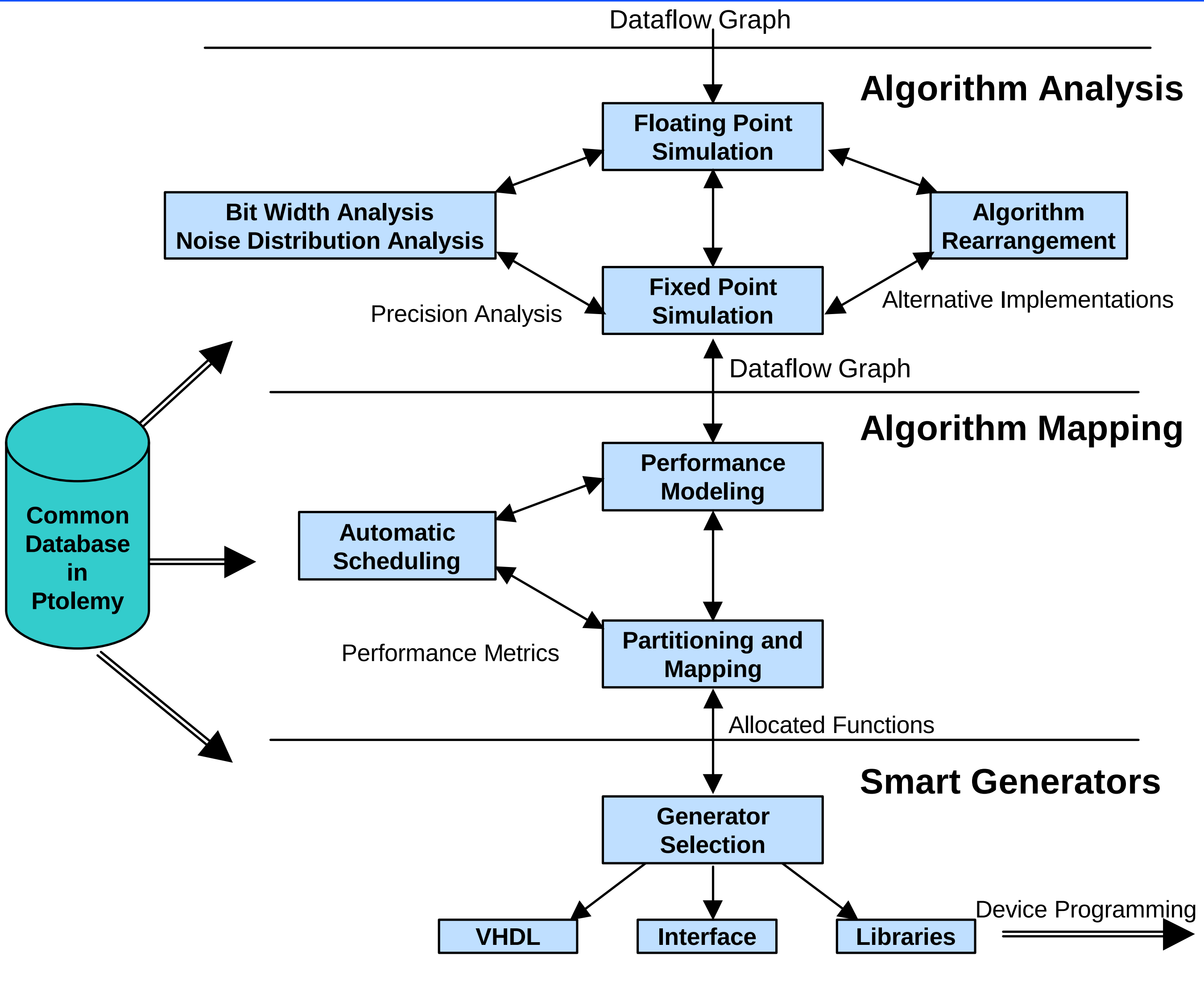
Schedule


Noise propagation, partitioning assistance, and hierarchical generation.
Automatic rearrangement and scheduling. General synthesis.
Bit-width propagation, performance modeling, and parameterized libraries.
Statistical simulation, temporal analysis, and interface generation.
Testbed Demo
Oct 97
Oct 98
Oct 99
Comms Demo
Detection Demo
ATR Demo
Initial Release into Ptolemy
Interim Release
Interim Release
Final Release

State of the Art

Tools for Mapping Signal Processing Algorithms to FPGAs						
Category	Examples	Algorithm Trades	Fixed Point Analysis	Performance Analysis	Logic Generation	Summary
Algorithm Design Environments	• Matlab • Khoros	+ Low level and high level building blocks + Rapid simulation – Alternative representations not explicitly supported	– Little built-in support – Requires algorithm re-entry	+ Operation counts can be measured – No prediction of hardware implications	– Not supported	+ Strong algorithm development support – Mapping to FPGAs not directly supported
Synthesis Tools	• Synopsis • Synplicity	– No built-in support for signal processing	– No built-in support	+ Hardware implications are directly calculated	+ Excellent support for RTL level design – Explicit clock and control signals required – Behavioral synthesis not generally accepted	+ Strength in logic generation – Weak algorithm development support
DSP Tools for Hardware Design	• HPADS • SPW • DSP Canvas • SystemView	+ Rapid simulation – Low level building blocks – Alternative representations not explicitly supported	+ Built-in support	+ Some prediction of hardware implications	+ "RTL-ish" building blocks directly synthesized – Explicit control signals often required	+ Strong at mapping low level algorithms – Moderate algorithm development support
C to FPGA Compilers	• Active Area of Research	– No built-in support for signal processing	– No built-in support	Area of research	+ Direct mapping of software to hardware – Logic generation oriented at the expression level	+ General-purpose approach – Not targeted to signal processing
Our Approach		+ Support both low level and high level signal processing blocks + Support alternative representations	+ Built-in support	+ Predict hardware implications	+ Support synthesis directly from high level algorithm representation	+ Combine best of existing tools with direct synthesis from algorithm representation

Analysis and Mapping in ACS Environment



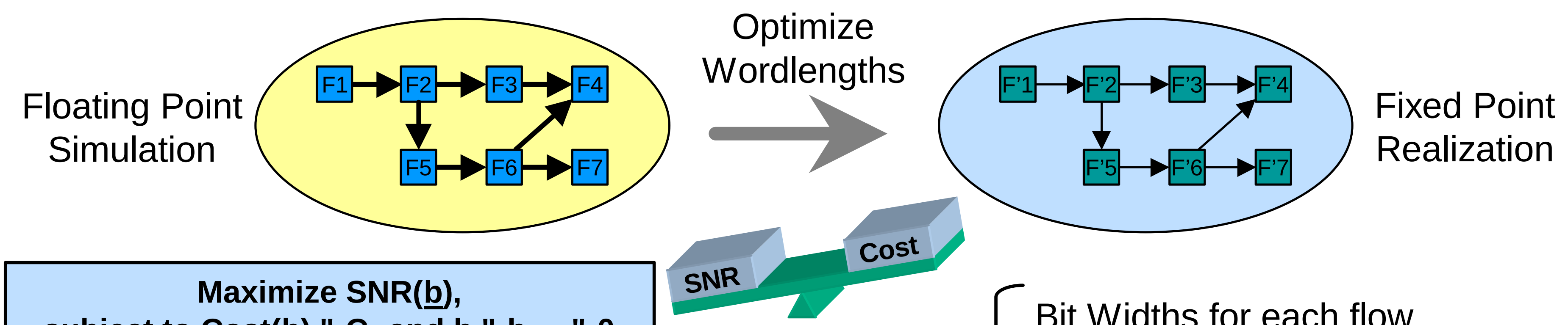
- SNR analysis
- Alternative implementations
- Functional approximations

- Timing and sizing estimation
- Scheduling – FSM and contexts
- Partitioning within a resource node

- Device program
- Interface program

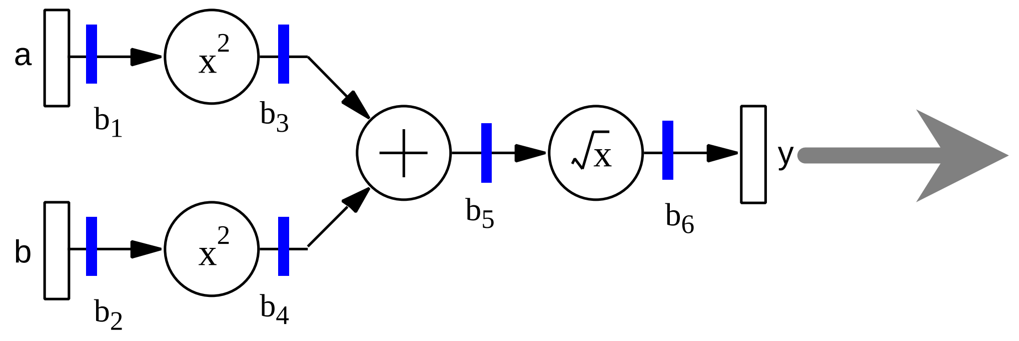
Adaptive Computing Resource

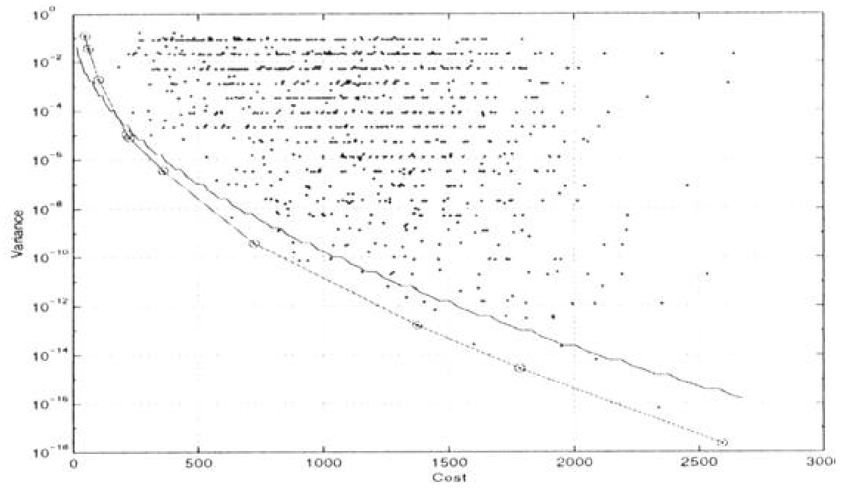
Automated Float to Fixed Point Translation



Maximize SNR(b),
subject to Cost(b) " C_0 and b " b_{min} " 0
or
Minimize Cost(b),
subject to SNR(b) " SNR_0 and b " b_{min} " 0

Yields
- Bit Widths for each flow
- Cost estimates (area/complexity)
- Quantization Noise (SNR)

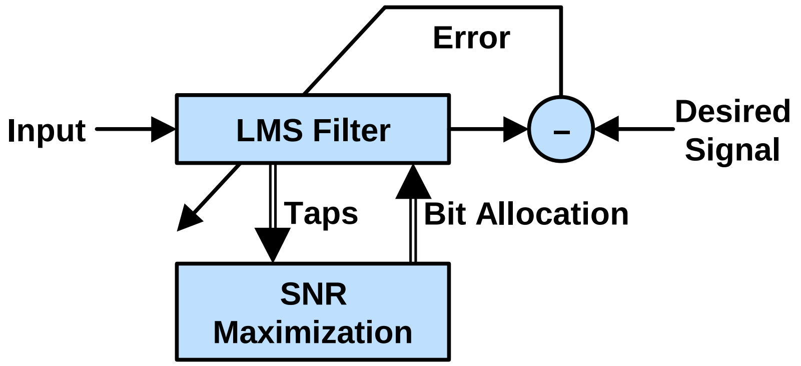


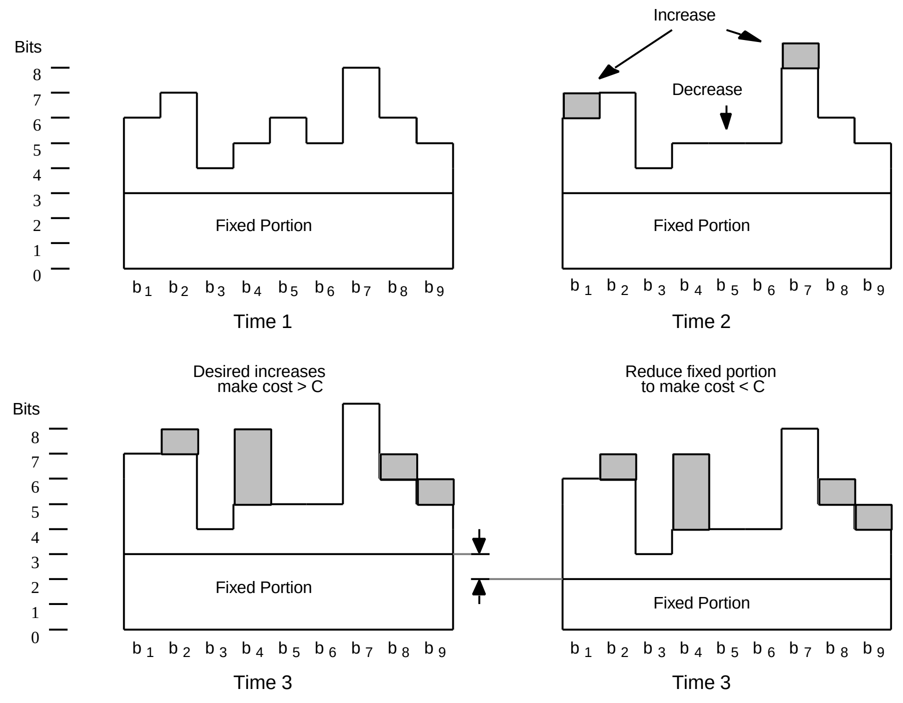


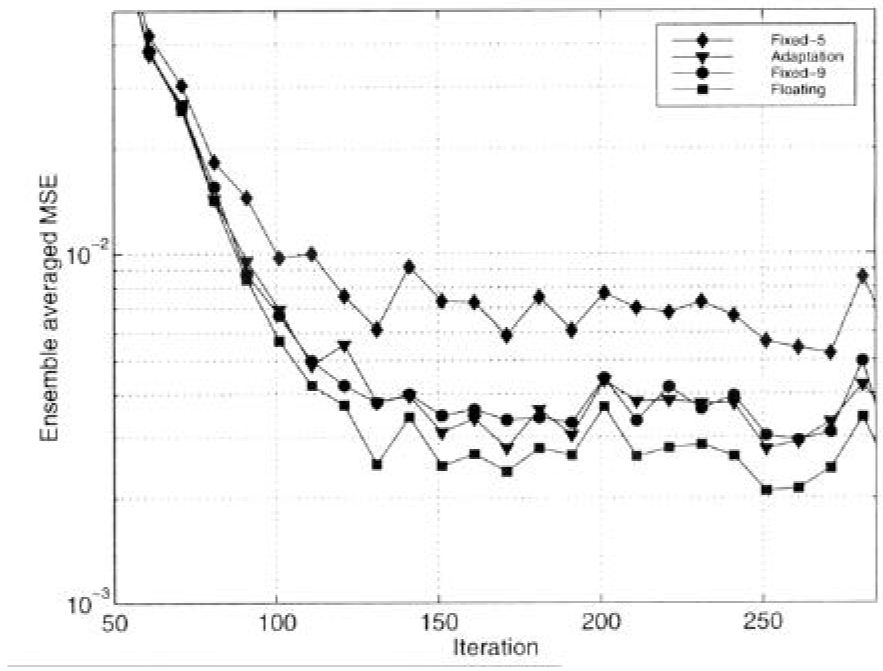
Dynamic Wordlength Adaptation

Maximize SNR(b[t]),
subject to Cost(b[t]) " C_0 and b[t] " b_{min} " 0

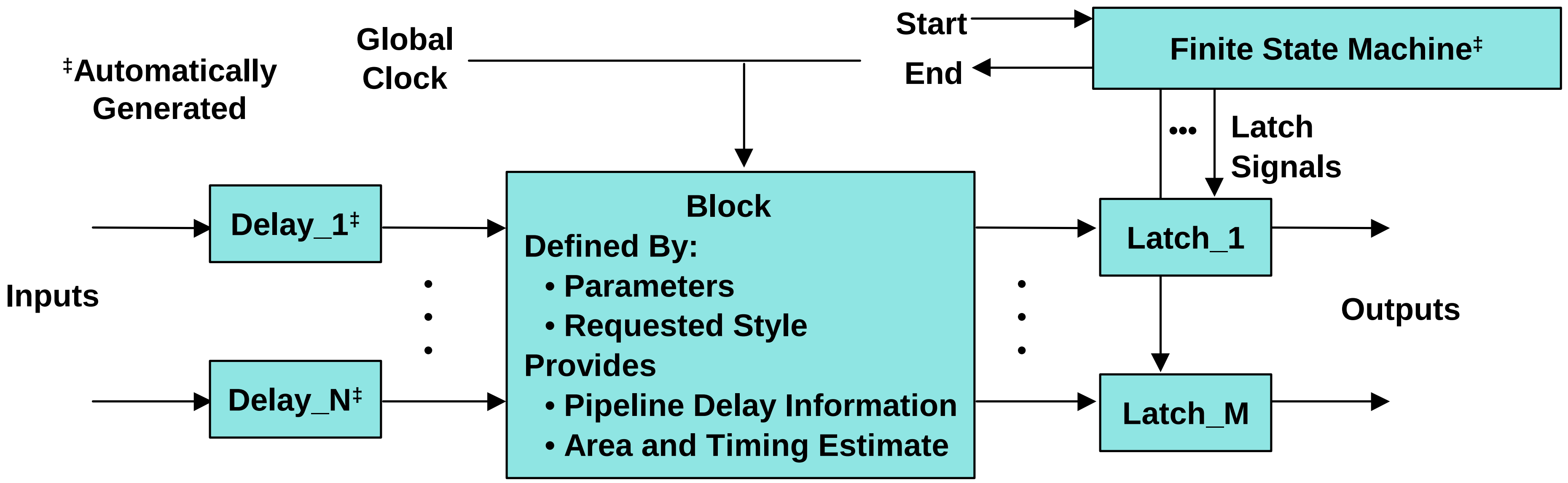
Fixed plus time-varying solution







Target Architectures



Today
- Uni-Rate Synchronous Dataflow
- Single Reprogrammable Device
- Fully Pipelined Processing
- Automatic Pipeline Alignment
- Automatic Controller Generation
- Memory-Based I/O
- Data Stream Multiplexing
- One-to-One Mapping of Functions to Blocks

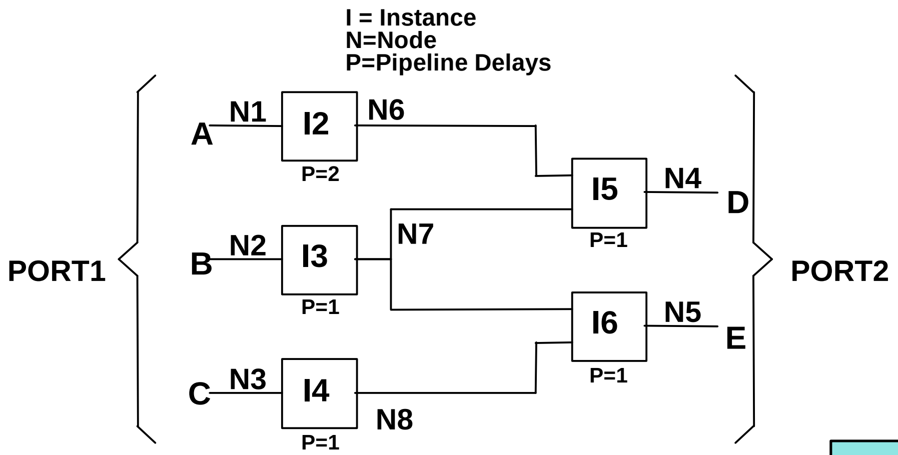
→

Future Additions
- Multi-Rate Dataflow
- Interconnected Devices
- Dynamic Reconfiguration
- Asynchronous Processing
- FIFO and Sensor Interfaces
- Many-to-One Mapping of Functions to Blocks

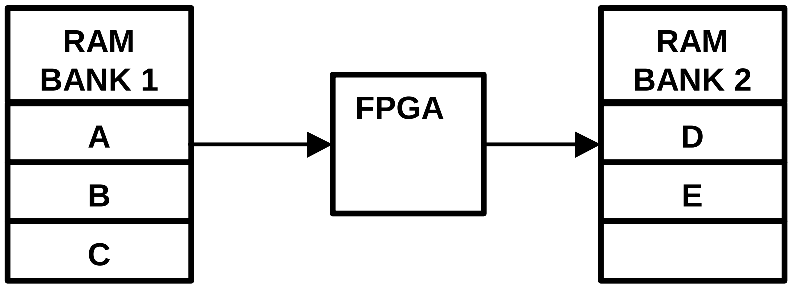
Automatic Scheduling

Pipeline alignment and schedule determination required for logic synthesis

Input

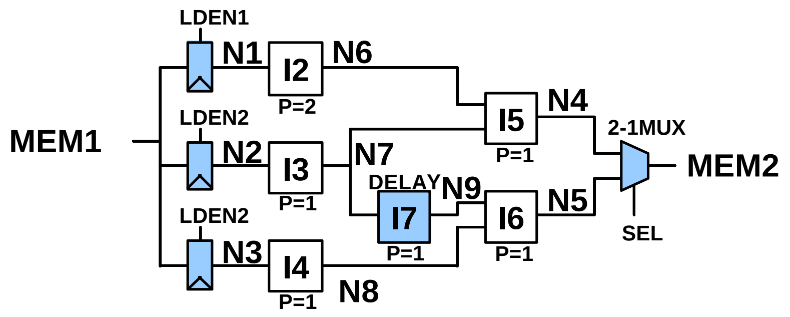


THE ALGORITHM DATAFLOW GRAPH



DATAPATH AND VARIABLE LOCATIONS

Output



🔹 ADDED TO NETLIST BY SEQUENCER GENERATOR

MODIFIED ALGORITHM DATAFLOW GRAPH

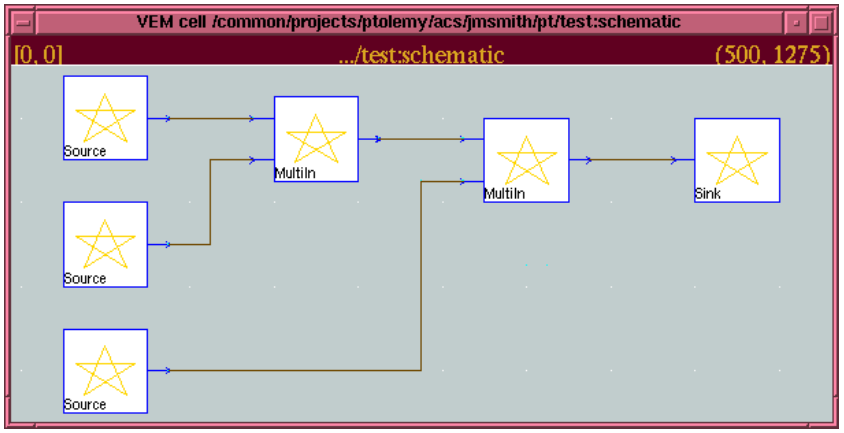
Node	Activation Sequence															
N1		🟦			🟥			🟪			🔹					
N2			🟦			🟥			🟪			🔹				
N3				🟦			🟥			🟪			🔹			
N4					🟦			🟥			🟪			🔹		
N5						🟦			🟥			🟪			🔹	
N6				🟦			🟥			🟪			🔹			
N7				🟦			🟥			🟪			🔹			
N8					🟦			🟥			🟪			🔹		
N9					🟦			🟥			🟪			🔹		
SEL					🟦	🟦		🟥	🟥		🟪	🟪		🔹	🔹	
LD1	🟦			🟥			🟪			🔹						
LD2		🟦			🟥			🟪			🔹					
LD3			🟦			🟥			🟪			🔹				
PORT1	🟦	🟦	🟦	🟥	🟥	🟥	🟪	🟪	🟪	🔹	🔹	🔹				
PORT2					🟦	🟦		🟥	🟥		🟪	🟪		🔹	🔹	

FINAL ALGORITHM SCHEDULE

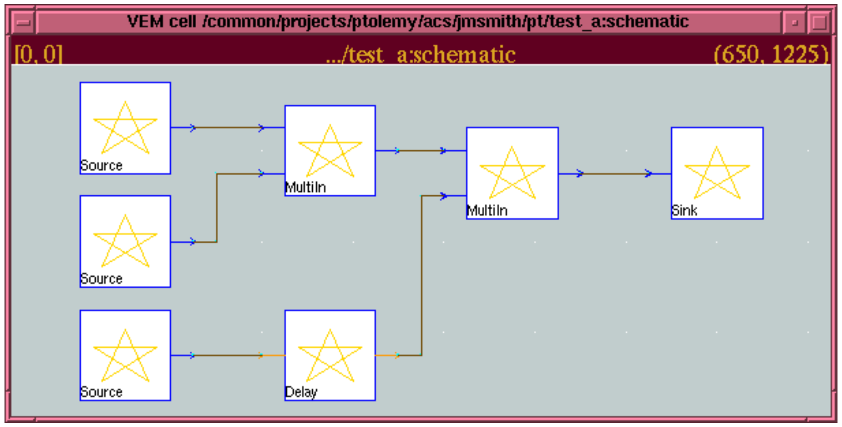
Scheduler Implementation

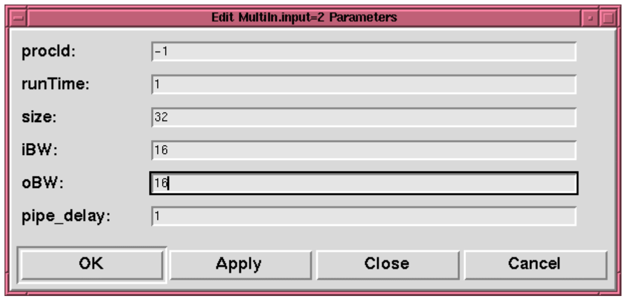
- **Implemented in existing Ptolemy CGC domain.**
- **Parameterizable scheduling blocks support algorithm testing**

Input



Output

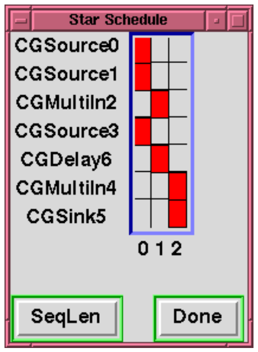




Edit MultiIn.input=2 Parameters

procId:	-1
runTime:	1
size:	32
iBW:	16
oBW:	16
pipe_delay:	1

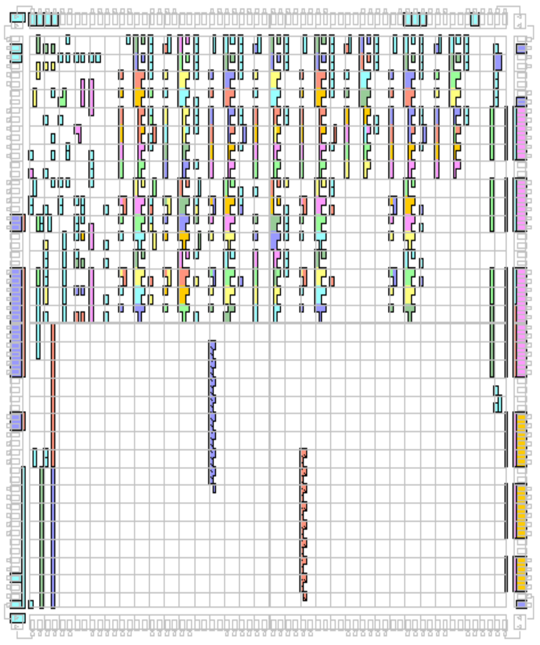
Input Parameters



Output Schedule

Benefits of Function-Specific Implementations

Before

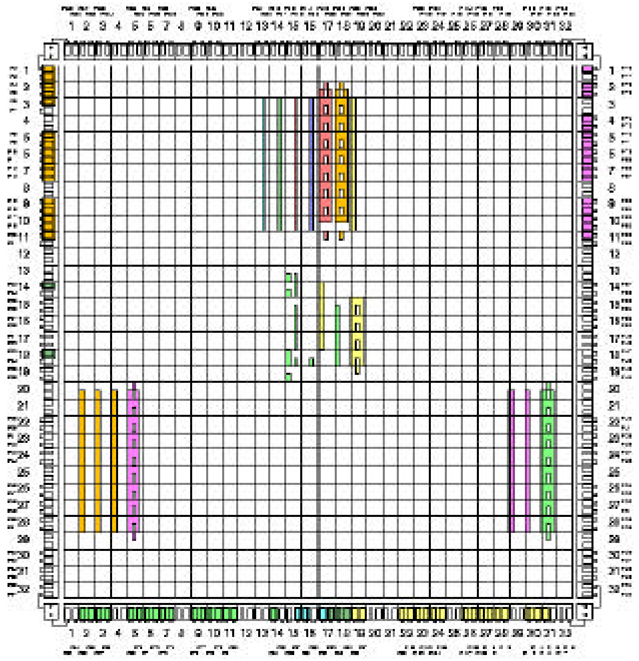


Floor Plan

Improvements

- **Algorithm-specific address generator**
- **Algorithm-specific sequence generator**
- **Reduced overhead from 50% of Xilinx 4025 to 10%**
- **Final design is 1/3 the area of original design**
- **Supports multiple memories rather than a single memory**
- **Support arbitrary number of logical ports rather than previous limit of three ports**

After

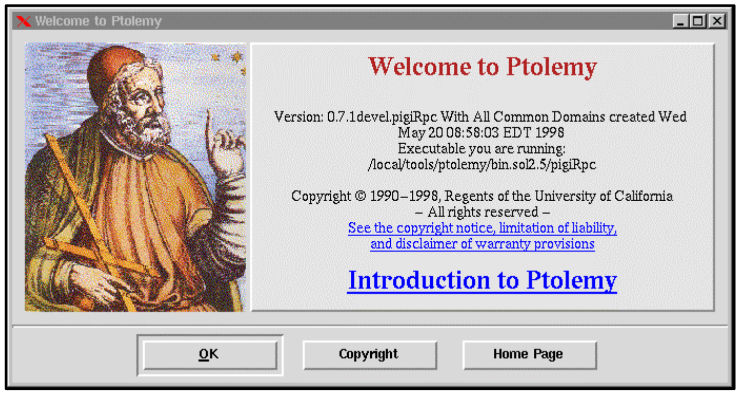


Floor Plan

From General-Purpose to Function-Specific

Ptolemy and the ACS Domain

- Ptolemy - simulation/design environment from the University of California, Berkeley (http://ptolemy.eecs.berkeley.edu)
- New ACS domain developed to facilitate movement among simulation and code/design generation (released in 0.7.1, 6/98)
- ACS Stars (basic building block) are composed of a Corona (interface) and multiple cores (implementations)
- Core (implementation) selection is via targeting mechanism

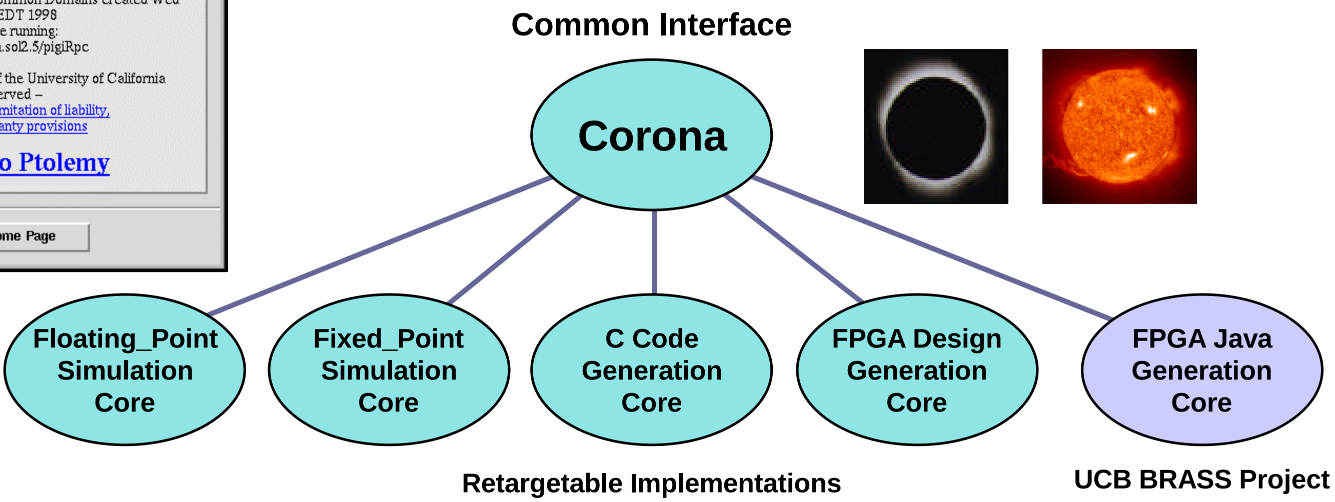


Welcome to Ptolemy

Version: 0.7.1devel.pigiRpc With All Common Domains created Wed May 20 08:58:03 EDT 1998
Executable you are running:
/local/tools/ptolemy/bin.sol2.5/pigiRpc

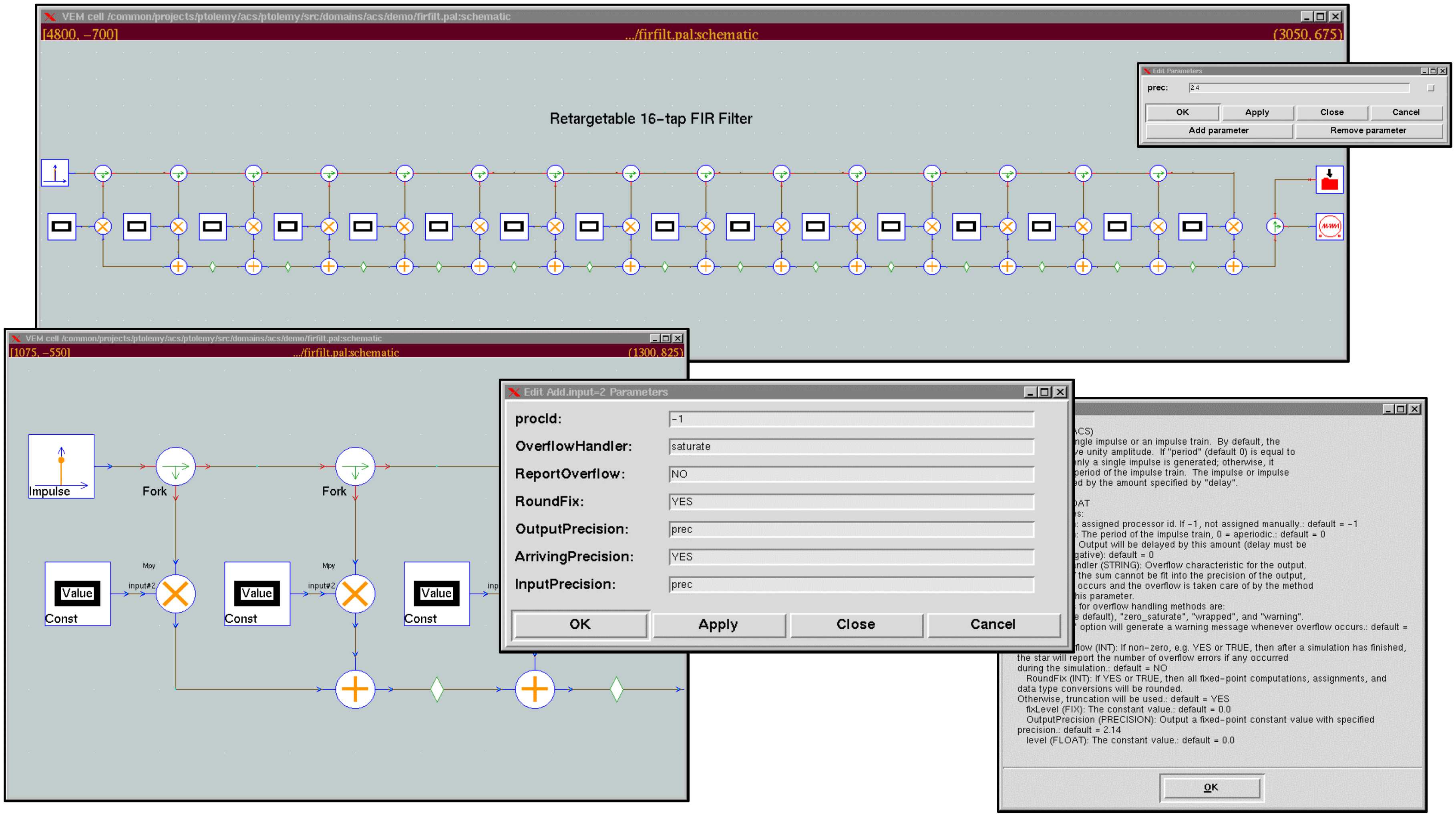
Copyright © 1990–1998, Regents of the University of California
– All rights reserved –
See the copyright notice, limitation of liability, and disclaimer of warranty provisions

Introduction to Ptolemy


Common Interface
Corona
Floating_Point Simulation Core
Fixed_Point Simulation Core
C Code Generation Core
FPGA Design Generation Core
FPGA Java Generation Core
Retargetable Implementations
UCB BRASS Project

Top Level Example

- **FIR filter to be implemented in both floating point and fixed point simulation**



VEM cell /common/projects/ptolemy/acs/ptolemy/src/domains/acs/demo/firfilt.pal:schematic
[4800, -700]
.../firfilt.pal:schematic
(3050, 675)
Retargetable 16-tap FIR Filter
Edit Parameters
OK
Apply
Close
Cancel
Add parameter
Remove parameter
VEM cell /common/projects/ptolemy/acs/ptolemy/src/domains/acs/demo/firfilt.pal:schematic
[1075, -550]
.../firfilt.pal:schematic
(1300, 825)
Impulse
Fork
Fork
Mpy
Mpy
input#2
input#2
inp
Value
Const
Value
Const
Value
Const
Edit Add.input=2 Parameters
OK
Apply
Close
Cancel
OK

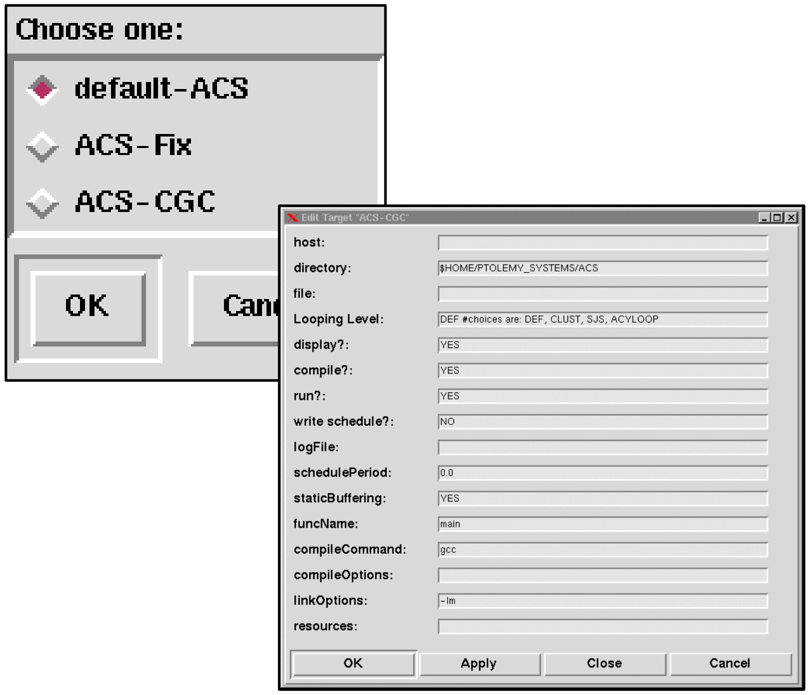
prec: 2.4

procId:	-1
OverflowHandler:	saturate
ReportOverflow:	NO
RoundFix:	YES
OutputPrecision:	prec
ArrivingPrecision:	YES
InputPrecision:	prec

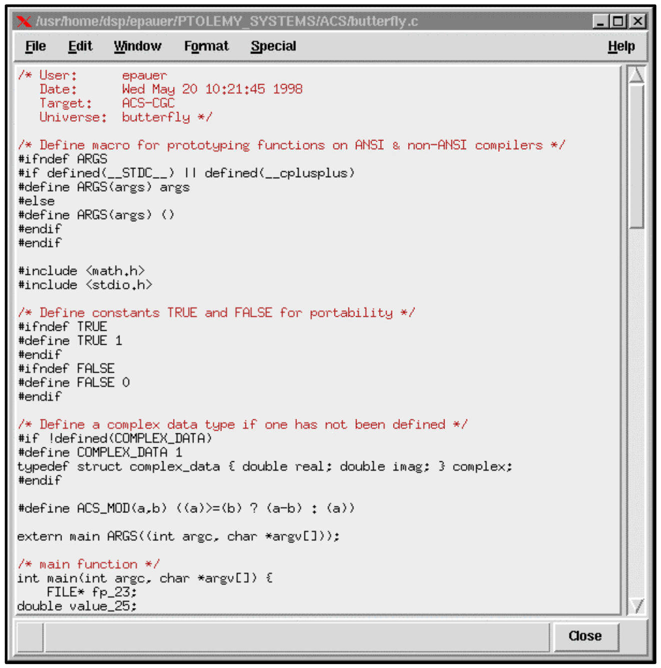
```
ACS)
ngle impulse or an impulse train.  By default, the
ve unity amplitude.  If "period" (default 0) is equal to
nly a single impulse is generated; otherwise, it
period of the impulse train.  The impulse or impulse
ed by the amount specified by "delay".

OAT
es:
: assigned processor id. If -1, not assigned manually.: default = -1
: The period of the impulse train, 0 = aperiodic.: default = 0
 Output will be delayed by this amount (delay must be
gative): default = 0
ndler (STRING): Overflow characteristic for the output.
f the sum cannot be fit into the precision of the output,
 occurs and the overflow is taken care of by the method
his parameter.
 for overflow handling methods are:
e default), "zero_saturate", "wrapped", and "warning".
' option will generate a warning message whenever overflow occurs.: default =

flow (INT): If non-zero, e.g. YES or TRUE, then after a simulation has finished,
the star will report the number of overflow errors if any occurred
during the simulation.: default = NO
   RoundFix (INT): If YES or TRUE, then all fixed-point computations, assignments, and
data type conversions will be rounded.
Otherwise, truncation will be used.: default = YES
   fixLevel (FIX): The constant value.: default = 0.0
   OutputPrecision (PRECISION): Output a fixed-point constant value with specified
precision.: default = 2.14
   level (FLOAT): The constant value.: default = 0.0
```

Selecting Among Alternative Implementations

- Alternative implementations are represented as "targets"
- Targets can have parameters
- Floating point simulation, fixed point simulation, and C code generation are integrated today. FPGA generation being worked.



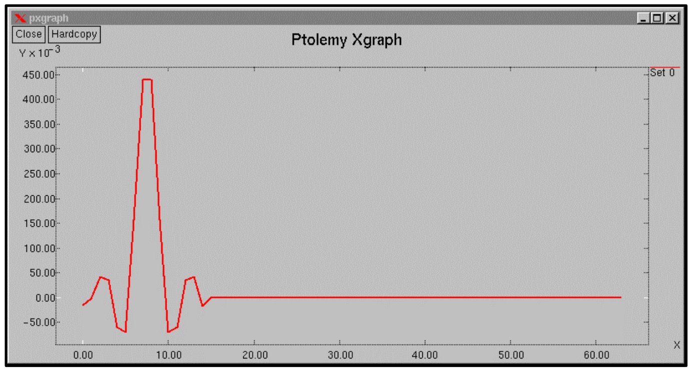
host:	
directory:	$HOME/PTOLEMY_SYSTEMS/ACS
file:	
Looping Level:	DEF #choices are: DEF, CLUST, SJS, ACYLOOP
display?:	YES
compile?:	YES
run?:	YES
write schedule?:	NO
logFile:	
schedulePeriod:	0.0
staticBuffering:	YES
funcName:	main
compileCommand:	gcc
compileOptions:	
linkOptions:	-lm
resources:	

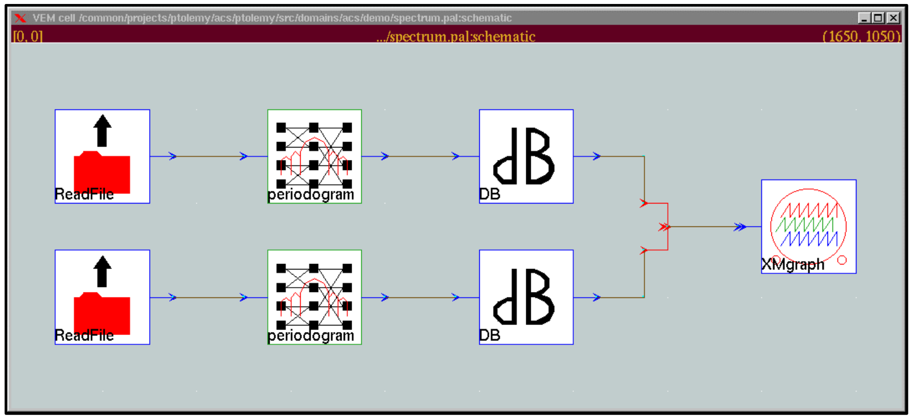


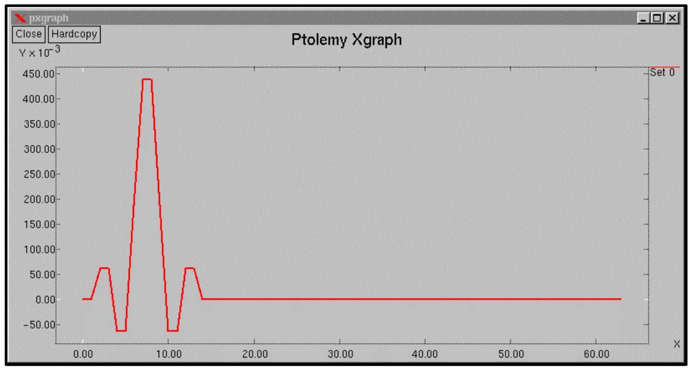
```
/* User:      epauer
   Date:      Wed May 20 10:21:45 1998
   Target:    ACS-CGC
   Universe:  butterfly */

/* Define macro for prototyping functions on ANSI & non-ANSI compilers */
#ifndef ARGS
#if defined(__STDC__) || defined(__cplusplus)
#define ARGS(args) args
#else
#define ARGS(args) ()
#endif
#endif

#include <math.h>
#include <stdio.h>

/* Define constants TRUE and FALSE for portability */
#ifndef TRUE
#define TRUE 1
#endif
#ifndef FALSE
#define FALSE 0
#endif

/* Define a complex data type if one has not been defined */
#if !defined(COMPLEX_DATA)
#define COMPLEX_DATA 1
typedef struct complex_data { double real; double imag; } complex;
#endif

#define ACS_MOD(a,b) ((a)>=(b) ? (a-b) : (a))

extern main ARGS((int argc, char *argv[]));

/* main function */
int main(int argc, char *argv[]) {
    FILE* fp_23;
double value_25;
```

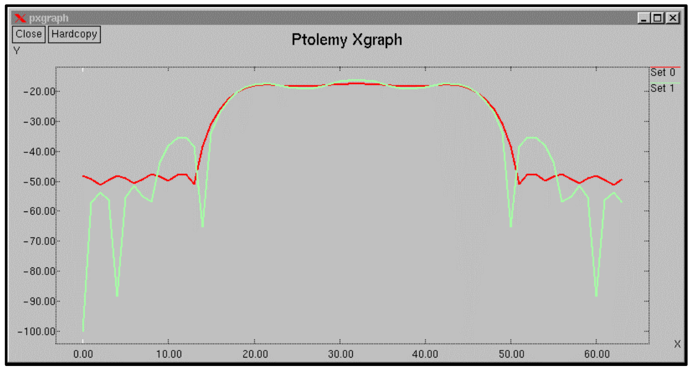
Comparing Implementations

- **Comparison of floating point and fixed point implementations**

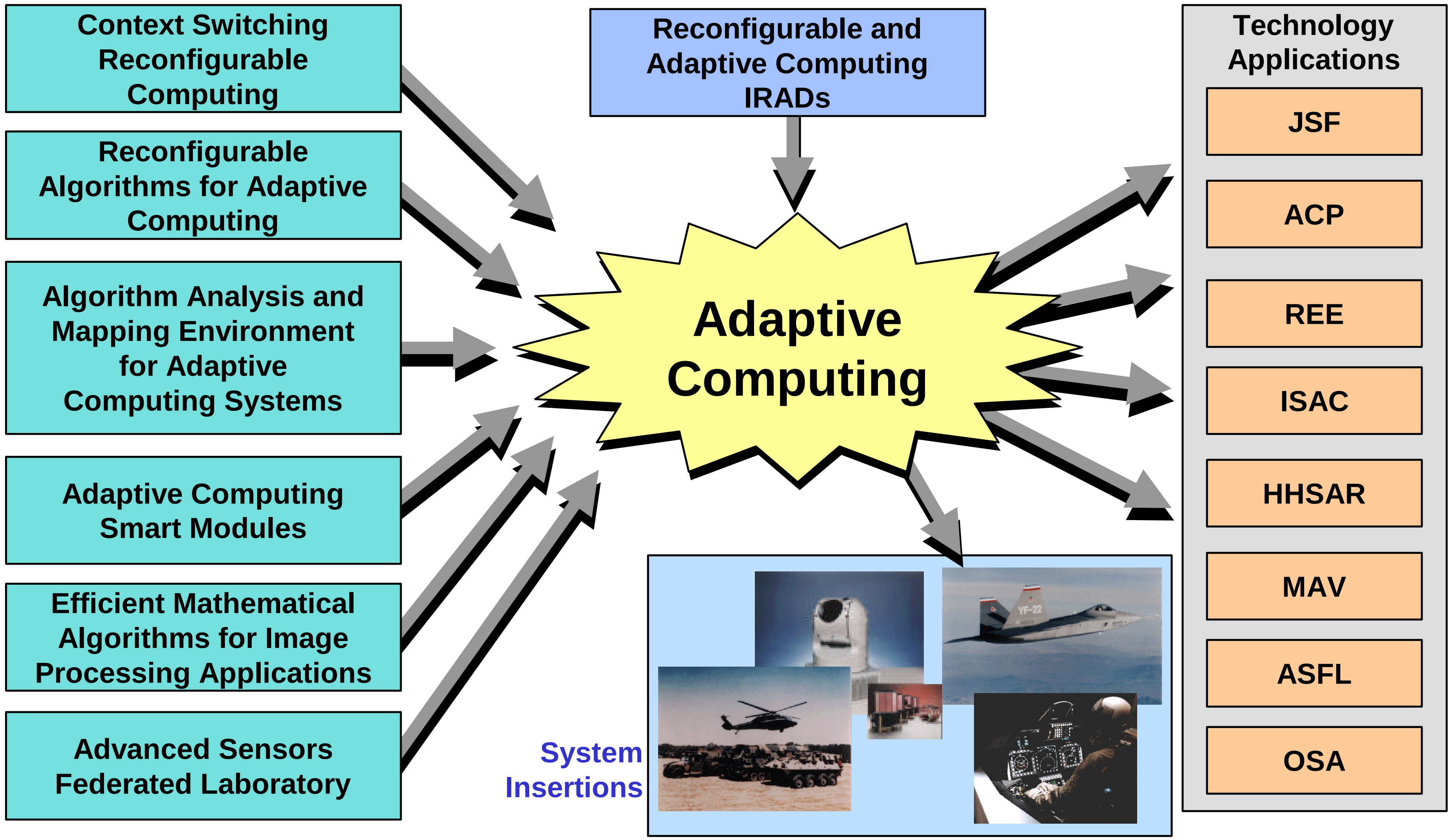



VEM cell /common/projects/ptolemy/acs/ptolemy/src/domains/acs/demo/spectrum.pal:schematic
[0, 0]
.../spectrum.pal:schematic
(1650, 1050)
ReadFile
periodogram
DB
ReadFile
periodogram
DB
XMgraph





Related Work at Sanders


Context Switching Reconfigurable Computing
Reconfigurable Algorithms for Adaptive Computing
Algorithm Analysis and Mapping Environment for Adaptive Computing Systems
Adaptive Computing Smart Modules
Efficient Mathematical Algorithms for Image Processing Applications
Advanced Sensors Federated Laboratory
Reconfigurable and Adaptive Computing IRADs
Adaptive Computing
System Insertions
Technology Applications
JSF
ACP
REE
ISAC
HHSAR
MAV
ASFL
OSA