طراحی سیستم‌های تعبیه‌شده

Embedded System Design

فصل پنجم ـ قسمت هشتم

پیاده‌سازی سیستم‌های تعبیه‌شده

Implementing Embedded Systems: Hardware / Software Codesign

کاظم فولادی
دانشکده‌ی مهندسی برق و کامپیوتر
دانشگاه تهران

kazim@fouladi.ir

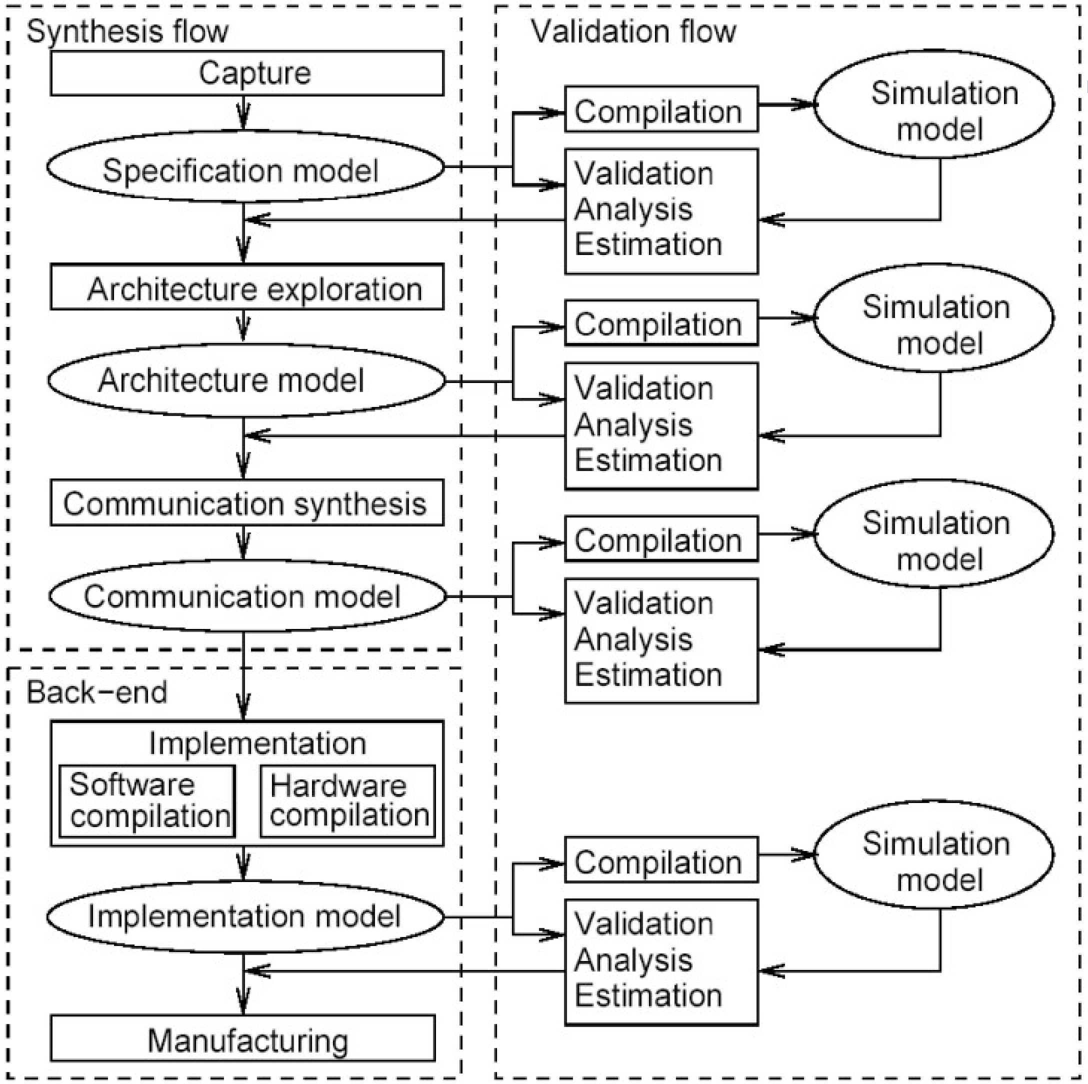
جریان‌های طراحی واقعی و ابزارها
Actual design flows and tools

Design steps can be used in different combinations.
In the following, we will present some examples …

1. SpecC [Gajski, Dömer, Gerstlauer]
   Focus on intellectual property (IP) components

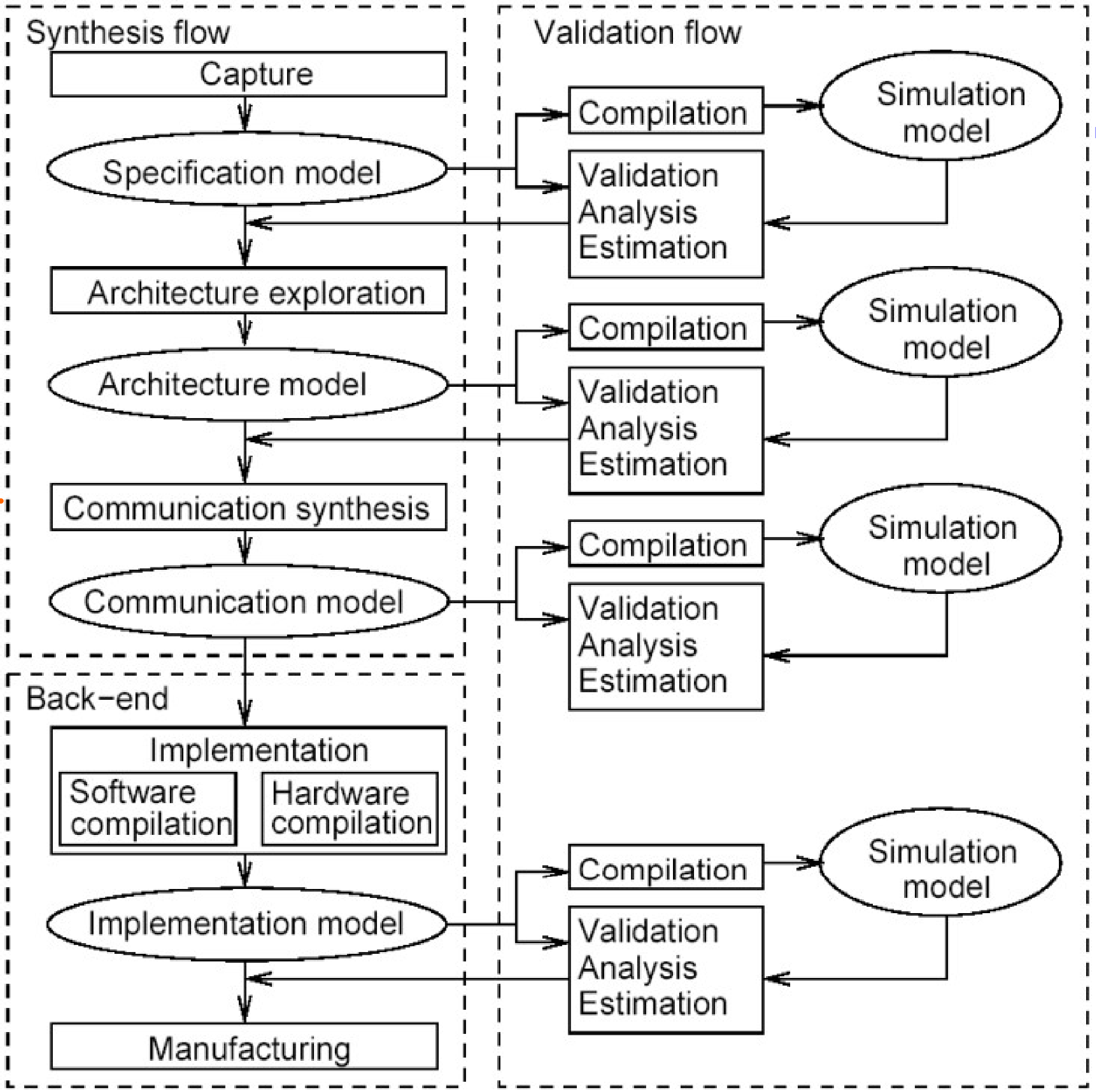
SpecC

Comprises allocation (selecting components from a library), partitioning (mapping of parts of the system specification onto the components) & scheduling (serialization of execution).

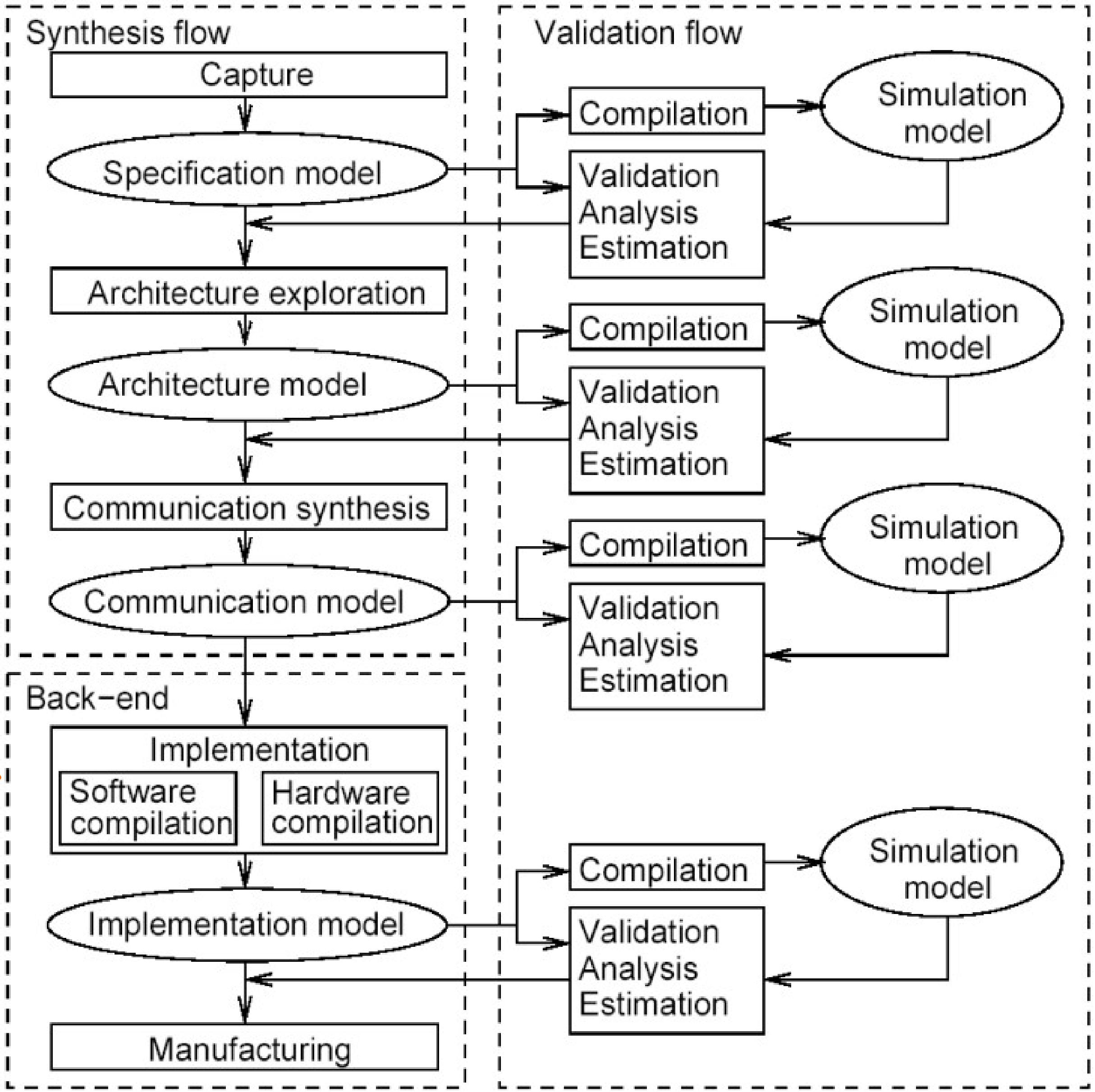


SpecC

abstract busses replaced by actual wires in a series of refinements

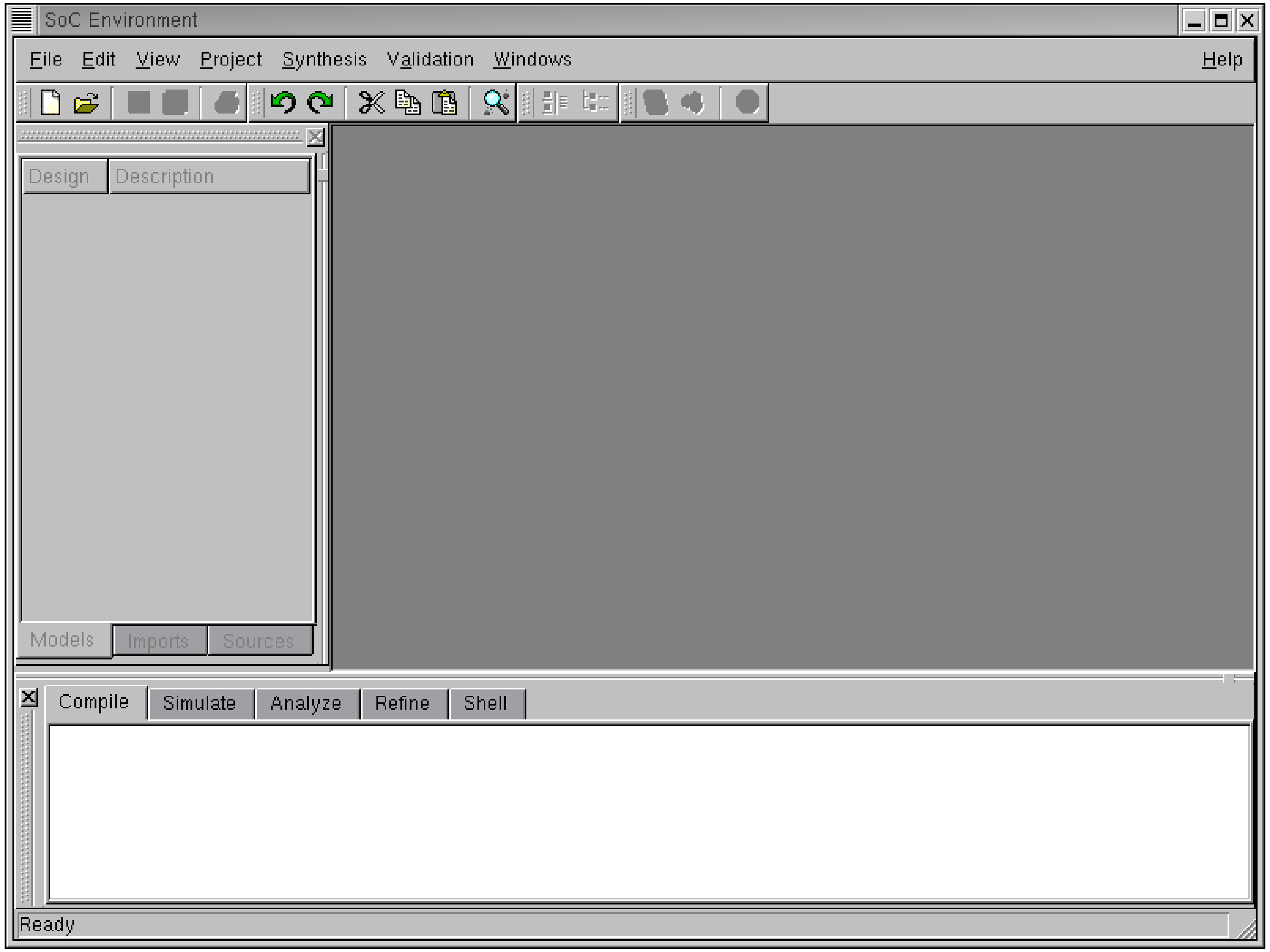


SpecC

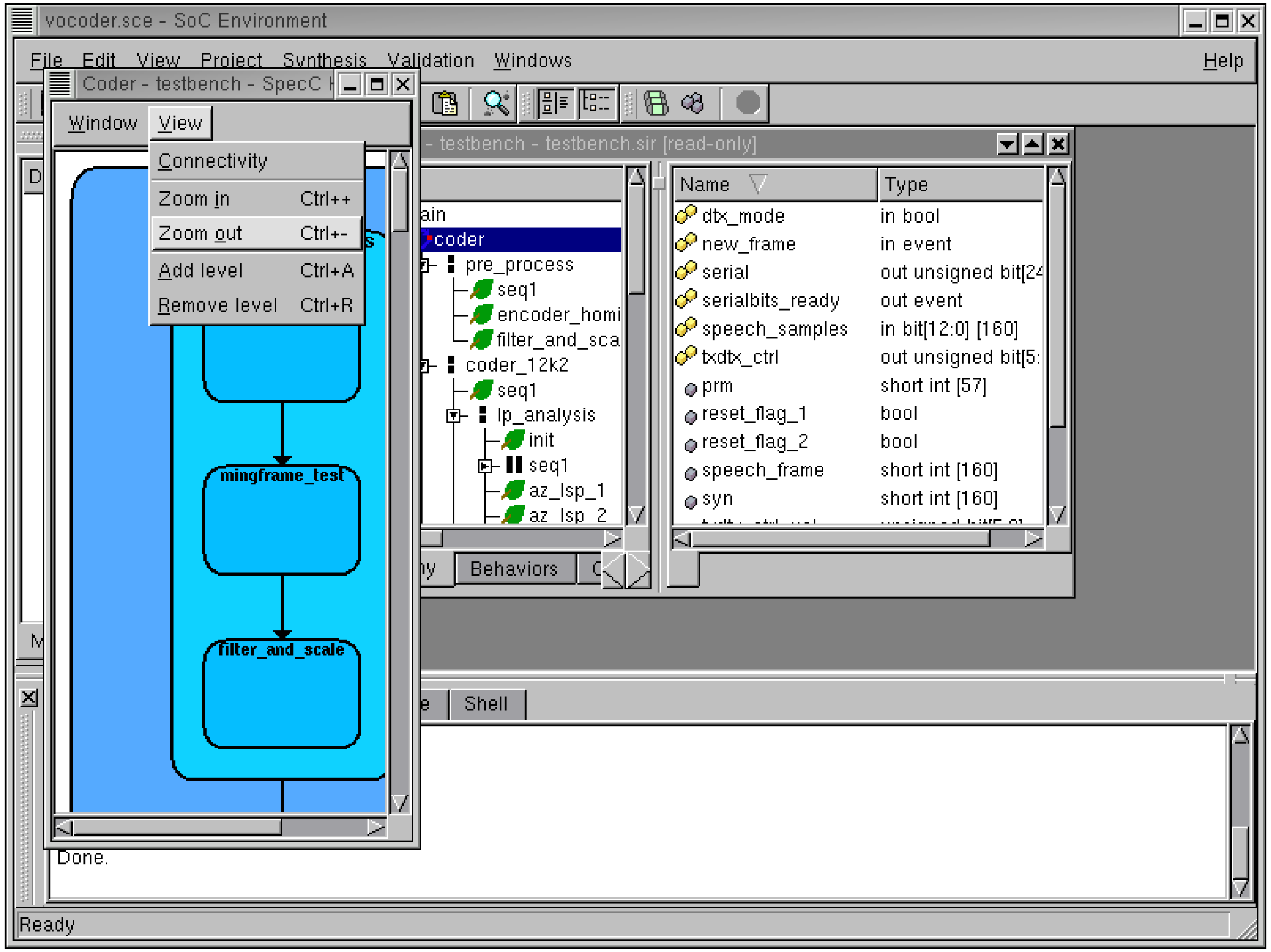


Compilers are used to generate binary machine code and hardware synthesis tools are used to generate custom hardware.

An actual example
- Getting started with the SCE window -


SoC Environment
File Edit View Project Synthesis Validation Windows Help
Design Description
Models Imports Sources
Compile Simulate Analyze Refine Shell
Ready

Browsing the specification

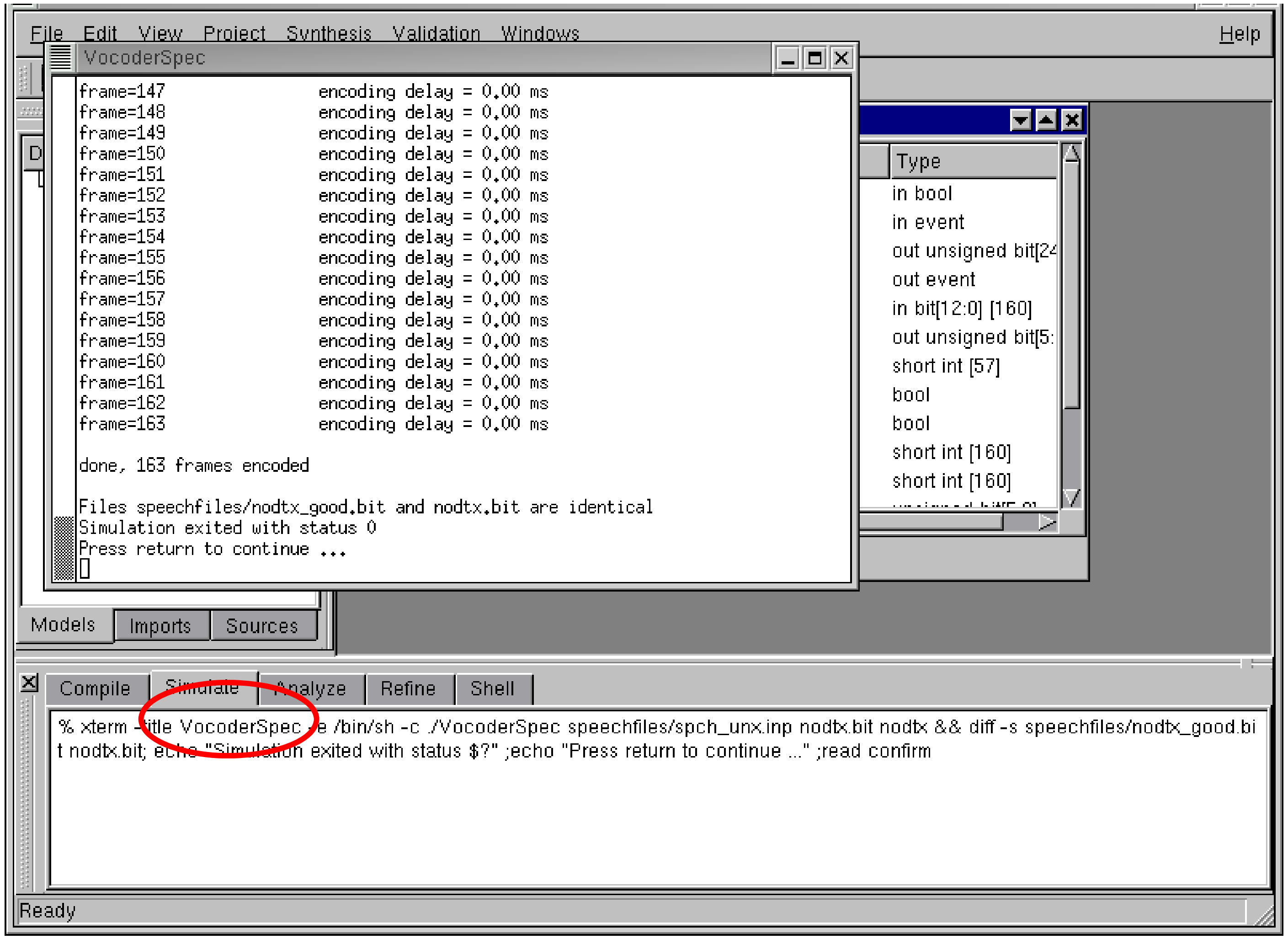

vocoder.sce - SoC Environment
File Edit View Project Synthesis Validation Windows Help
Coder - testbench - SpecC
Window View
Connectivity
Zoom in Ctrl++
Zoom out Ctrl+-
Add level Ctrl+A
Remove level Ctrl+R
mingframe_test
filter_and_scale
- testbench - testbench.sir [read-only]
Behaviors
Shell
Ready

```
ain
coder
  pre_process
    seq1
    encoder_homi
    filter_and_sca
  coder_12k2
    seq1
    lp_analysis
      init
      seq1
      az_lsp_1
      az_lsp_2
```

Name	Type
dtx_mode	in bool
new_frame	in event
serial	out unsigned bit[24
serialbits_ready	out event
speech_samples	in bit[12:0] [160]
txdtx_ctrl	out unsigned bit[5:
prm	short int [57]
reset_flag_1	bool
reset_flag_2	bool
speech_frame	short int [160]
syn	short int [160]

Done.

An actual example
- Validation by simulation -

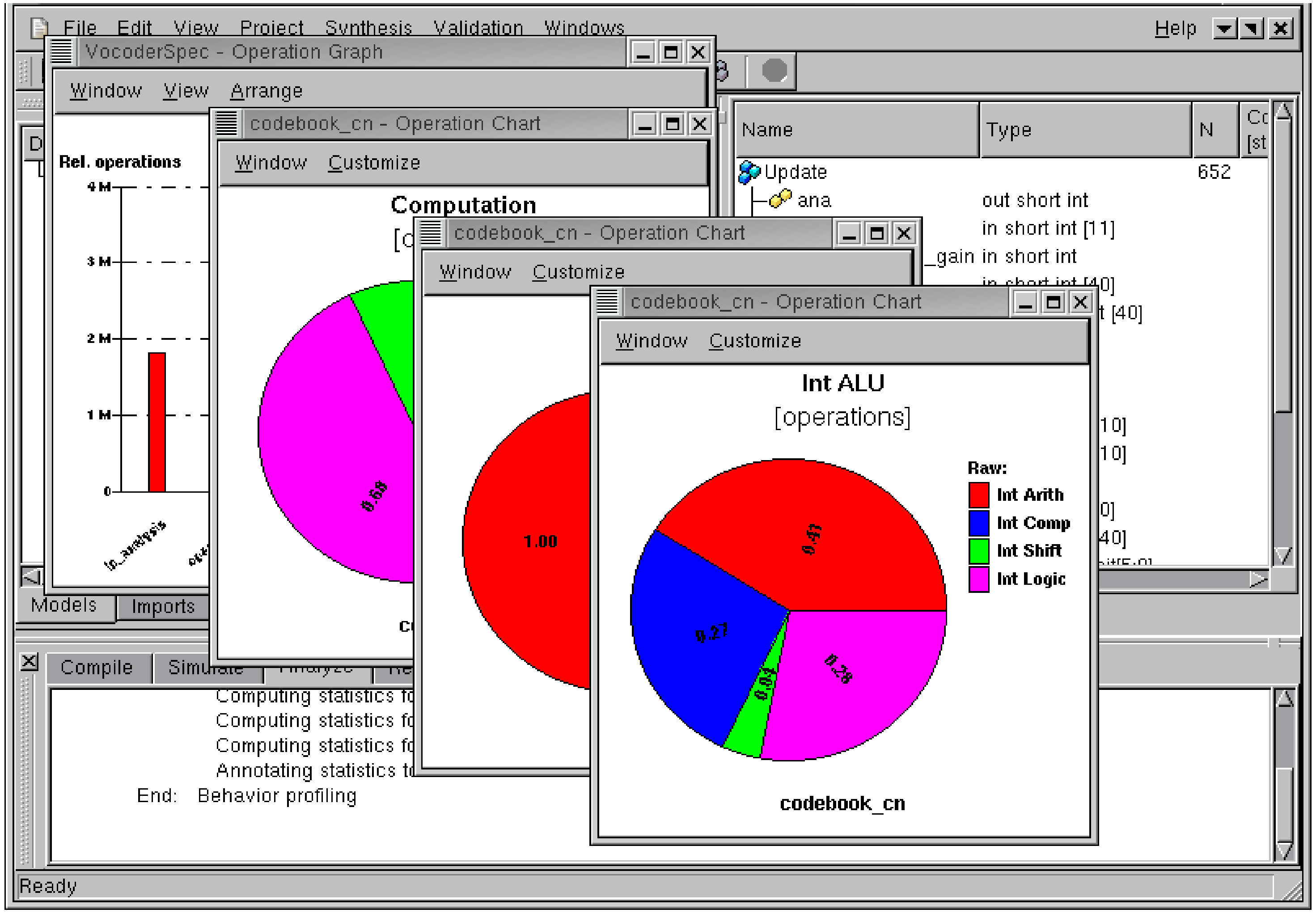

File Edit View Project Synthesis Validation Windows Help
VocoderSpec
Models Imports Sources
Compile Simulate Analyze Refine Shell
Ready

```
frame=147               encoding delay = 0.00 ms
frame=148               encoding delay = 0.00 ms
frame=149               encoding delay = 0.00 ms
frame=150               encoding delay = 0.00 ms
frame=151               encoding delay = 0.00 ms
frame=152               encoding delay = 0.00 ms
frame=153               encoding delay = 0.00 ms
frame=154               encoding delay = 0.00 ms
frame=155               encoding delay = 0.00 ms
frame=156               encoding delay = 0.00 ms
frame=157               encoding delay = 0.00 ms
frame=158               encoding delay = 0.00 ms
frame=159               encoding delay = 0.00 ms
frame=160               encoding delay = 0.00 ms
frame=161               encoding delay = 0.00 ms
frame=162               encoding delay = 0.00 ms
frame=163               encoding delay = 0.00 ms

done, 163 frames encoded

Files speechfiles/nodtx_good.bit and nodtx.bit are identical
Simulation exited with status 0
Press return to continue ...
```

Type
in bool
in event
out unsigned bit[24
out event
in bit[12:0] [160]
out unsigned bit[5:
short int [57]
bool
bool
short int [160]
short int [160]

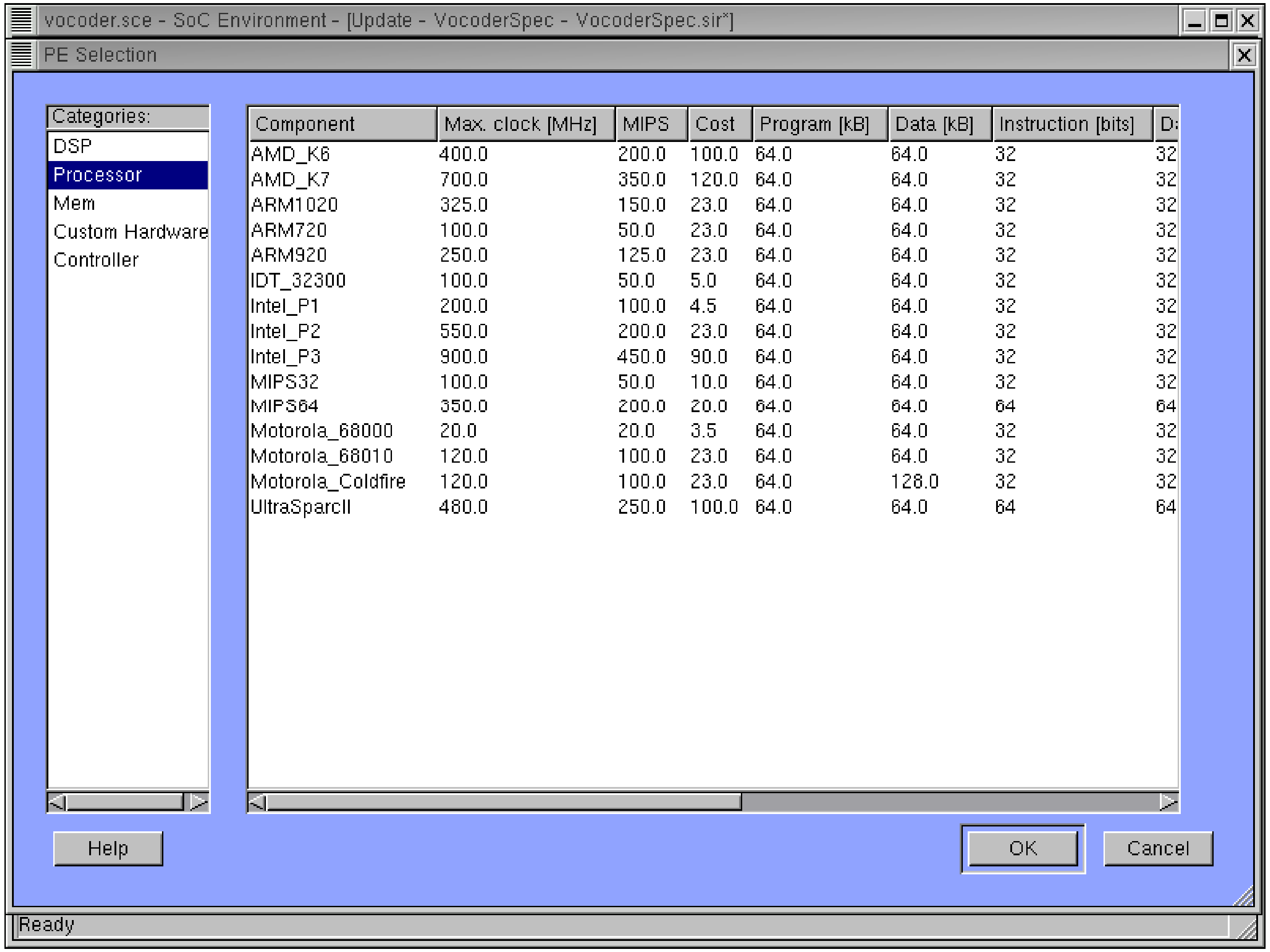
```
% xterm -title VocoderSpec -e /bin/sh -c ./VocoderSpec speechfiles/spch_unx.inp nodtx.bit nodtx && diff -s speechfiles/nodtx_good.bi
t nodtx.bit; echo "Simulation exited with status $?" ;echo "Press return to continue ..." ;read confirm
```

Analyze profiling results



```
Computing statistics fo
Computing statistics fo
Computing statistics fo
Annotating statistics to
End:   Behavior profiling
```

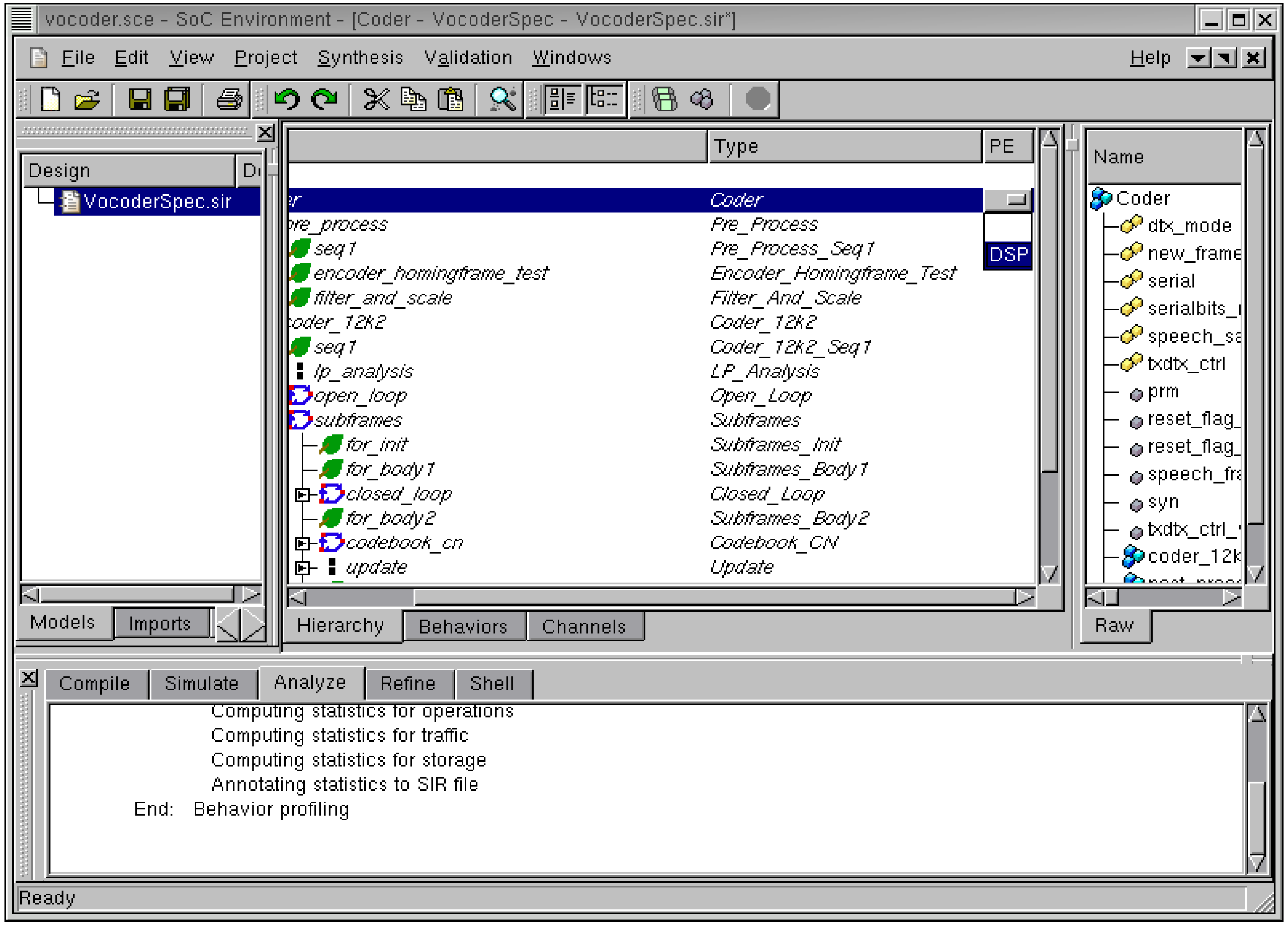
PE selection


vocoder.sce - SoC Environment - [Update - VocoderSpec - VocoderSpec.sir*]
PE Selection
Help
OK
Cancel
Ready

Categories:
- DSP
- Processor
- Mem
- Custom Hardware
- Controller

Component	Max. clock [MHz]	MIPS	Cost	Program [kB]	Data [kB]	Instruction [bits]	D
AMD_K6	400.0	200.0	100.0	64.0	64.0	32	32
AMD_K7	700.0	350.0	120.0	64.0	64.0	32	32
ARM1020	325.0	150.0	23.0	64.0	64.0	32	32
ARM720	100.0	50.0	23.0	64.0	64.0	32	32
ARM920	250.0	125.0	23.0	64.0	64.0	32	32
IDT_32300	100.0	50.0	5.0	64.0	64.0	32	32
Intel_P1	200.0	100.0	4.5	64.0	64.0	32	32
Intel_P2	550.0	200.0	23.0	64.0	64.0	32	32
Intel_P3	900.0	450.0	90.0	64.0	64.0	32	32
MIPS32	100.0	50.0	10.0	64.0	64.0	32	32
MIPS64	350.0	200.0	20.0	64.0	64.0	64	64
Motorola_68000	20.0	20.0	3.5	64.0	64.0	32	32
Motorola_68010	120.0	100.0	23.0	64.0	64.0	32	32
Motorola_Coldfire	120.0	100.0	23.0	64.0	128.0	32	32
UltraSparcII	480.0	250.0	100.0	64.0	64.0	64	64

The top level behavior is mapped to the DSP


vocoder.sce - SoC Environment - [Coder - VocoderSpec - VocoderSpec.sir*]

File Edit View Project Synthesis Validation Windows Help

Design: VocoderSpec.sir

	Type	PE
er	Coder	
pre_process	Pre_Process	
seq1	Pre_Process_Seq1	DSP
encoder_homingframe_test	Encoder_Homingframe_Test	
filter_and_scale	Filter_And_Scale	
coder_12k2	Coder_12k2	
seq1	Coder_12k2_Seq1	
lp_analysis	LP_Analysis	
open_loop	Open_Loop	
subframes	Subframes	
for_init	Subframes_Init	
for_body1	Subframes_Body1	
closed_loop	Closed_Loop	
for_body2	Subframes_Body2	
codebook_cn	Codebook_CN	
update	Update	

Name: Coder, dtx_mode, new_frame, serial, serialbits_, speech_sa, txdtx_ctrl, prm, reset_flag_, reset_flag_, speech_fra, syn, txdtx_ctrl_, coder_12k

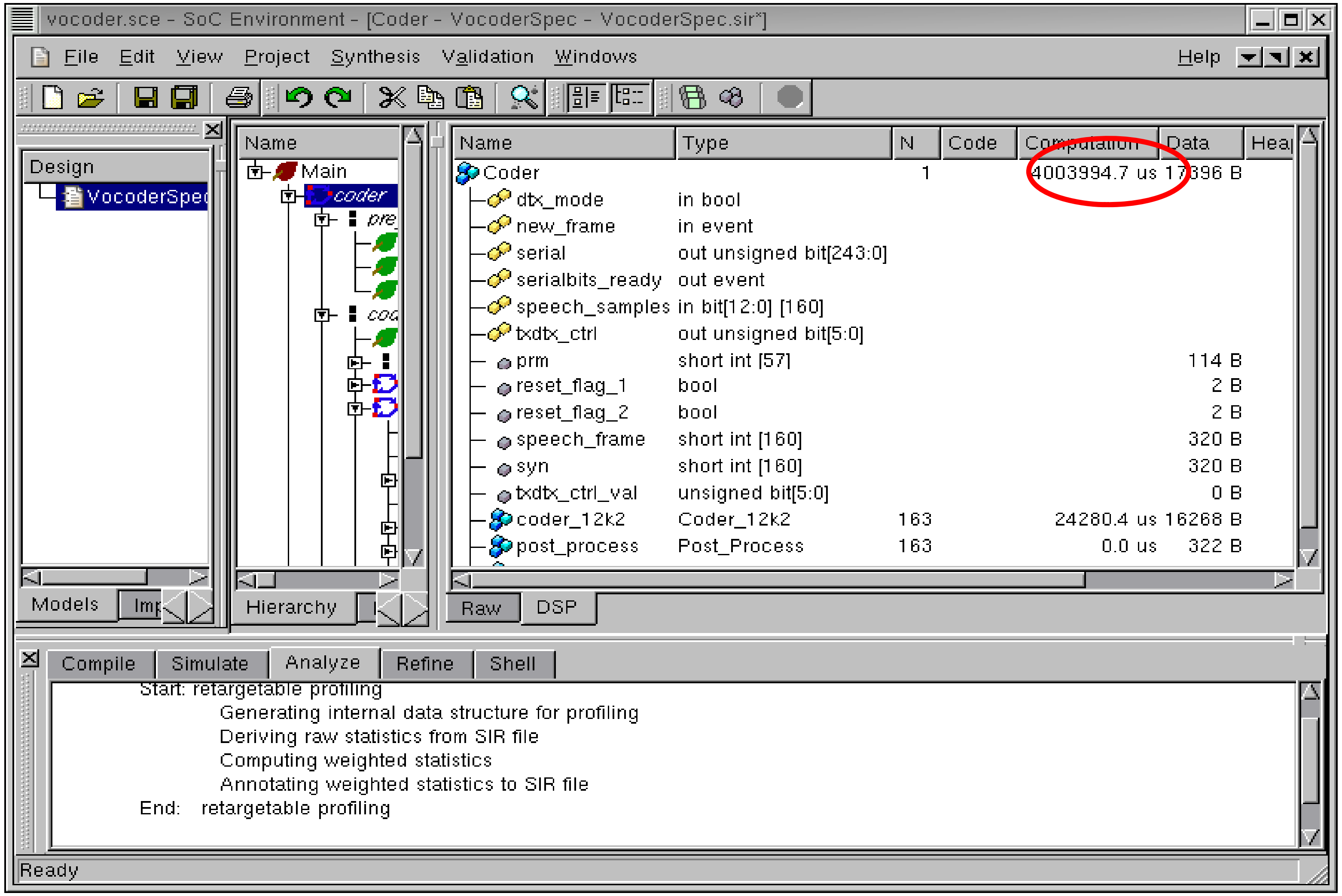
Models Imports | Hierarchy Behaviors Channels | Raw

Compile Simulate Analyze Refine Shell

```
                Computing statistics for operations
                Computing statistics for traffic
                Computing statistics for storage
                Annotating statistics to SIR file
        End:   Behavior profiling
```

Ready

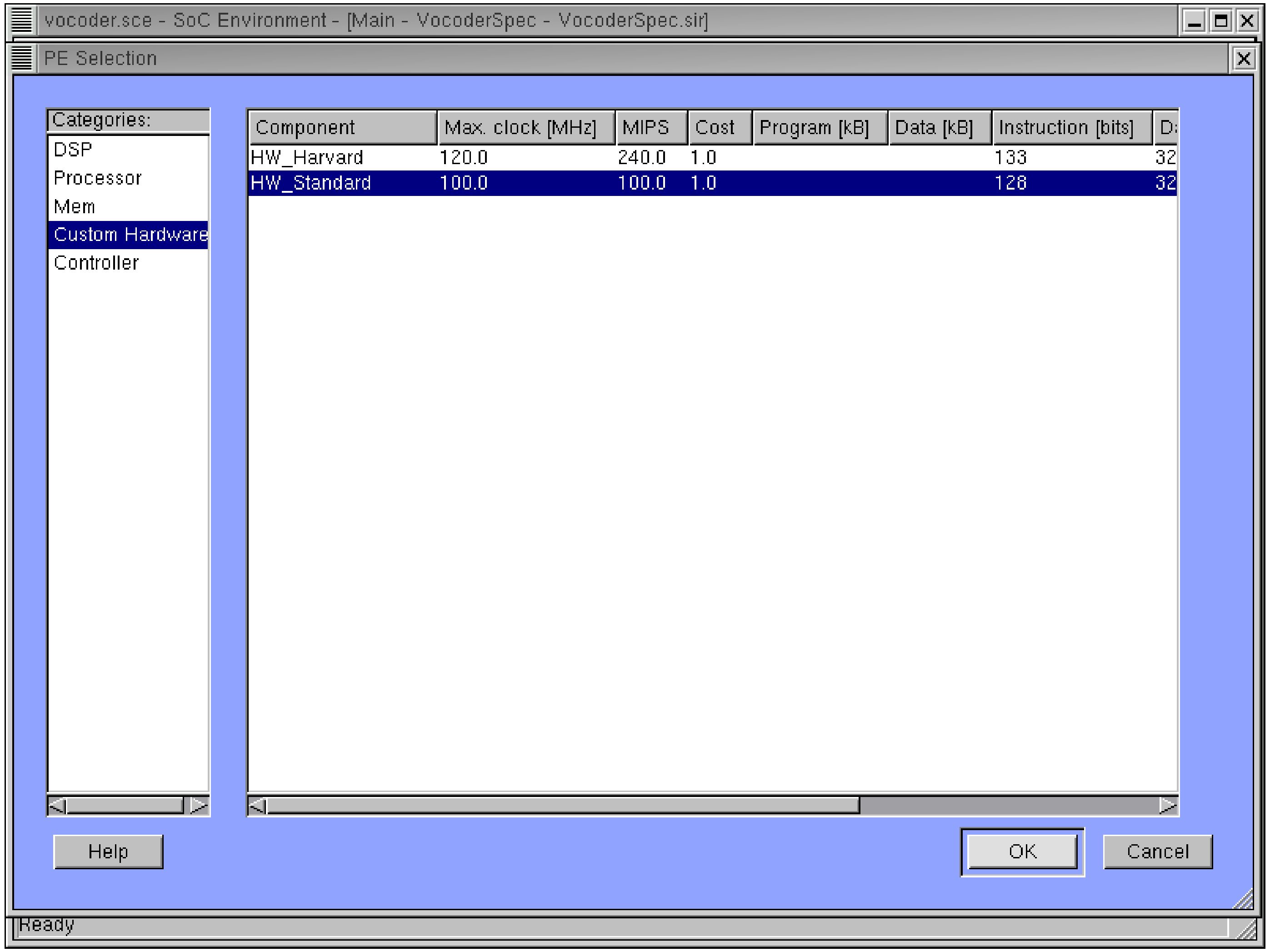
Estimate the performance (too slow)


vocoder.sce - SoC Environment - [Coder - VocoderSpec - VocoderSpec.sir*]
File Edit View Project Synthesis Validation Windows Help
Design
VocoderSpec
Models
Imp
Name
Main
coder
pre
coc
Hierarchy
Raw
DSP
Compile Simulate Analyze Refine Shell
Ready

Name	Type	N	Code	Computation	Data	Heap
Coder		1		4003994.7 us	17396 B	
dtx_mode	in bool					
new_frame	in event					
serial	out unsigned bit[243:0]					
serialbits_ready	out event					
speech_samples	in bit[12:0] [160]					
txdtx_ctrl	out unsigned bit[5:0]					
prm	short int [57]				114 B	
reset_flag_1	bool				2 B	
reset_flag_2	bool				2 B	
speech_frame	short int [160]				320 B	
syn	short int [160]				320 B	
txdtx_ctrl_val	unsigned bit[5:0]				0 B	
coder_12k2	Coder_12k2	163		24280.4 us	16268 B	
post_process	Post_Process	163		0.0 us	322 B	

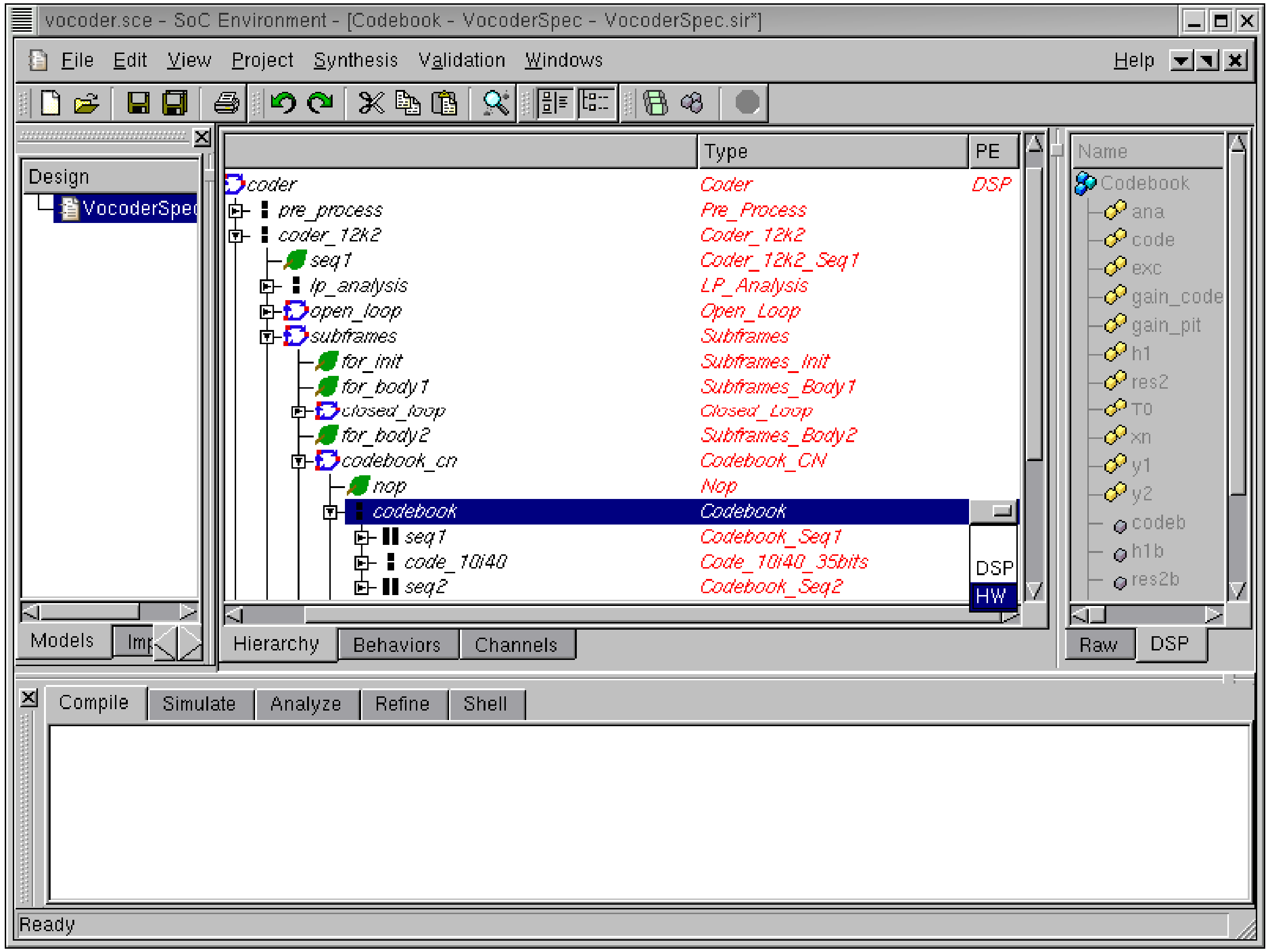
```
Start: retargetable profiling
          Generating internal data structure for profiling
          Deriving raw statistics from SIR file
          Computing weighted statistics
          Annotating weighted statistics to SIR file
End:   retargetable profiling
```

Selecting additional custom HW including datapath and controller



Component	Max. clock [MHz]	MIPS	Cost	Program [kB]	Data [kB]	Instruction [bits]	D:
HW_Harvard	120.0	240.0	1.0			133	32
HW_Standard	100.0	100.0	1.0			128	32

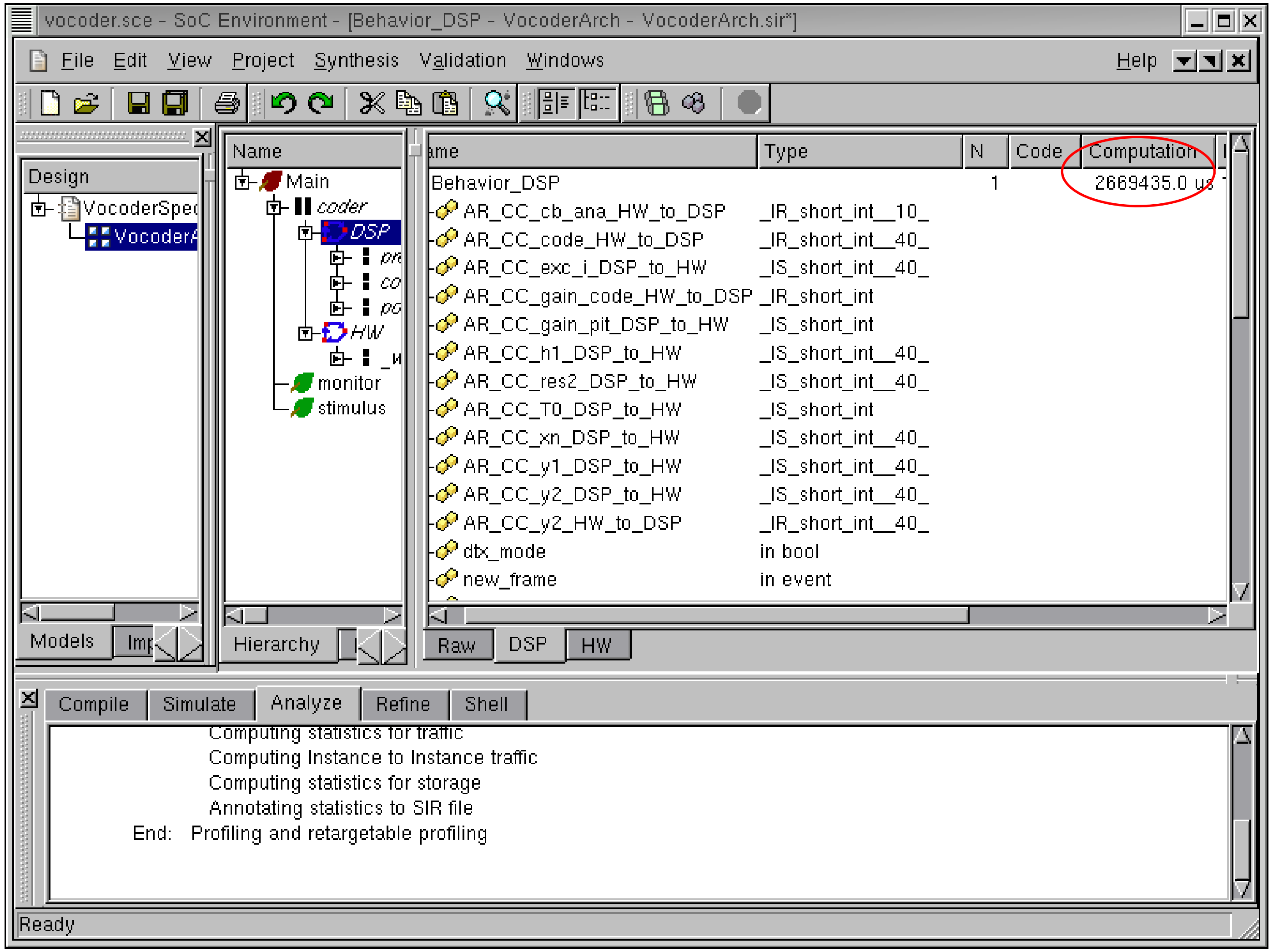
Binding “codebook” to the custom datapath


vocoder.sce - SoC Environment - [Codebook - VocoderSpec - VocoderSpec.sir*]
File Edit View Project Synthesis Validation Windows Help
Design
VocoderSpec
Models Imp
Hierarchy Behaviors Channels
Raw DSP
Compile Simulate Analyze Refine Shell
Ready

	Type	PE
coder	Coder	DSP
pre_process	Pre_Process	
coder_12k2	Coder_12k2	
seq1	Coder_12k2_Seq1	
lp_analysis	LP_Analysis	
open_loop	Open_Loop	
subframes	Subframes	
for_init	Subframes_Init	
for_body1	Subframes_Body1	
closed_loop	Closed_Loop	
for_body2	Subframes_Body2	
codebook_cn	Codebook_CN	
nop	Nop	
codebook	Codebook	
seq1	Codebook_Seq1	
code_10i40	Code_10i40_35bits	DSP
seq2	Codebook_Seq2	HW

Name
Codebook
ana
code
exc
gain_code
gain_pit
h1
res2
T0
xn
y1
y2
codeb
h1b
res2b

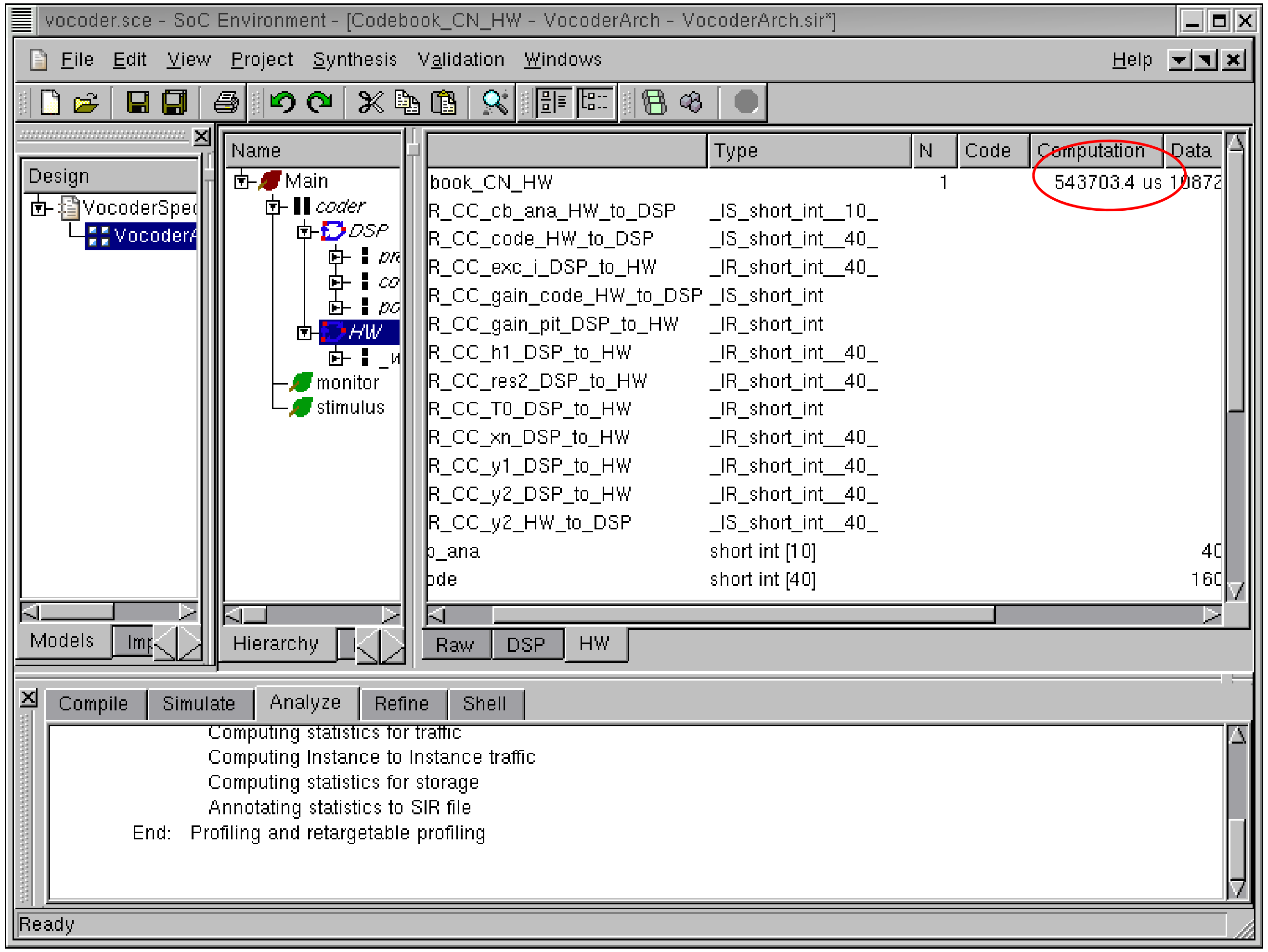
Execution time of the DSP


vocoder.sce - SoC Environment - [Behavior_DSP - VocoderArch - VocoderArch.sir*]
File Edit View Project Synthesis Validation Windows Help
Design
VocoderSpec
VocoderA
Name
Main
coder
DSP
HW
monitor
stimulus
Models
Hierarchy
Raw DSP HW
Compile Simulate Analyze Refine Shell
Ready

ame	Type	N	Code	Computation
Behavior_DSP		1		2669435.0 us
AR_CC_cb_ana_HW_to_DSP	_IR_short_int__10_			
AR_CC_code_HW_to_DSP	_IR_short_int__40_			
AR_CC_exc_i_DSP_to_HW	_IS_short_int__40_			
AR_CC_gain_code_HW_to_DSP	_IR_short_int			
AR_CC_gain_pit_DSP_to_HW	_IS_short_int			
AR_CC_h1_DSP_to_HW	_IS_short_int__40_			
AR_CC_res2_DSP_to_HW	_IS_short_int__40_			
AR_CC_T0_DSP_to_HW	_IS_short_int			
AR_CC_xn_DSP_to_HW	_IS_short_int__40_			
AR_CC_y1_DSP_to_HW	_IS_short_int__40_			
AR_CC_y2_DSP_to_HW	_IS_short_int__40_			
AR_CC_y2_HW_to_DSP	_IR_short_int__40_			
dtx_mode	in bool			
new_frame	in event			

```
            Computing statistics for traffic
            Computing Instance to Instance traffic
            Computing statistics for storage
            Annotating statistics to SIR file
    End:  Profiling and retargetable profiling
```

Execution time for hardware


vocoder.sce - SoC Environment - [Codebook_CN_HW - VocoderArch - VocoderArch.sir*]
File Edit View Project Synthesis Validation Windows Help
Design
VocoderSpec
VocoderA
Models Imp
Name
Main
coder
DSP
HW
monitor
stimulus
Hierarchy
Raw DSP HW
Compile Simulate Analyze Refine Shell
Ready

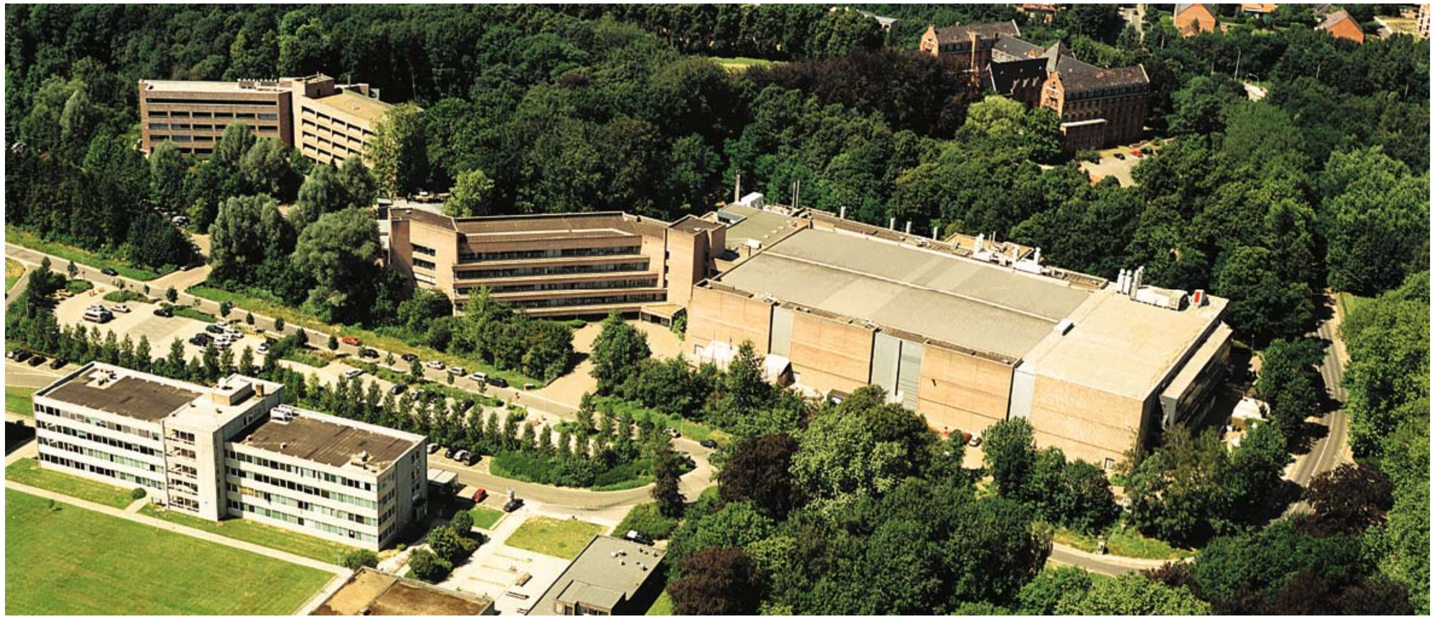
	Type	N	Code	Computation	Data
book_CN_HW		1		543703.4 us	10872
R_CC_cb_ana_HW_to_DSP	_IS_short_int__10_				
R_CC_code_HW_to_DSP	_IS_short_int__40_				
R_CC_exc_i_DSP_to_HW	_IR_short_int__40_				
R_CC_gain_code_HW_to_DSP	_IS_short_int				
R_CC_gain_pit_DSP_to_HW	_IR_short_int				
R_CC_h1_DSP_to_HW	_IR_short_int__40_				
R_CC_res2_DSP_to_HW	_IR_short_int__40_				
R_CC_T0_DSP_to_HW	_IR_short_int				
R_CC_xn_DSP_to_HW	_IR_short_int__40_				
R_CC_y1_DSP_to_HW	_IR_short_int__40_				
R_CC_y2_DSP_to_HW	_IR_short_int__40_				
R_CC_y2_HW_to_DSP	_IS_short_int__40_				
o_ana	short int [10]				40
ode	short int [40]				160

```
                Computing statistics for traffic
                Computing Instance to Instance traffic
                Computing statistics for storage
                Annotating statistics to SIR file
        End:  Profiling and retargetable profiling
```

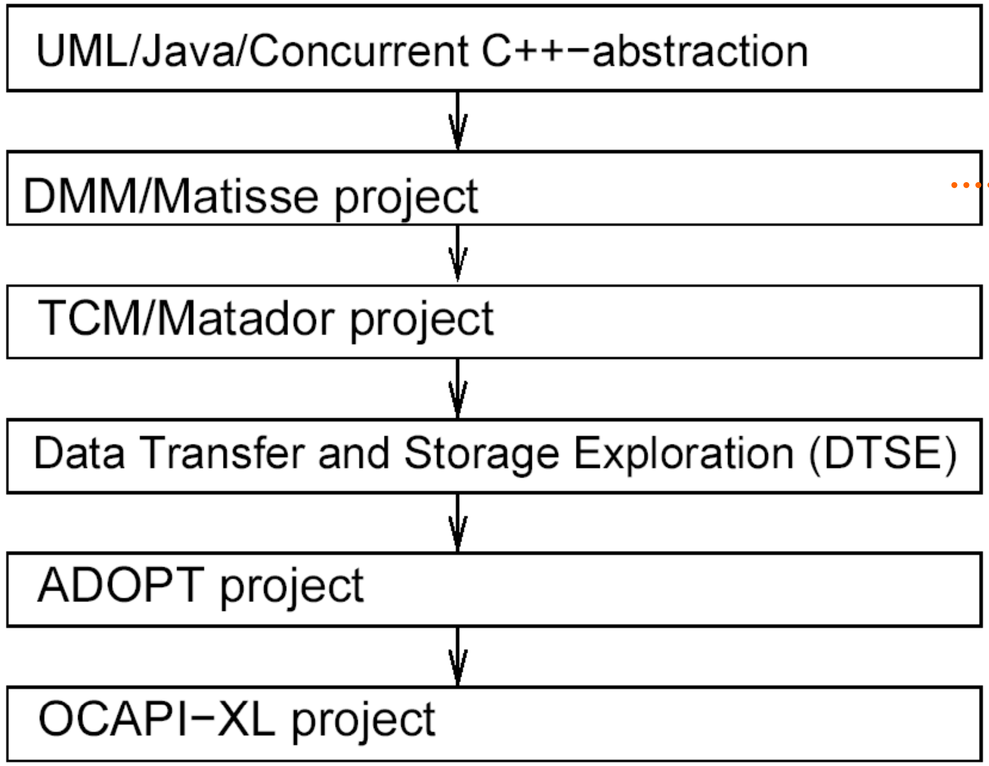
Assuming that 0.54+2.66 sec is acceptable

2. IMEC tool flow

IMEC = Interuniversitair Micro-Electronica Centrum,
Leuven, Belgium
(Large research facility)

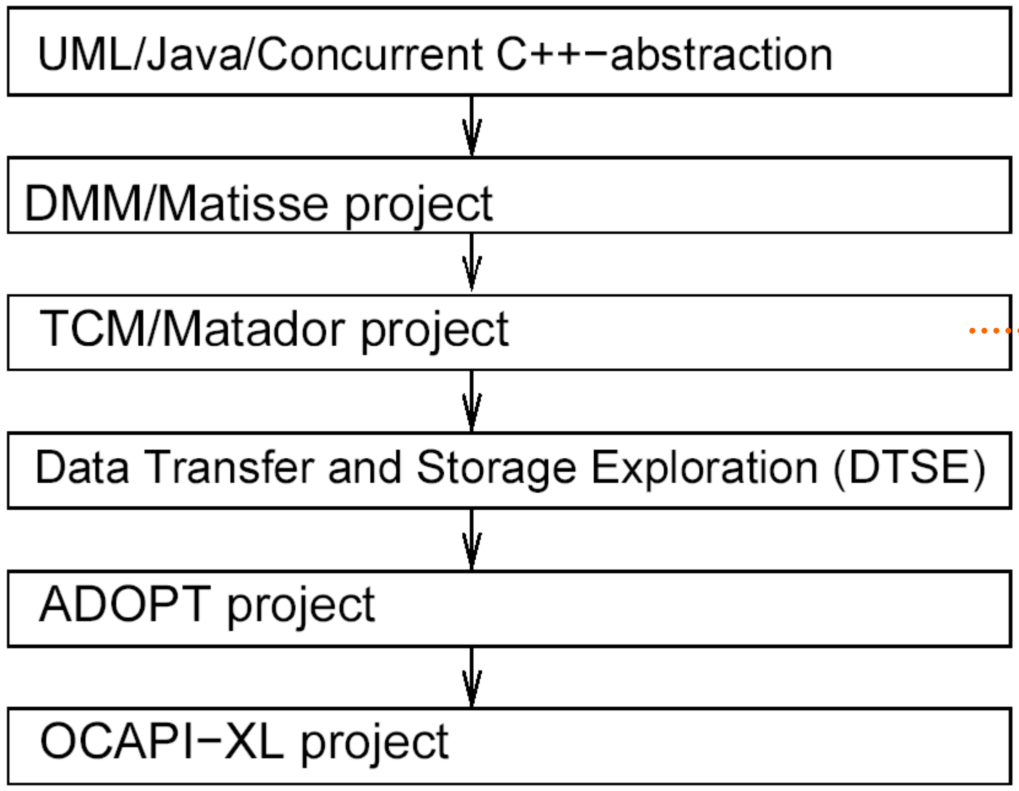


Imec tool flow
- Global view -


UML/Java/Concurrent C++−abstraction
DMM/Matisse project
TCM/Matador project
Data Transfer and Storage Exploration (DTSE)
ADOPT project
OCAPI−XL project

Considers the system at the concurrent process level as a set of concurrent and dynamic processes, whose specification consists of algorithms, abstract data types, communication primitives, and real-time requirements. Tools can perform source code transformations on the dynamic data types & provide also a memory pool organization in the virtual memory space.

Imec tool flow - Matador/TCM -


UML/Java/Concurrent C++−abstraction
DMM/Matisse project
TCM/Matador project
Data Transfer and Storage Exploration (DTSE)
ADOPT project
OCAPI−XL project

Again considering a system of concurrent processes. For these tools, the emphasis is on mapping tasks to processors. Different configurations of multi-processor systems are evaluated and curves of designs that are non-inferior to others are generated. These curves provide a view of the design space, and are the basis for final design decisions.

Matador/Task Concurrency Management (TCM)

Wong et al. [Wong et al., 2001]: configurations for a personal MPEG-4 player: combination of StrongArm processors and custom accelerators.
Found 4 configurations satisfying timing constraint of 30 ms.

Processor combination	1	2	3	4
Number of high speed processors	6	5	4	3
Number of low speed processors	0	3	5	7
Total number of processors	6	8	9	10

For combinations 1 and 4, only one allocation of tasks to processors meets the timing constraints. For combinations 2 and 3, different time budgets lead to different task to processor mappings and different energy consumptions.

Pareto points

Multiobjective optimization:

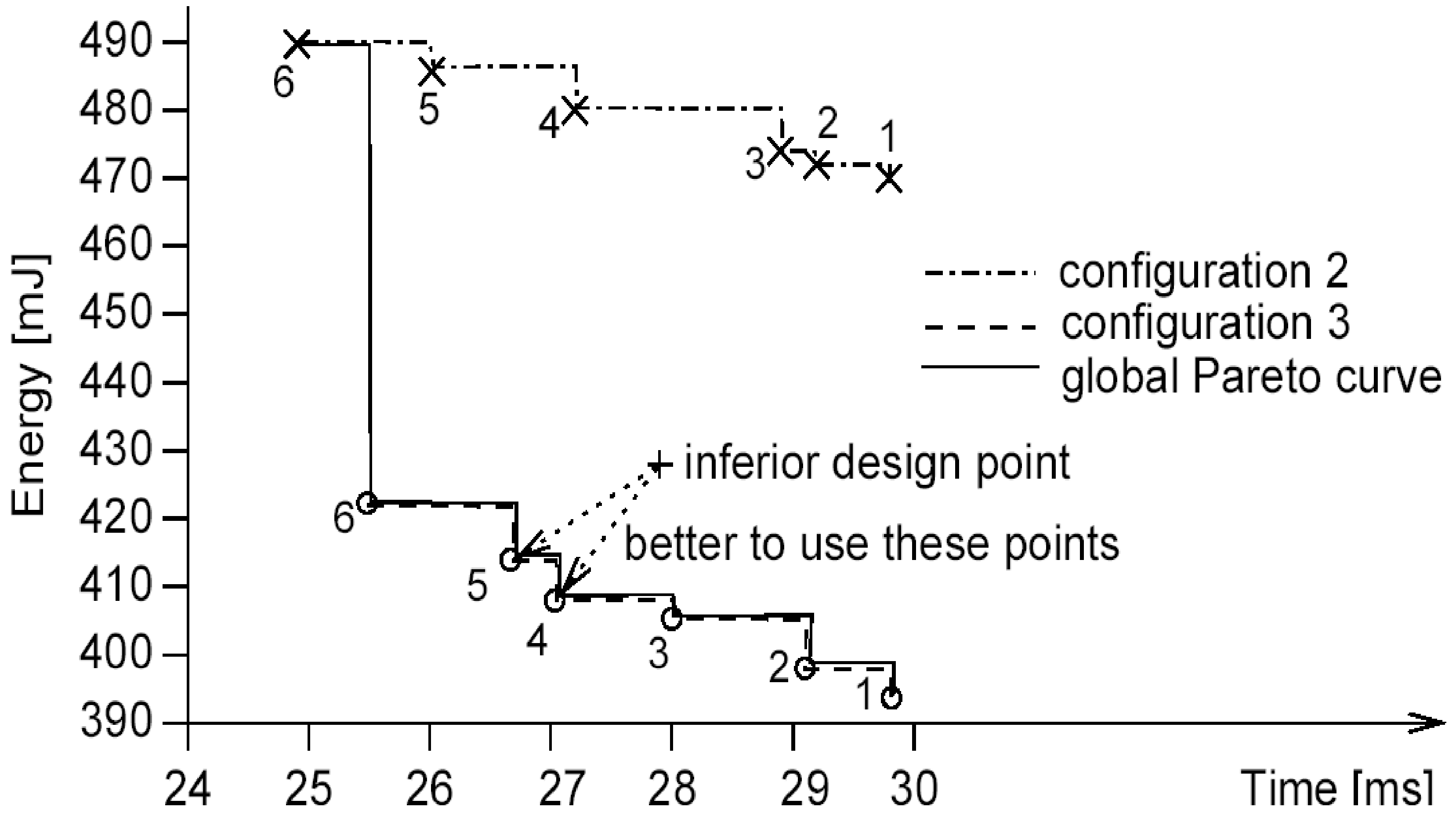
– several conflicting criteria

  – eg. performance vs. cost vs. power consumption

Definition: A (design) point J_i is **dominated** by point J_k, if J_k is equal or better than J_i in each criterion ($J_i \leq J_k$).

Definition: A (design) point is **Pareto-optimal** or a **Pareto point**, if it is not dominated by any other point.

Pareto curves



Data Transfer and Storage Exploration (DTSE)

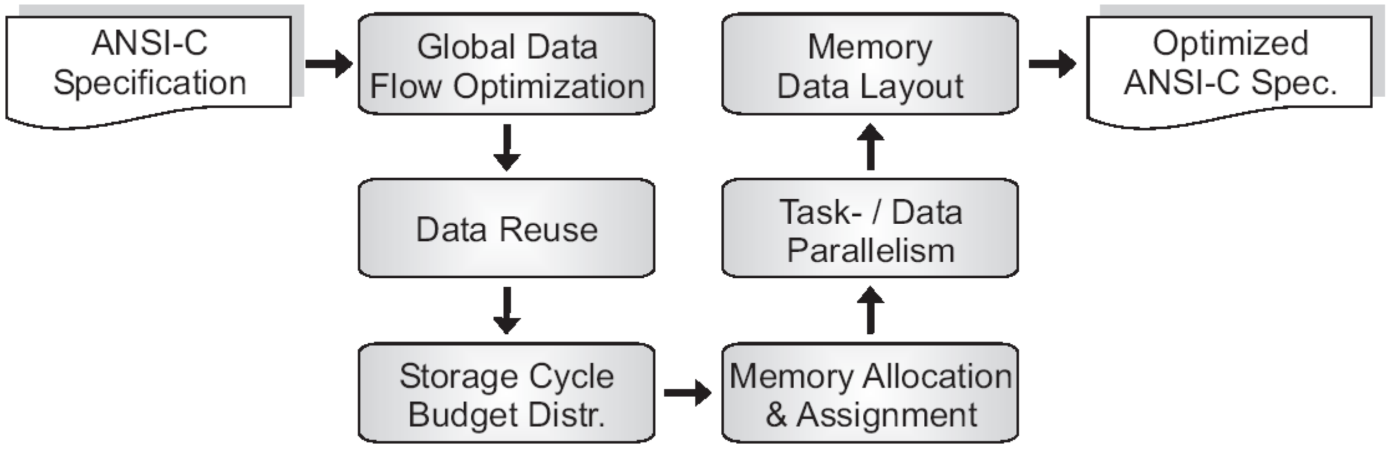
Reduction of the data transfers between processing components and at a reduction of the storage requirements.

DTSE has proven to be highly effective in minimizing the energy cost related to data memory hierarchies.

For several typical embedded applications, reductions of main memory accesses, cache accesses and energy consumption between 50% and 80% have been reported.

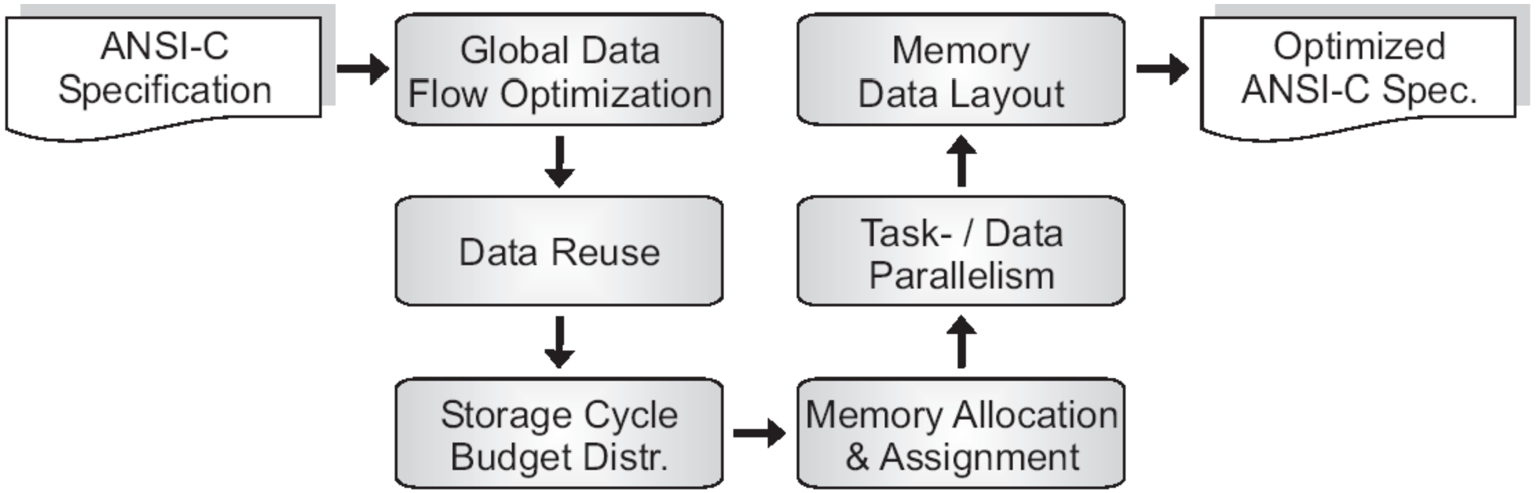
© Falk, 2004

Data Transfer and Exploration (DTSE)



- A memory oriented data flow analysis is performed,
- followed by global data flow and loop transformations to reduce the amount of background memories;
- data reuse transformations exploit a distributed memories;
- storage cycle budget distribution determines the bandwidth requirements and the balancing of the available cycle budget over the different memory accesses.

Data Transfer and Exploration (DTSE)



- The memory hierarchy layer assignment produces a netlist of memory units & an assignment of variables to memories.
- For multi-processor systems, task- / data-parallelism exploitation minimizes communication & storage overhead caused by the parallel execution of subsystems;
- Data layout transformations map variables with non-overlapping lifetimes in the same physical memory location.

Address optimization (ADOPT)

Addressing is simplified in address optimization (ADOPT) tools.

DTSE steps increase the addressing complexity of transformed applications, by introducing more complex index expressions, conditions ..

This overhead is neglected by the DTSE methodology.

The optimized memory system, will be power efficient but slow.

Parts of the additional complexity can be removed by source code optimizations for *address optimization* (ADOPT):

- algebraic cost minimization first minimizes operation instances. The goal is to find factorizations of addressing expressions in order to be able to reuse computations as much as possible. The reuse of expressions is based on common subexpression elimination techniques.

Address optimization (ADOPT)

High penalty by DTSE addressing using modulo (%) & /.

Optimization of this addressing is a major goal in ADOPT.

/ and % can be replaced by variables as follows:

```
for (j=0; j<n; j++)
  a[j/6][j%6] = ...;
```

```
int jdiv6=0, jmod6=0;
for (j=0; j<n; j++, jmod6 ++) {
  if (jmod6 >= 6) {
   jmod6 -= 6; jdiv6 ++; }
   a[jdiv6] [jmod6] = ...; }
```

☞ addressing only consists of ++ and -- operators.

In many examples, ADOPT reduces runtimes by about 3.

Backend

- Compilers
- Mapping to configurable hardware using OCAPI-XL

Cosynthesis for embedded micro-architectures (COSYMA)

Ernst et al., Univ. Braunschweig

Cosynthesis for embedded micro-architectures (COSYMA)

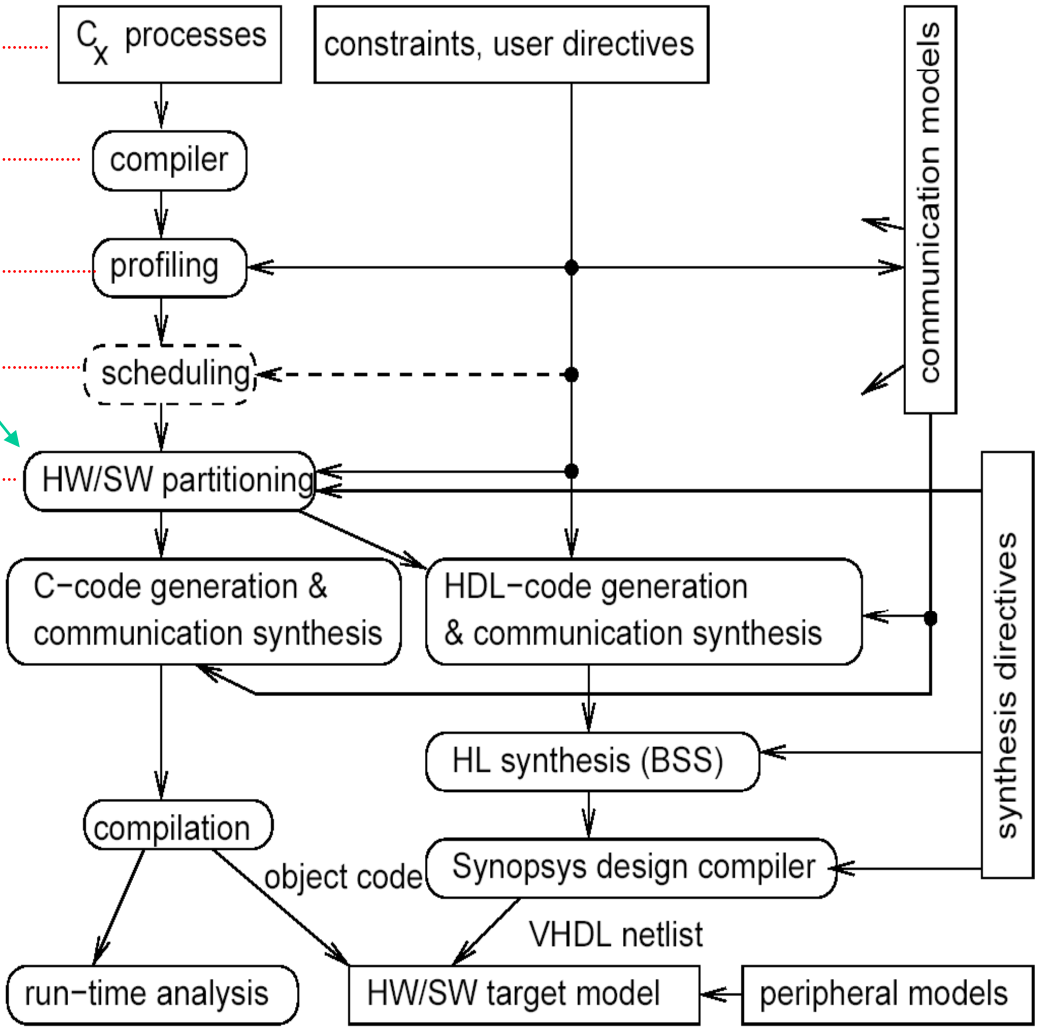
C + process header

Based on SUIF

Simulation + analytical

1 process or integrated in part.

Granularity=basic block



Ptolemy II

Ptolemy II supports specifications using different models of computation. In particular, it supports:

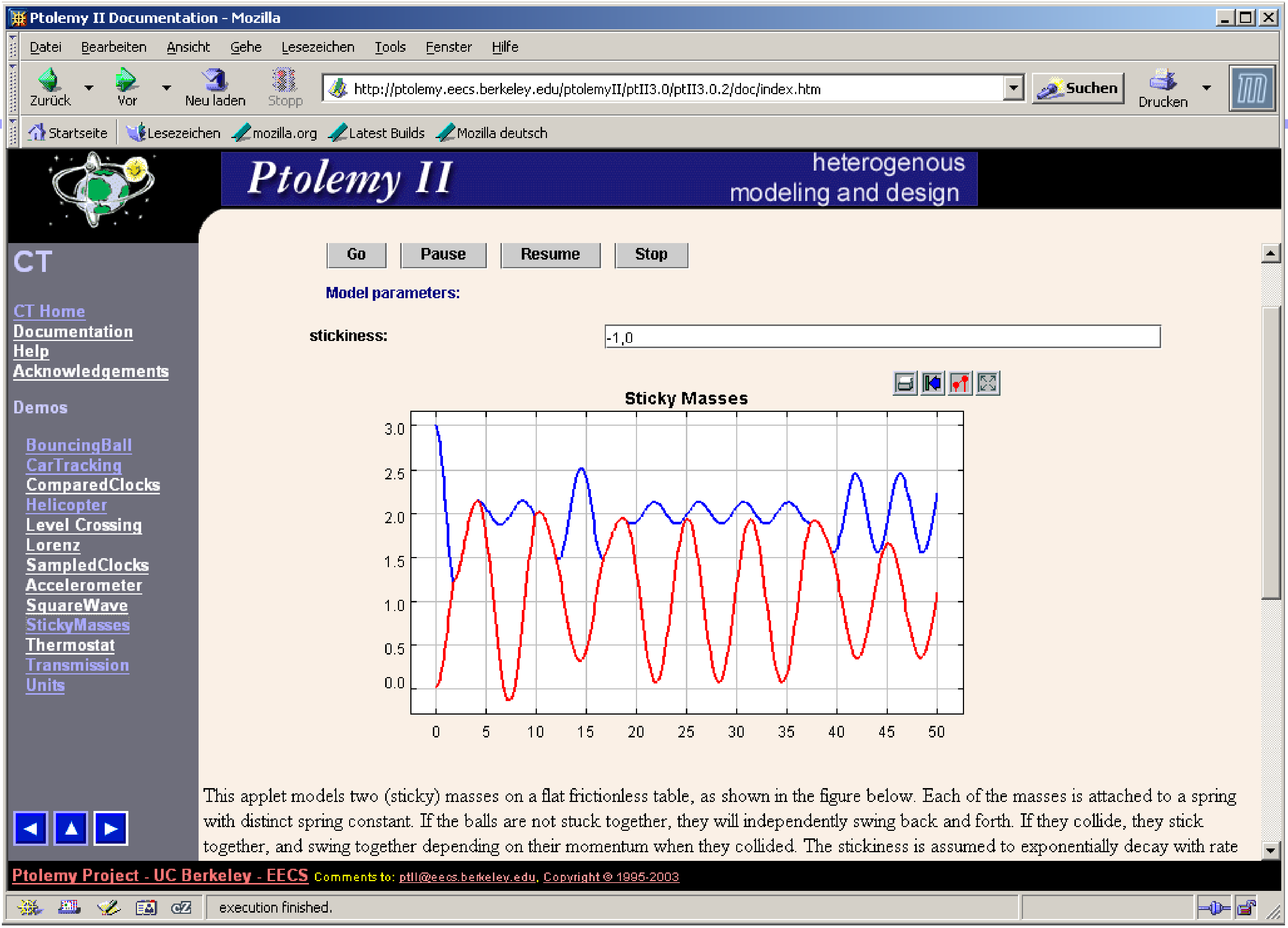
1. Communicating sequential processes (CSP).
2. Continuous time (CT): appropriate for ME, analog circuits. Supported by extensible differential equation solvers.
3. Discrete event model (DE): model used by many (e.g. VHDL) simulators.
4. Distributed discrete events (DDE).
5. Finite state machines (FSM).
6. Process networks (PN), using Kahn process networks
7. Synchronous dataflow (SDF)
8. Synchronous/reactive (SR) MoC. Discrete time, signals do not need to have a value at every clock tick. Esterel used.

Ptolemy II

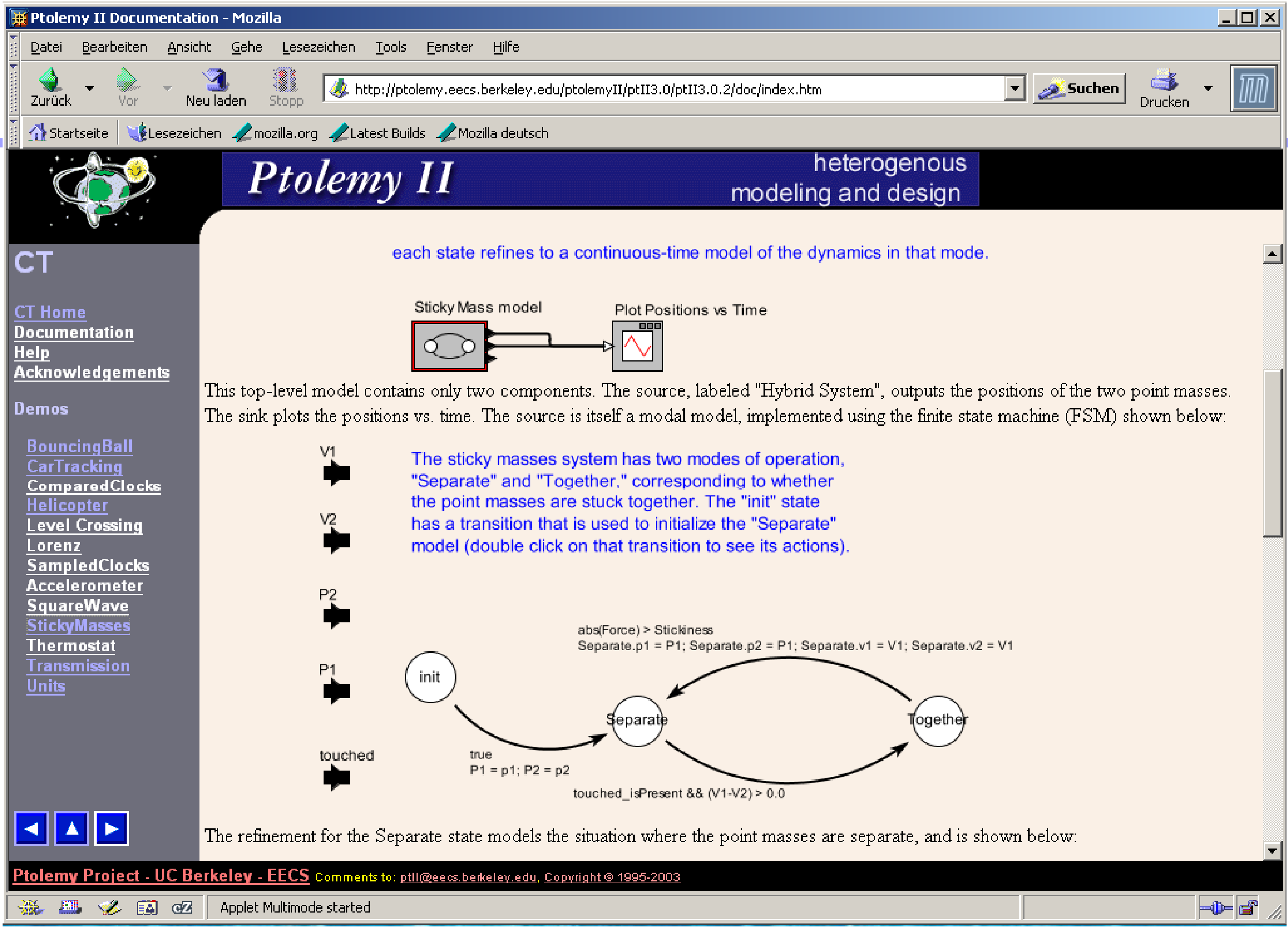
Source …

http://ptolemy.eecs.berkeley.edu/ptolemyII/ptII3.0/ptII3.0.2/doc/index.htm

Show Ptolemy II


Ptolemy II Documentation - Mozilla
Datei Bearbeiten Ansicht Gehe Lesezeichen Tools Fenster Hilfe
Zurück Vor Neu laden Stopp
http://ptolemy.eecs.berkeley.edu/ptolemyII/ptII3.0/ptII3.0.2/doc/index.htm
Suchen Drucken
Startseite Lesezeichen mozilla.org Latest Builds Mozilla deutsch
Ptolemy II
heterogenous modeling and design
CT
CT Home
Documentation
Help
Acknowledgements
Demos
BouncingBall
CarTracking
ComparedClocks
Helicopter
Level Crossing
Lorenz
SampledClocks
Accelerometer
SquareWave
StickyMasses
Thermostat
Transmission
Units
Go Pause Resume Stop
Model parameters:
stickiness:
-1,0
Sticky Masses
3.0 2.5 2.0 1.5 1.0 0.5 0.0
0 5 10 15 20 25 30 35 40 45 50
Ptolemy Project - UC Berkeley - EECS
Comments to: ptII@eecs.berkeley.edu, Copyright © 1995-2003
execution finished.

This applet models two (sticky) masses on a flat frictionless table, as shown in the figure below. Each of the masses is attached to a spring with distinct spring constant. If the balls are not stuck together, they will independently swing back and forth. If they collide, they stick together, and swing together depending on their momentum when they collided. The stickiness is assumed to exponentially decay with rate


Ptolemy II Documentation - Mozilla
Datei Bearbeiten Ansicht Gehe Lesezeichen Tools Fenster Hilfe
Zurück Vor Neu laden Stopp
http://ptolemy.eecs.berkeley.edu/ptolemyII/ptII3.0/ptII3.0.2/doc/index.htm
Suchen Drucken
Startseite Lesezeichen mozilla.org Latest Builds Mozilla deutsch
Ptolemy II
heterogenous modeling and design
CT
CT Home
Documentation
Help
Acknowledgements
Demos
BouncingBall
CarTracking
ComparedClocks
Helicopter
Level Crossing
Lorenz
SampledClocks
Accelerometer
SquareWave
StickyMasses
Thermostat
Transmission
Units
Sticky Mass model
Plot Positions vs Time
V1
V2
P2
P1
touched
init
Separate
Together
abs(Force) > Stickiness
Separate.p1 = P1; Separate.p2 = P1; Separate.v1 = V1; Separate.v2 = V1
true
P1 = p1; P2 = p2
touched_isPresent && (V1-V2) > 0.0
Applet Multimode started

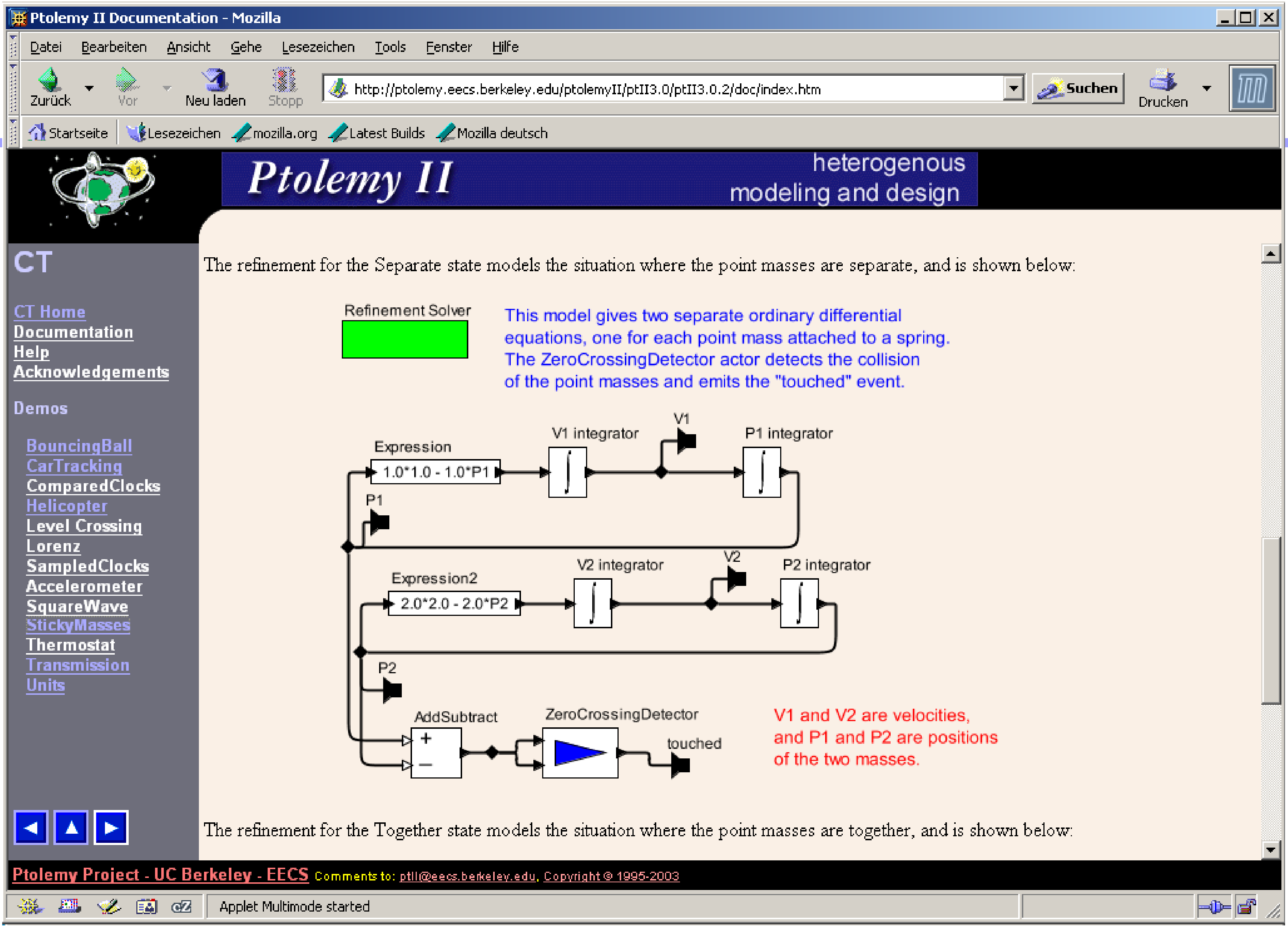
each state refines to a continuous-time model of the dynamics in that mode.

This top-level model contains only two components. The source, labeled "Hybrid System", outputs the positions of the two point masses. The sink plots the positions vs. time. The source is itself a modal model, implemented using the finite state machine (FSM) shown below:

The sticky masses system has two modes of operation, "Separate" and "Together," corresponding to whether the point masses are stuck together. The "init" state has a transition that is used to initialize the "Separate" model (double click on that transition to see its actions).

The refinement for the Separate state models the situation where the point masses are separate, and is shown below:

Ptolemy Project - UC Berkeley - EECS Comments to: ptII@eecs.berkeley.edu, Copyright © 1995-2003



The refinement for the Separate state models the situation where the point masses are separate, and is shown below:

This model gives two separate ordinary differential equations, one for each point mass attached to a spring. The ZeroCrossingDetector actor detects the collision of the point masses and emits the "touched" event.

V1 and V2 are velocities, and P1 and P2 are positions of the two masses.

The refinement for the Together state models the situation where the point masses are together, and is shown below:

Comments to: ptII@eecs.berkeley.edu, Copyright © 1995-2003

Octopus

Addressing the poor match between the focus of object-oriented design techniques on the software object structure and the need to allocate operations to tasks.

This poor match was the main concern that was addressed in the design of OCTOPUS.

Totally software-oriented flow used at Nokia

Octopus

1. In the **systems requirement phase**, behavior is described by use case diagrams and use cases. The structure of the environment is described by a so-called context diagram.

2 In the **system architecture phase**, the structure is broken down into subsystems. Major interfaces between the subsystems are identified, but their behavior is not.

3 The **subsystem analysis phase** is done ∀ subsystems. Class diagrams for the subsystems are generated.
Behaviors of subsystems can be defined in various ways, including StateCharts, so-called event lists and event sheets.

4 The **subsystem design phase** generates outlines for processes/threads, classes and interprocess messages.

5 The **subsystem implementation phase** generates actual code of the selected programming language.