ESE Front End 2.0

D. Gajski, S. Abdi, Y. Hwang, L. Yu, H. Cho, I. Viskic

Center for Embedded Computer Systems

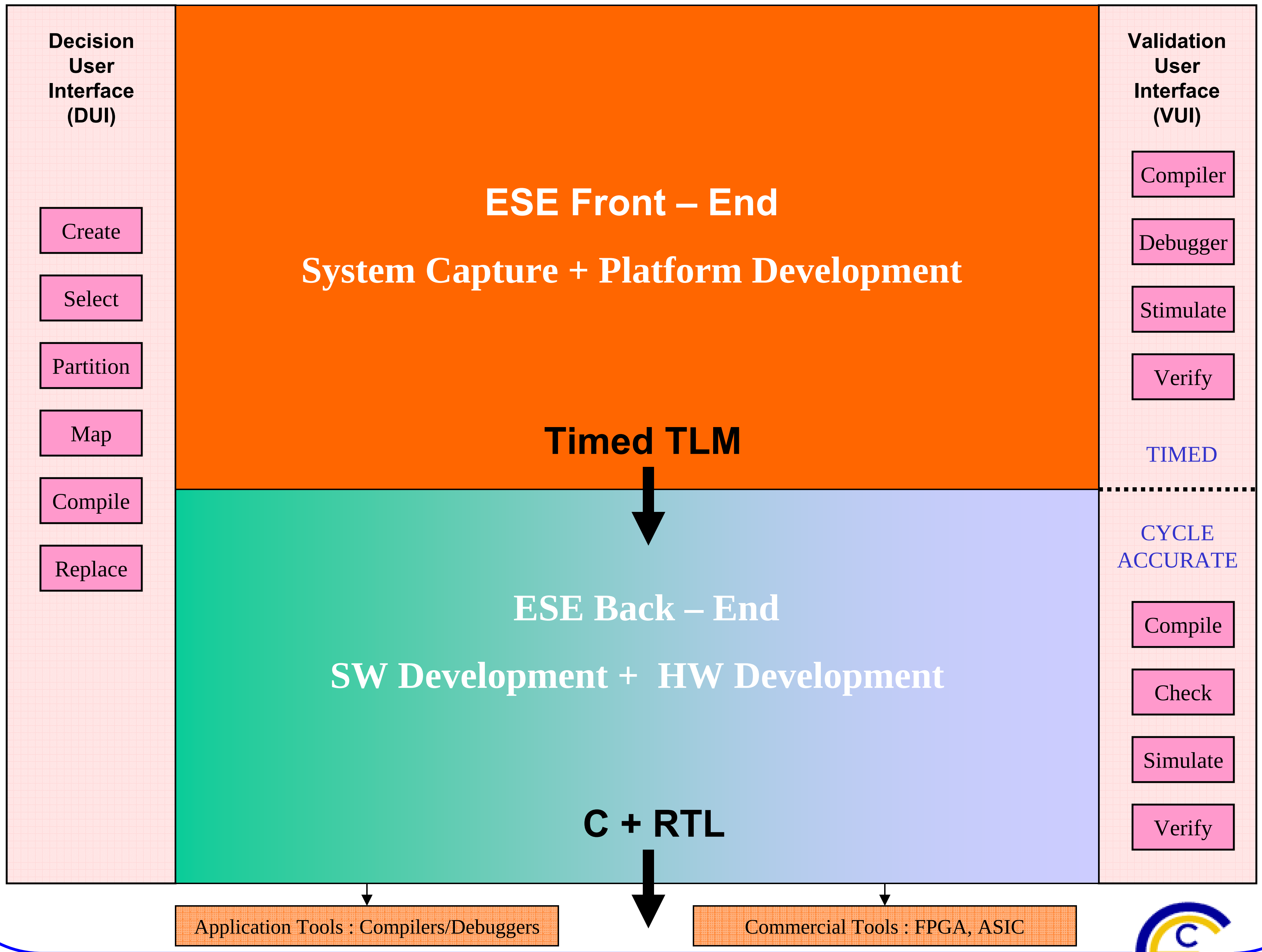
University of California, Irvine

http://www.cecs.uci.edu

Technology Advantages

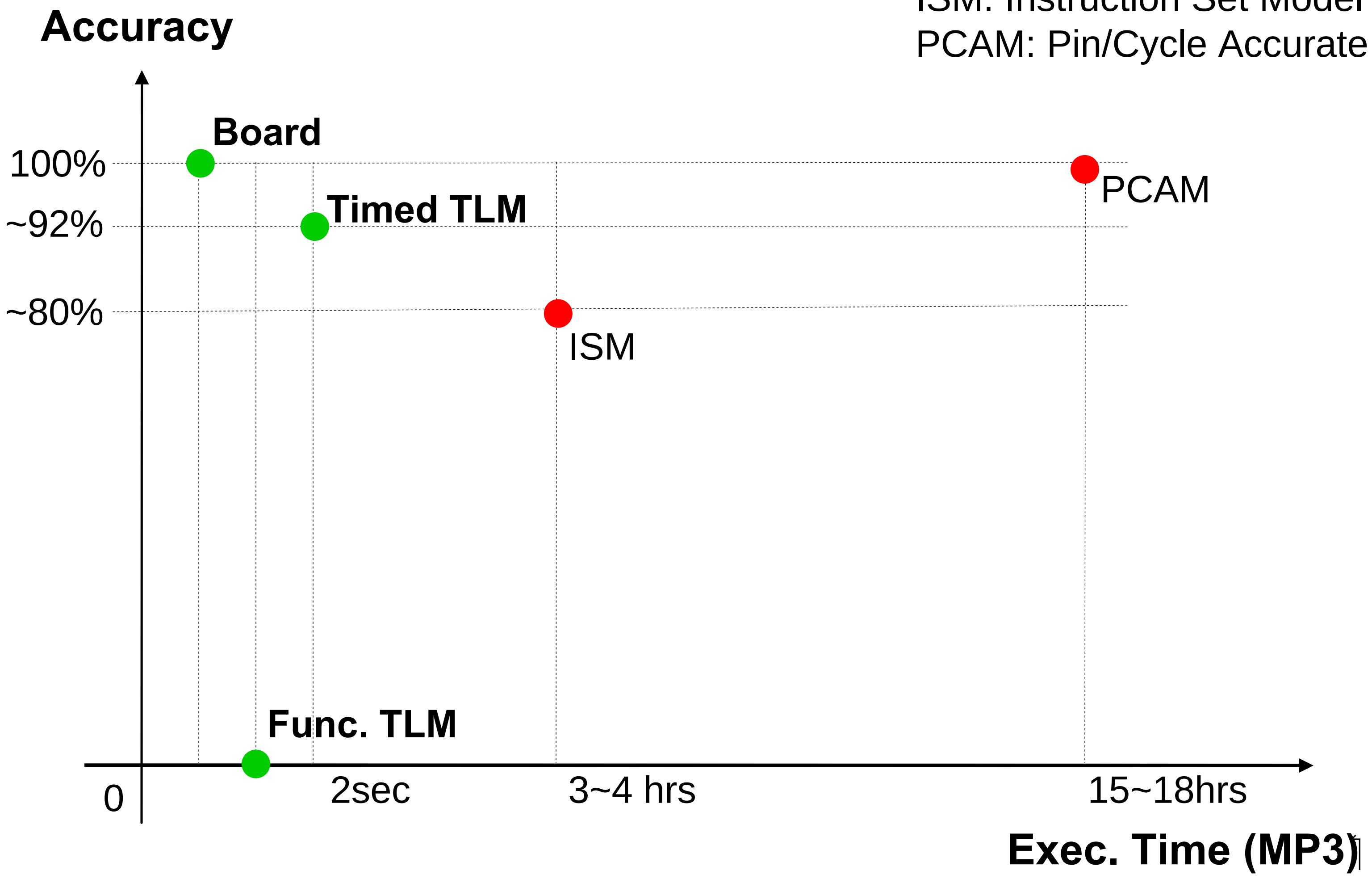
- No basic change in design methodology required
  - ESE supported design follows present design process
- Productivity gain of more than 1000X demonstrated
  - Designers do not write models
- Simple design update: 1-day change
  - No rework for new design decisions
- High error-reduction: Automation + verification
  - Error-prone tasks are automated
- Simplified globally-distributed design
  - Fast exchange of design decisions and easy impact estimates
- Benefit through derivatives designs
  - No need for complete redesign
- Better market penetration through customization
- Shorter Time-to-Market through automation

ES Environment


Decision User Interface (DUI)
Create
Select
Partition
Map
Compile
Replace
ESE Front – End
System Capture + Platform Development
Timed TLM
ESE Back – End
SW Development + HW Development
C + RTL
Validation User Interface (VUI)
Compiler
Debugger
Stimulate
Verify
TIMED
CYCLE ACCURATE
Compile
Check
Simulate
Verify
Application Tools : Compilers/Debuggers
Commercial Tools : FPGA, ASIC

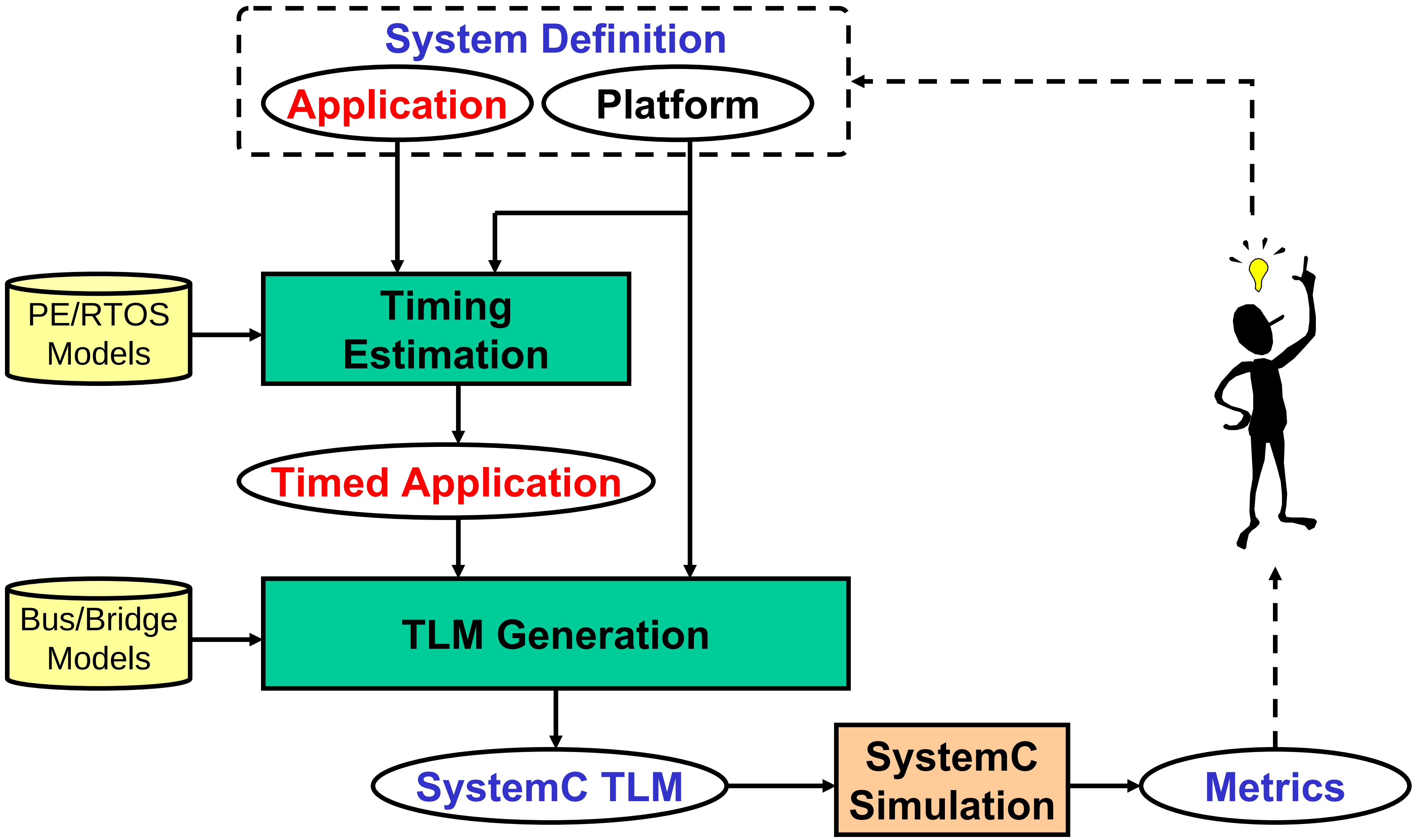
Model Accuracy vs. Execution Time

TLM: Transaction Level Model
ISM: Instruction Set Model
PCAM: Pin/Cycle Accurate Model

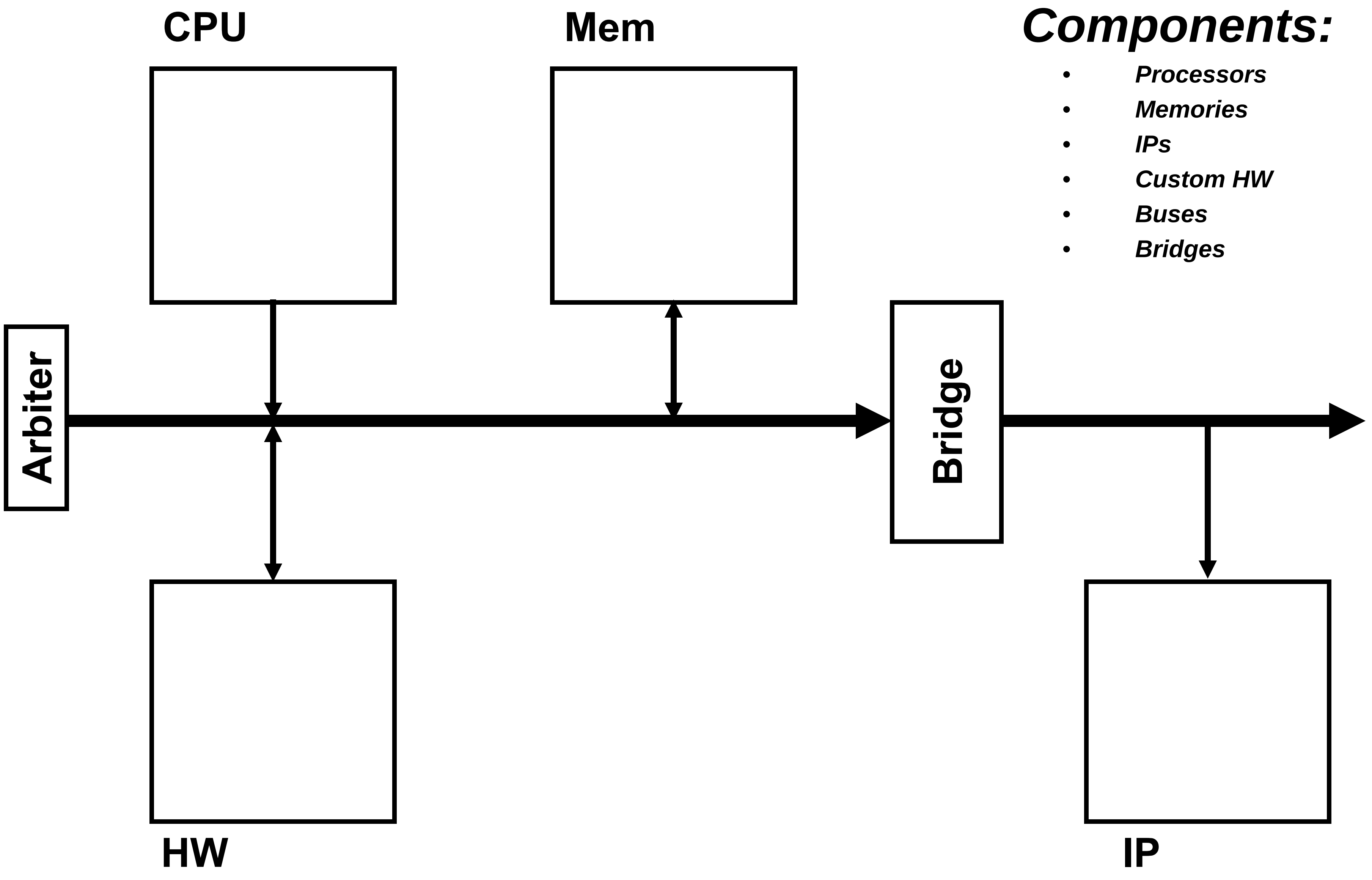


Time and accuracy trade off among different models

ESE Front End Tool Flow



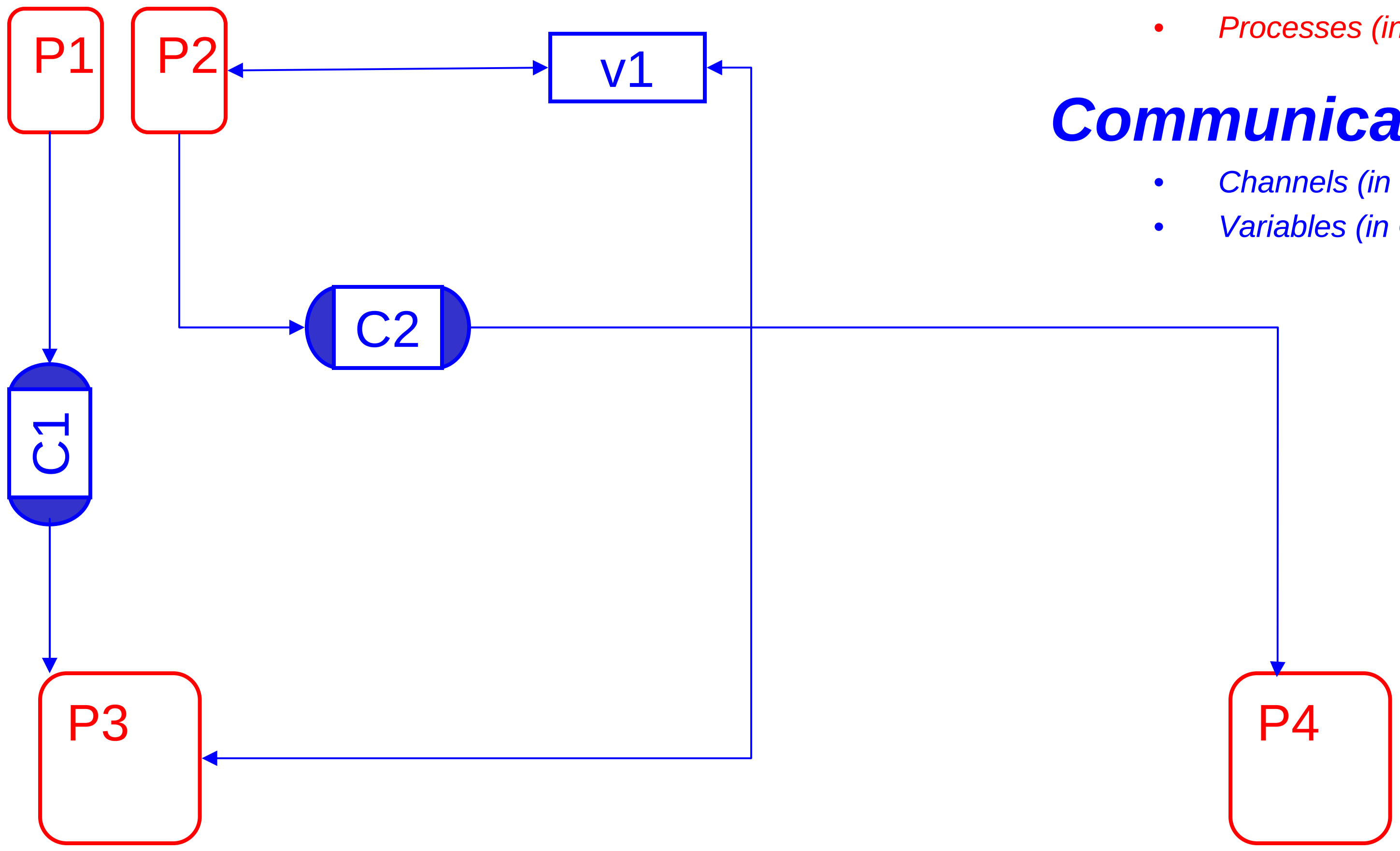
Platform Architecture



Components:

- *Processors*
- *Memories*
- *IPs*
- *Custom HW*
- *Buses*
- *Bridges*

Application Spec



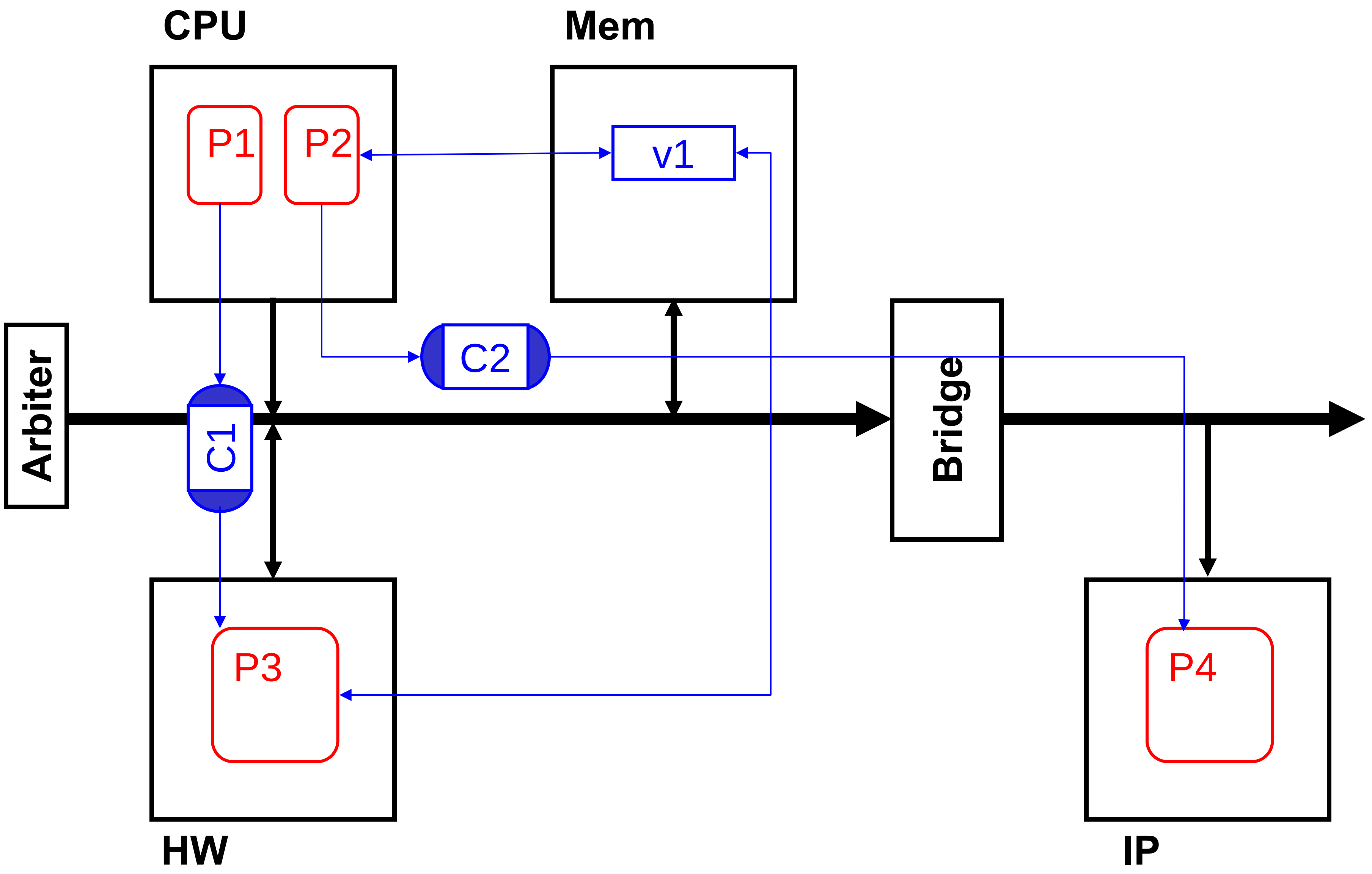
Computation

- *Processes (in C)*

Communication

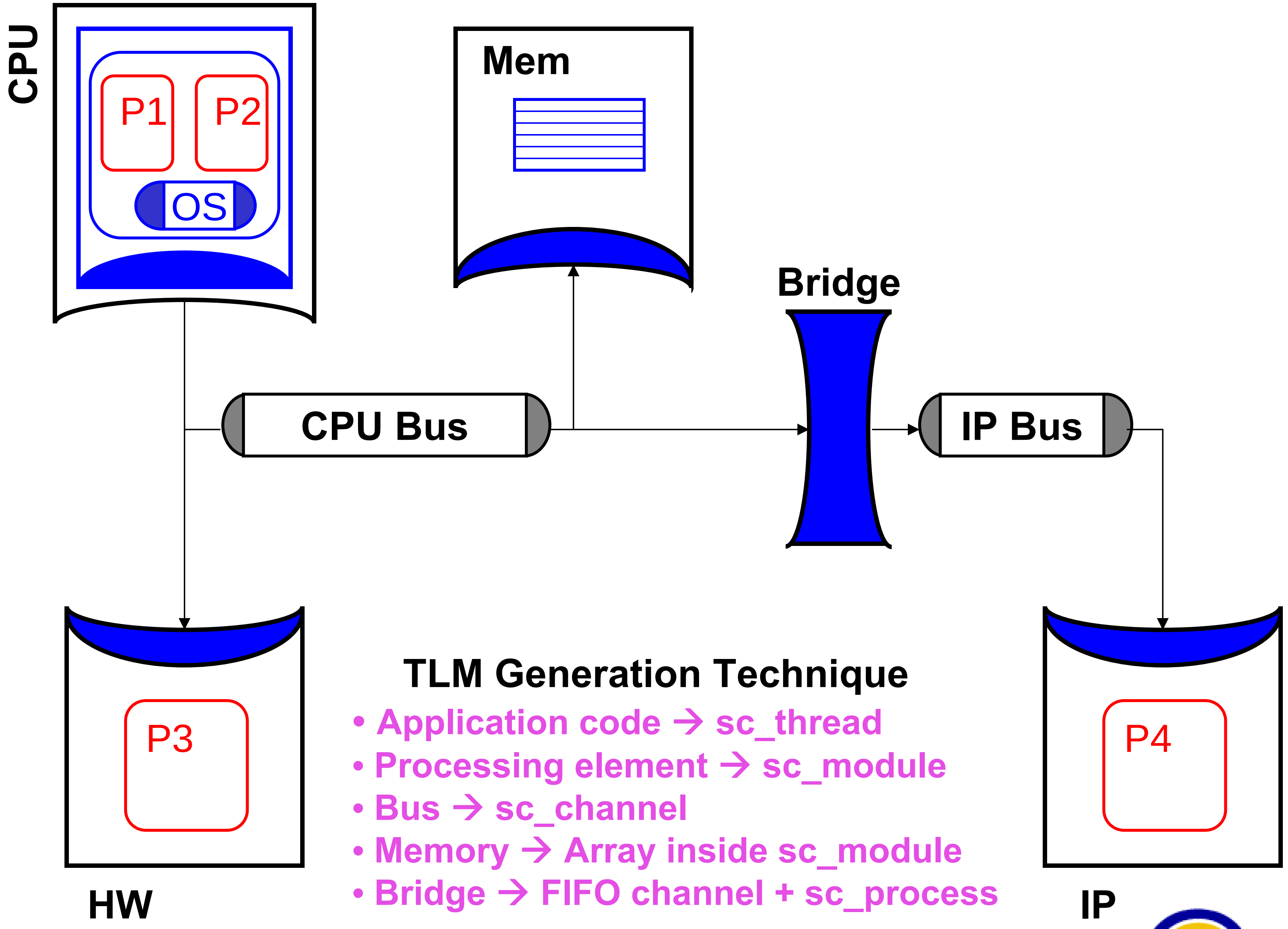
- *Channels (in C)*
- *Variables (in C)*

Input: System Definition



System Definition* = *Platform* + *Application

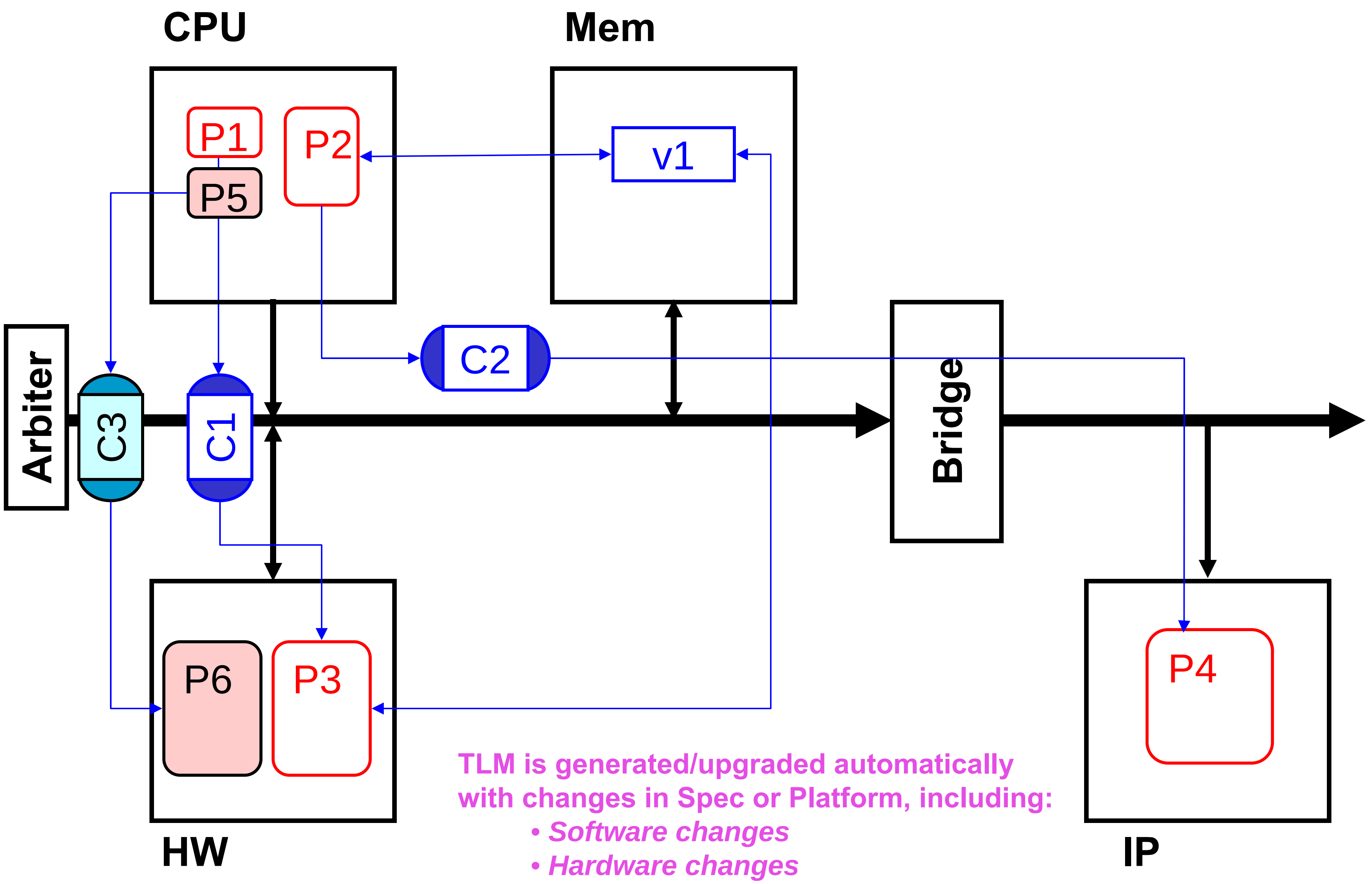
Output: SystemC Timed TLM



TLM Generation Technique

- Application code → sc_thread
- Processing element → sc_module
- Bus → sc_channel
- Memory → Array inside sc_module
- Bridge → FIFO channel + sc_process

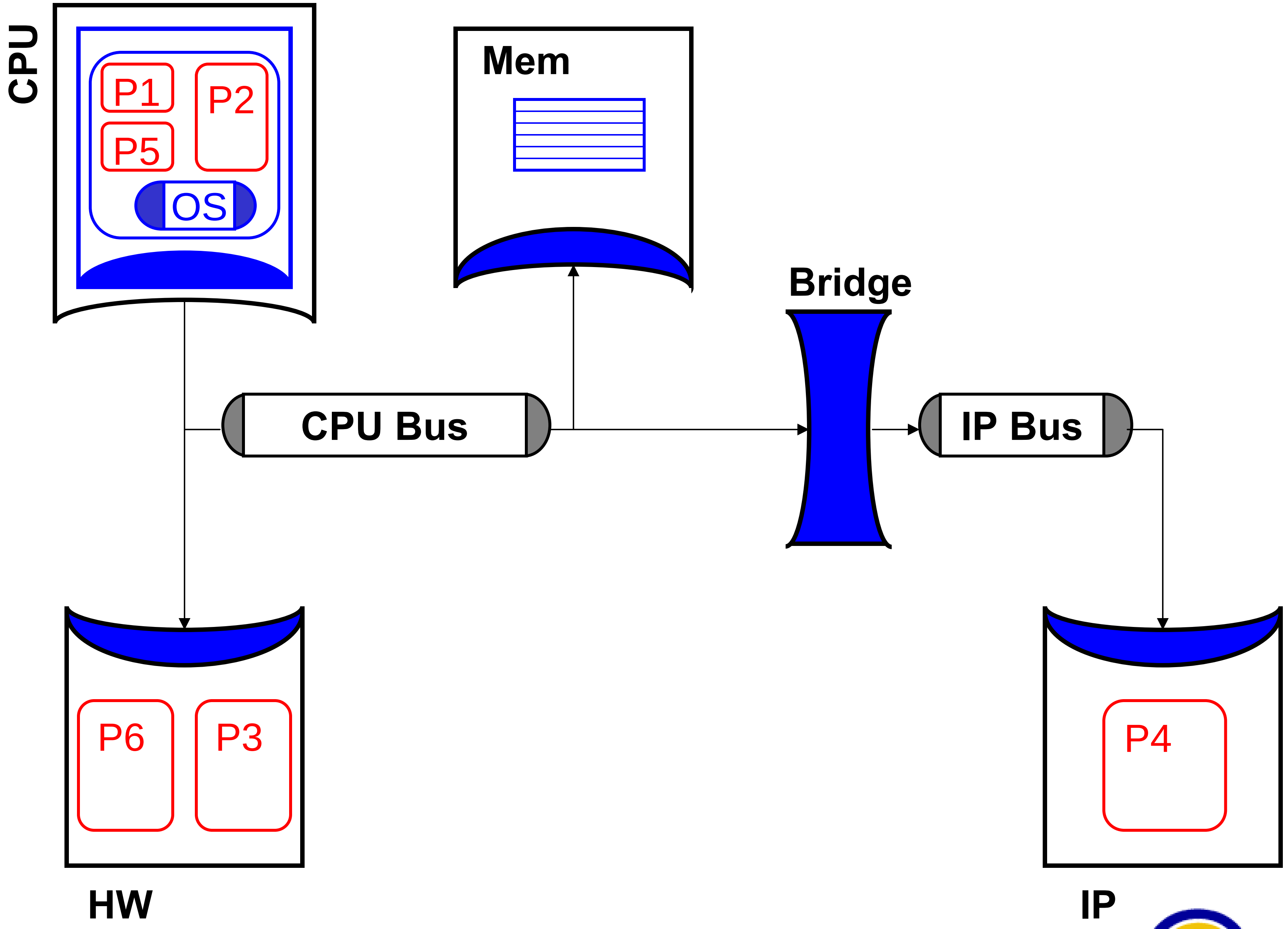
System Modifications



TLM is generated/upgraded automatically with changes in Spec or Platform, including:

- *Software changes*
- *Hardware changes*
- *Communication changes*

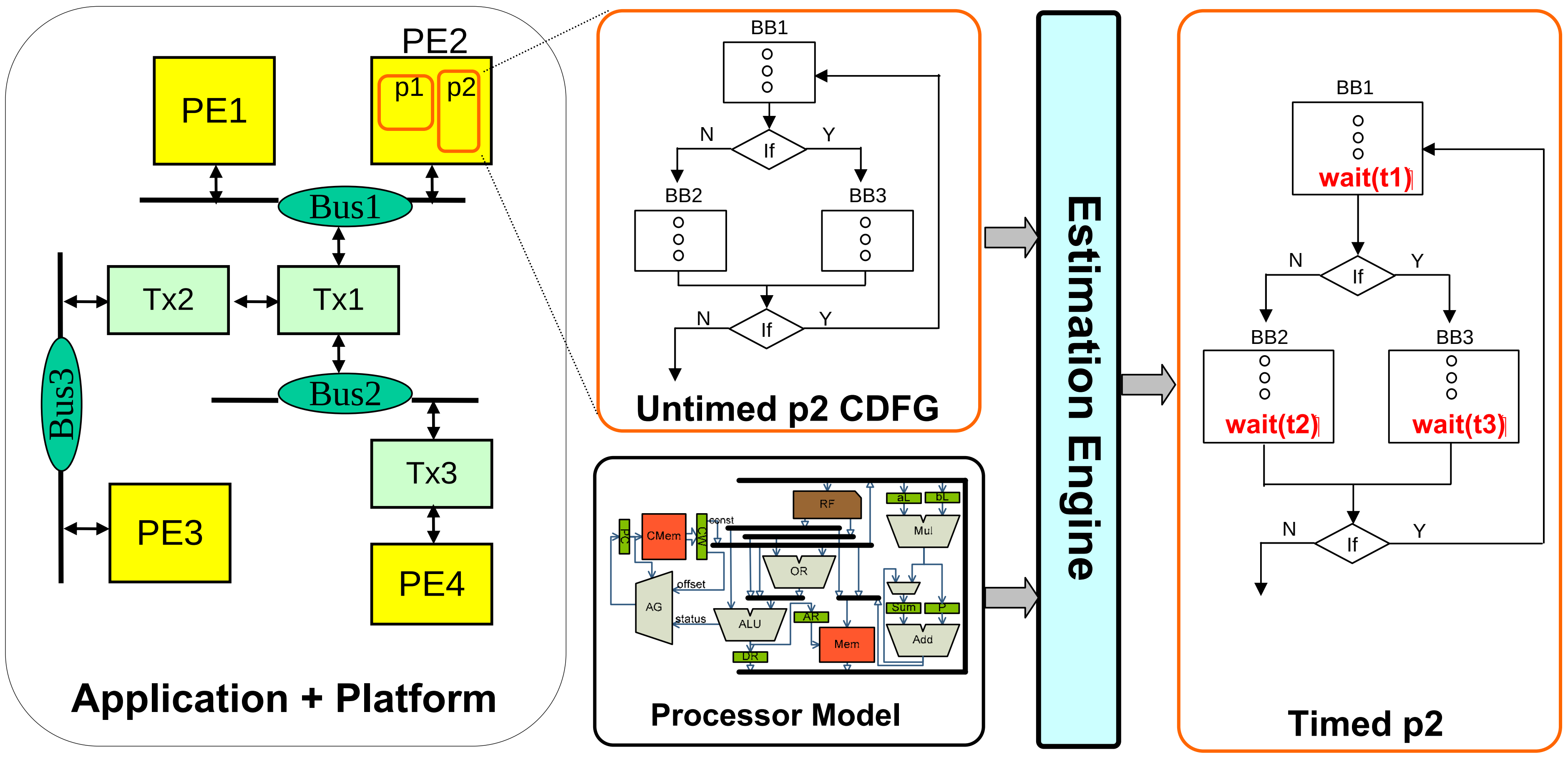
Output: Modified TLM



TLM Generation Features

- Processing Elements (PEs)
  - Any number of processes mapped to any PE
  - Any number of bus connections
- Connectivity
  - Point-to-point links
  - Shared bus architecture
  - Multi-hop transactions
  - NoC platforms
- Bridges and routers
  - Any size, number and partition of FIFOs
  - Any number of bus connections
  - Static and dynamic routing
- Memories
  - Any number of bus connections
  - Local (inside PE) and shared memories

Timing Estimation Technique

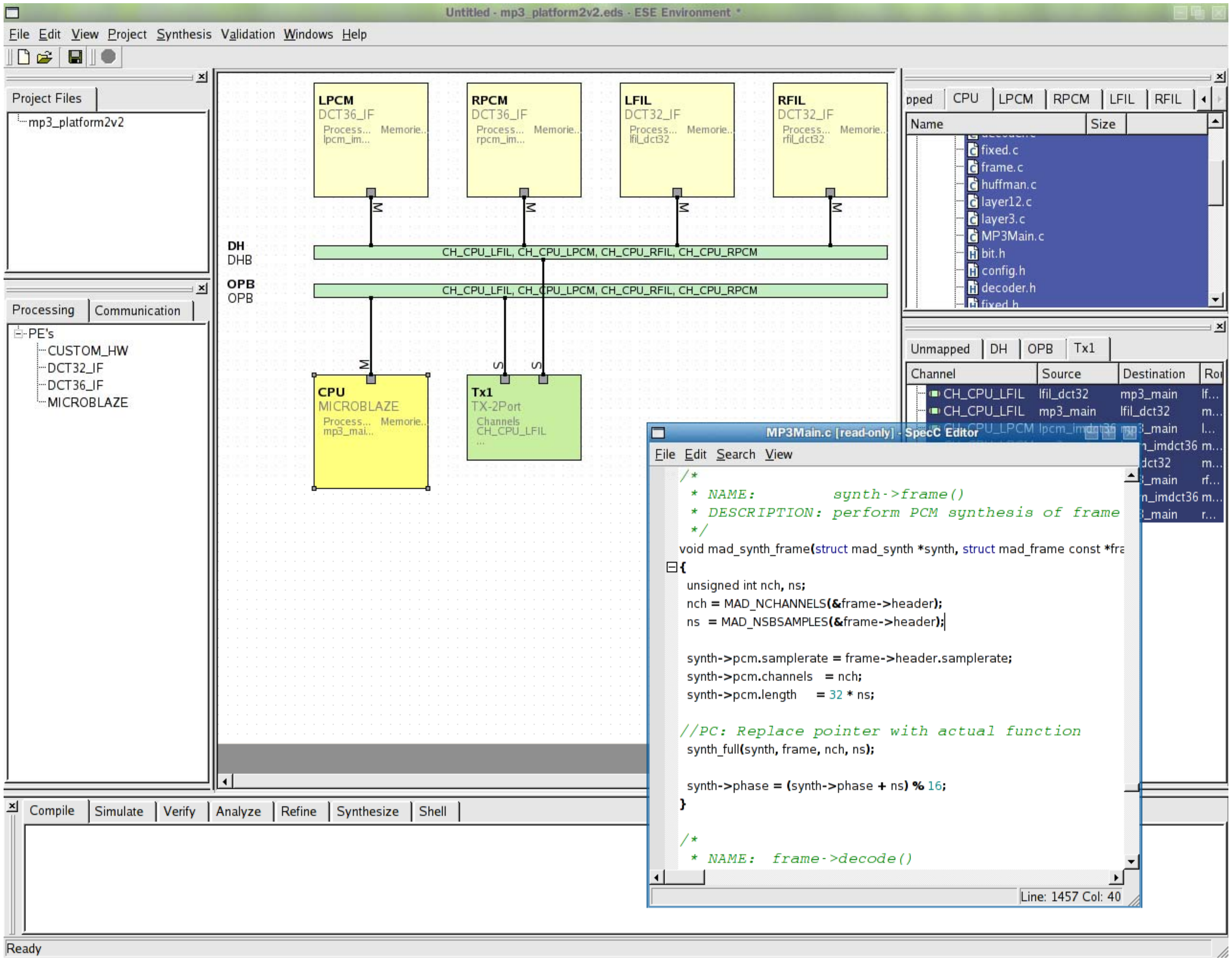


- DFG scheduling to compute basic block delay
- RTOS model added for PEs with multiple processes

Timing Estimation Features

- Retargetable Processor Models
  - Any type of control/datapath pipelining
  - Any number of pipelined datapaths
  - Multi-cycle units, forwarding, chaining
  - Branch prediction
  - VLIW and SuperScalar
- Statistical/Dynamic Cache Models
- RTOS models
- Integration with high level synthesis for custom HW
- Estimation reports
  - Basic block level, function level and transaction level

ESE: Platform and Application Capture



Untitled - mp3_platform2v2.eds - ESE Environment *

File Edit View Project Synthesis Validation Windows Help

Project Files

mp3_platform2v2

Processing | Communication

- PE's
  - CUSTOM_HW
  - DCT32_IF
  - DCT36_IF
  - MICROBLAZE

pped | CPU | LPCM | RPCM | LFIL | RFIL

Name	Size
fixed.c	
frame.c	
huffman.c	
layer12.c	
layer3.c	
MP3Main.c	
bit.h	
config.h	
decoder.h	
fixed.h	

Unmapped | DH | OPB | Tx1

Channel	Source	Destination	Ro
CH_CPU_LFIL	lfil_dct32	mp3_main	lf...
CH_CPU_LFIL	mp3_main	lfil_dct32	m...
CH_CPU_LPCM	lpcm_imdct36	mp3_main	l...
		n_imdct36	m...
		dct32	m...
		_main	rf...
		n_imdct36	m...
		_main	r...

MP3Main.c [read-only] - SpecC Editor

File Edit Search View

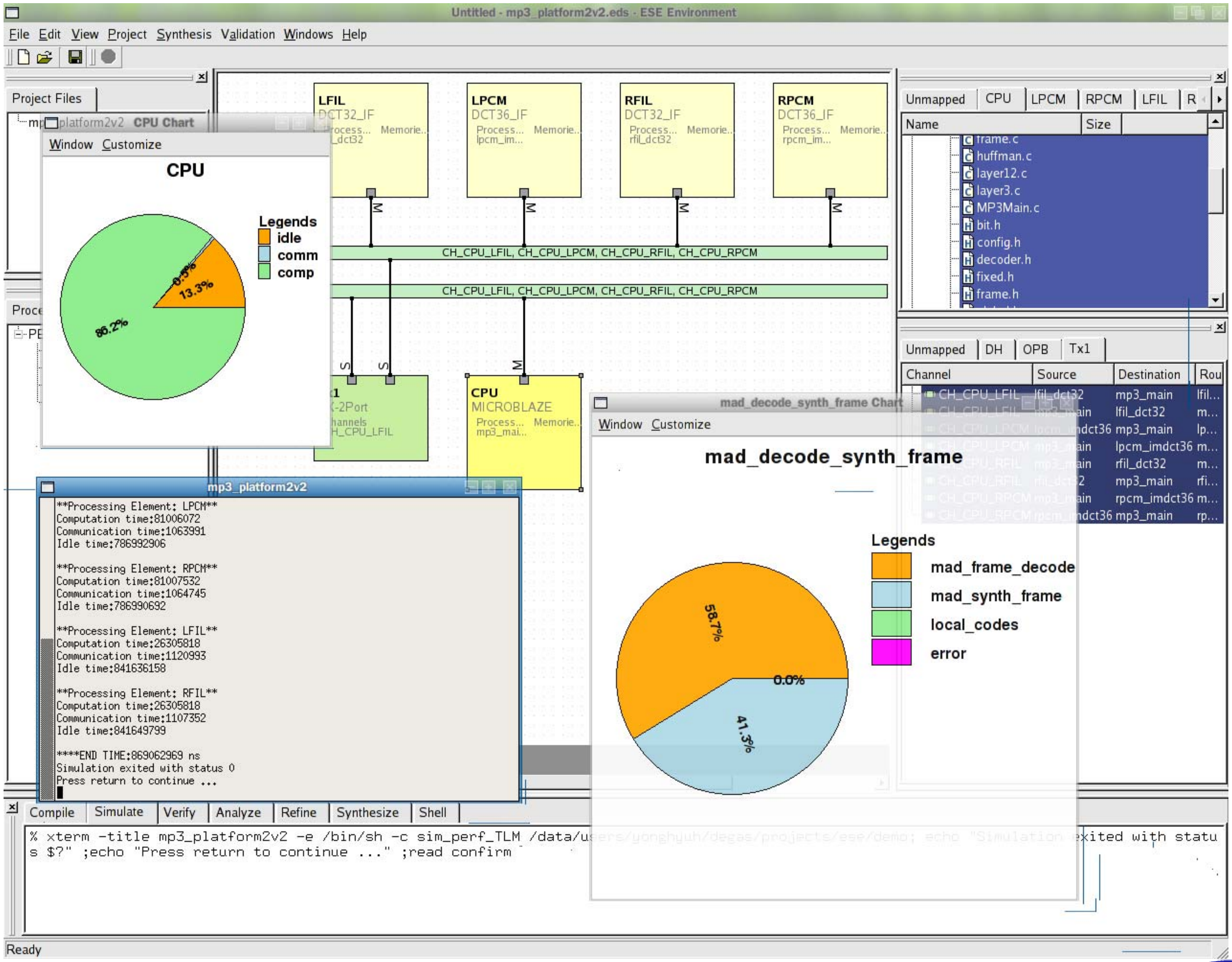
```
/*
 * NAME:         synth->frame()
 * DESCRIPTION: perform PCM synthesis of frame
 */
void mad_synth_frame(struct mad_synth *synth, struct mad_frame const *fra
{
 unsigned int nch, ns;
 nch = MAD_NCHANNELS(&frame->header);
 ns  = MAD_NSBSAMPLES(&frame->header);

 synth->pcm.samplerate = frame->header.samplerate;
 synth->pcm.channels   = nch;
 synth->pcm.length     = 32 * ns;

//PC: Replace pointer with actual function
 synth_full(synth, frame, nch, ns);

 synth->phase = (synth->phase + ns) % 16;
}

/*
 * NAME:  frame->decode()
```

Line: 1457 Col: 40

Compile | Simulate | Verify | Analyze | Refine | Synthesize | Shell

Ready

ESE: TLM Generation and Estimation



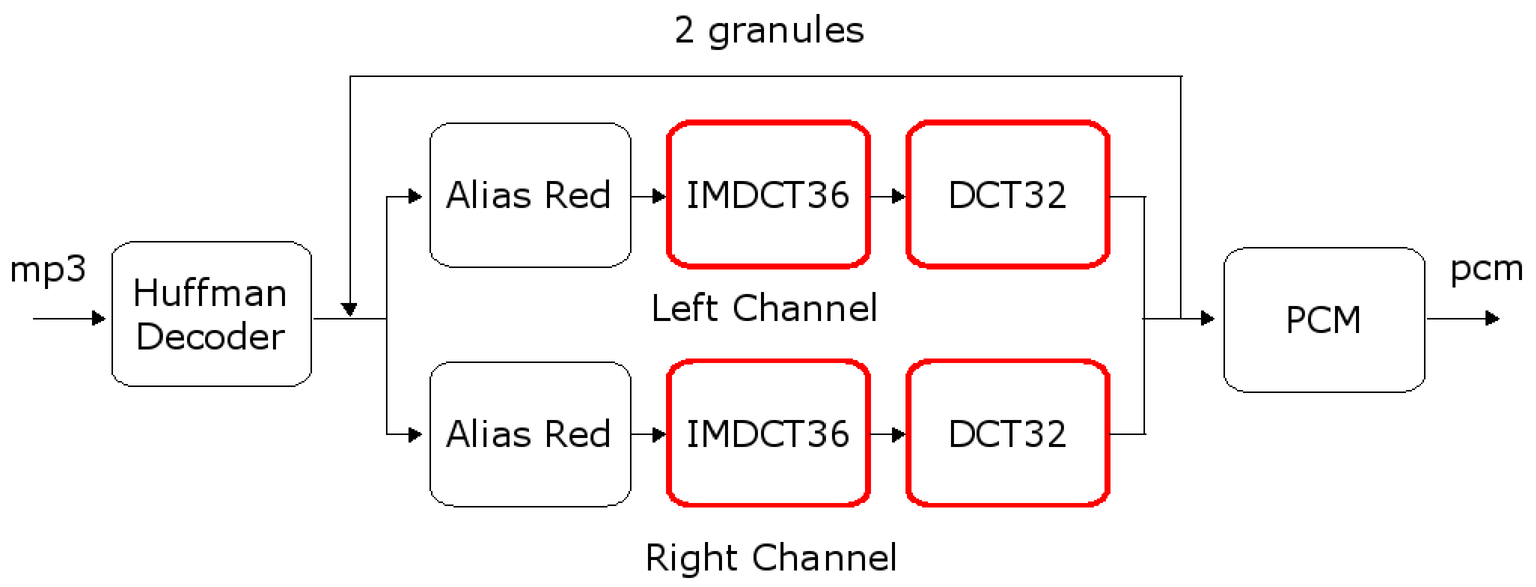
Channel	Source	Destination	Rou
CH_CPU_LFIL	lfil_dct32	mp3_main	lfil...
CH_CPU_LFIL	mp3_main	lfil_dct32	m...
CH_CPU_LPCM	lpcm_imdct36	mp3_main	lp...
CH_CPU_LPCM	mp3_main	lpcm_imdct36	m...
CH_CPU_RFIL	mp3_main	rfil_dct32	m...
CH_CPU_RFIL	rfil_dct32	mp3_main	rfi...
CH_CPU_RPCM	mp3_main	rpcm_imdct36	m...
CH_CPU_RPCM	rpcm_imdct36	mp3_main	rp...

```
**Processing Element: LPCM**
Computation time:81006072
Communication time:1063991
Idle time:786992906

**Processing Element: RPCM**
Computation time:81007532
Communication time:1064745
Idle time:786990692

**Processing Element: LFIL**
Computation time:26305818
Communication time:1120993
Idle time:841636158

**Processing Element: RFIL**
Computation time:26305818
Communication time:1107352
Idle time:841649799

****END TIME:869062969 ns
Simulation exited with status 0
Press return to continue ...
```

```
% xterm -title mp3_platform2v2 -e /bin/sh -c sim_perf_TLM /data/users/yonghyuh/degas/projects/ese/demo; echo "Simulation exited with statu
s $?" ;echo "Press return to continue ..." ;read confirm
```

MP3 Decoder Application

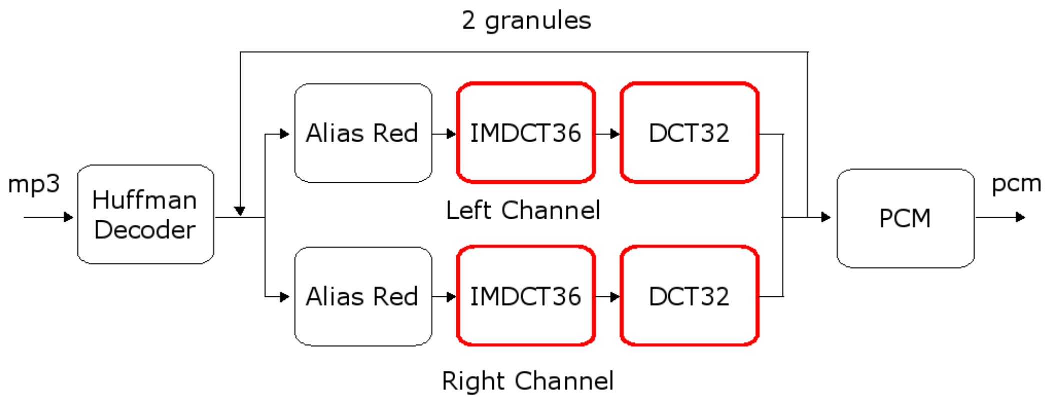
- Functional block diagram (major blocks only)

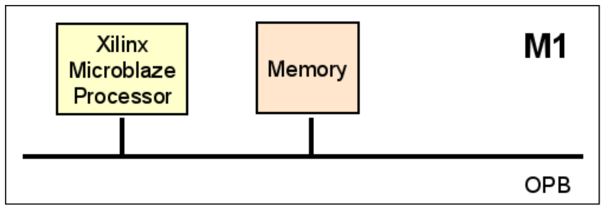


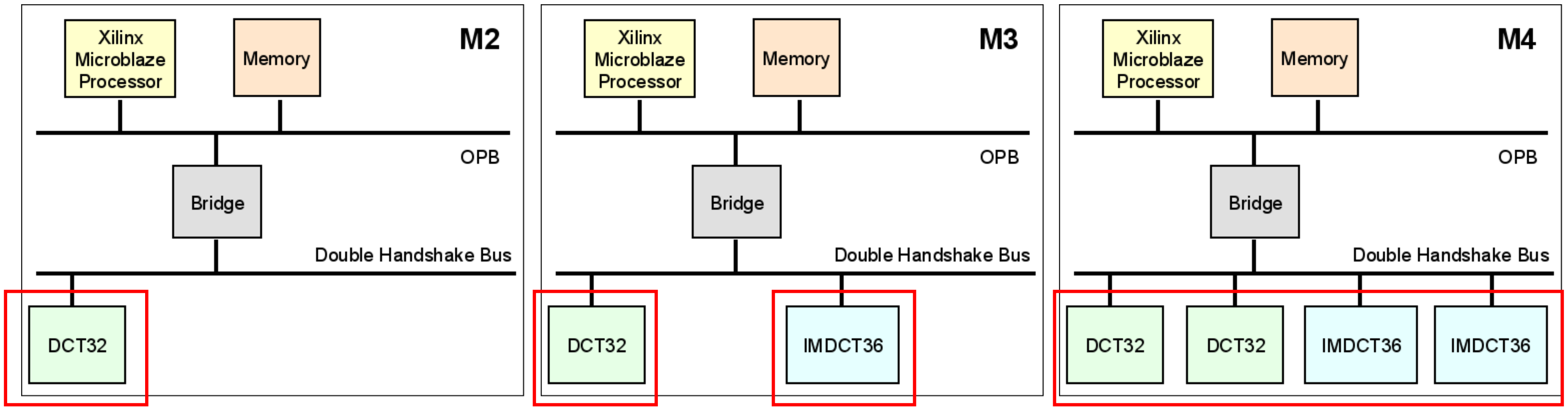
- Application features
  - 12K lines of C code
  - IMDCT and DCT are compute intensive
    - Candidates for HW implementation
  - Left channel and right channel are data independent
    - Concurrent execution possible

MP3 Platforms

- **MP3 Decoder on Xilinx Multimedia FPGA board**
  - Microblaze soft-core with 0/1/2/4 HW components







Results: Functional TLM Generation and Simulation

Design	SystemC LoC	Manual Coding	Func. TLM Generation	Func. TLM Simulation
M1	2095	2 weeks	0.63 s	0.01 s
M2	2894	3 weeks	0.66 s	0.01 s
M3	3148	4 weeks	0.66 s	0.01 s
M4	3653	4 weeks	0.74 s	0.01 s
Average	**2948**	**~3 weeks**	**~ 0.7 s**	**0.01 s**

- **Functional TLM generation in seconds vs. weeks of manual coding**
  - **Huge productivity gain**
- **Functional TLM simulation in fraction of a second**
  - **Early application development and debugging**

Results: Estimation Quality

Error %= (1 - Estimated cycles/ Board Cycles)*100

ISM Error

Cache size	M1	
	Board	ISM Error
0K/0K	27215K	39.48%
2K/2K	8914K	18.38%
8K/4K	5828K	3.55%
16K/16K	4413K	-16.32%
32K/16K	4384K	-16.60%
Average	N/A	**18.86%**

Timed TLM Error

Cache Size	M1	M2	M3	M4
0K/0K	6.27%	9.00%	18.18%	18.61%
2K/2K	6.68%	-7.16%	-15.79%	-9.35%
8K/4K	4.74%	9.13%	-1.66%	-0.18%
16K/16K	-13.83%	4.66%	2.63%	3.65%
32K/16K	-13.89%	-8.29%	1.57%	2.29%
Average	**9.08%**	**7.65%**	**7.97%**	**6.82%**

- **TLM estimation applicable to all designs**
  - **ISM only available for SW**
- **TLM estimation error < ½ of ISM error**
  - **Reliable design exploration with timed TLMs**

Results: Timed TLM Generation and Simulation

Timed TLM Generation	Timed TLM Simulation	ISM Sim.	CA Sim.
31 s	0.01 s	3.6 h	16 h
50 s	0.22 s	N/A	18 h
47 s	0.25 s		18 h
71 s	0.36 s		18 h
~ 1min	**~ 0.2 s**	**3.6 h**	**~ 18 h**

- **Timed TLM generated in minutes vs. hours of CA/ISM simulation**
  - **Early SW/HW performance estimation**
- **Timed TLM simulation in < 1 sec.**
  - **Extensive design exploration**

ESE Advantages

- Platform and Application can be easily captured using GUI
- Functional TLMs are automatically generated for development and testing of application code
- Timed TLMs are automatically generated for early design exploration
- Legacy SW and HW IPs can be easily added for design reuse and upgrade
- ESE allows concurrent development of platform SW, HW and application code

Acknowledgments

- **We would like to acknowledge the previous R&D teams who contributed many concepts and methods used in ESE 2.0**
  - SpecCharts/SpecSyn ('92): F. Vahid, S. Narayan, J. Gong, S. Bakshi
  - SpecC/SCE ('00) team: R. Doemer, J. Zhu, A. Gerstlauer, J. Peng, D. Shin, L. Cai, H. Yu
- **We also want to thank P. Chandraiah for MP3 reference code**