InterDesign Technologies

High-Speed Hardware/Software Co-Verification with CPU Model Generator from Software Code

Dai Araki InterDesign Technologies Inc.

Noriyoshi Ito Semiconductor Technology Academic Research Center (STARC)

Takao Shinsha* Semiconductor Technology Academic Research Center (STARC)

Yoshikazu Mori Oki Network LSI Co.,Ltd.

* Now he is at Applistar Corporation

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

High-speed CPU model for Co-simulation

InterDesign Technologies

Co-simulation

- Uses concurrent software and hardware modules
- Verifies HW/SW design with virtual platform
- Verify performances of system variations at various levels of abstraction, which helps to make design decisions

CPU RTL model with binary software

- Cycle-accurate but too slow

Instruction set simulator (ISS) with binary software

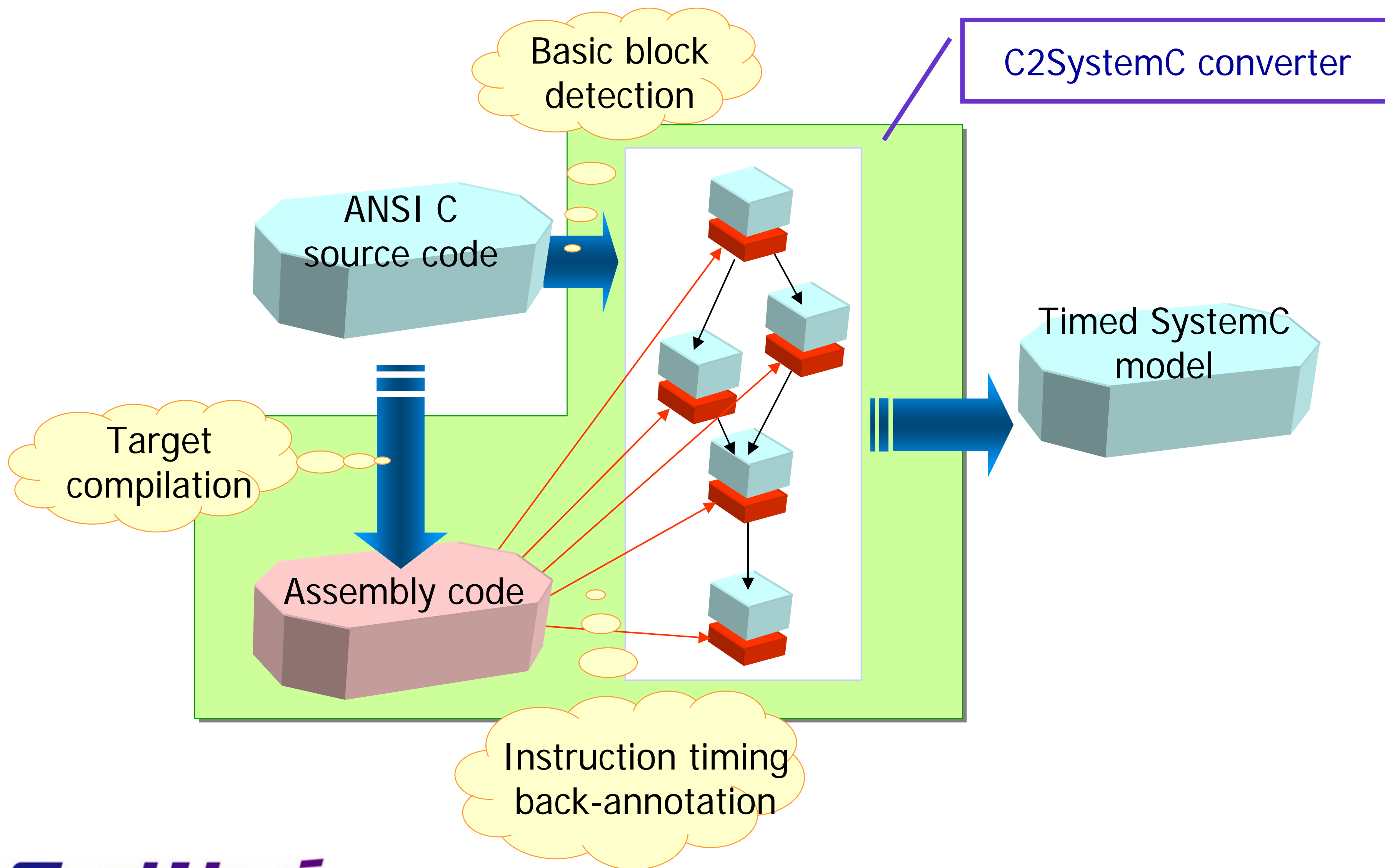
- 100k - 10M instructions/sec
- High-speed ISS loses cycle accuracy

Virtual CPU model with timing back-annotated software code

- Converts software source code into virtual CPU model in SystemC
- Over 100 M instructions/sec speed
- Cycle approximate accuracy

(C) 2006, InterDesign Technologies, Inc.

Basic Idea of Timing back-annotation



Key features of C2SystemC conversion

InterDesign Technologies

Converts ANSI C source code into SystemC (C++)

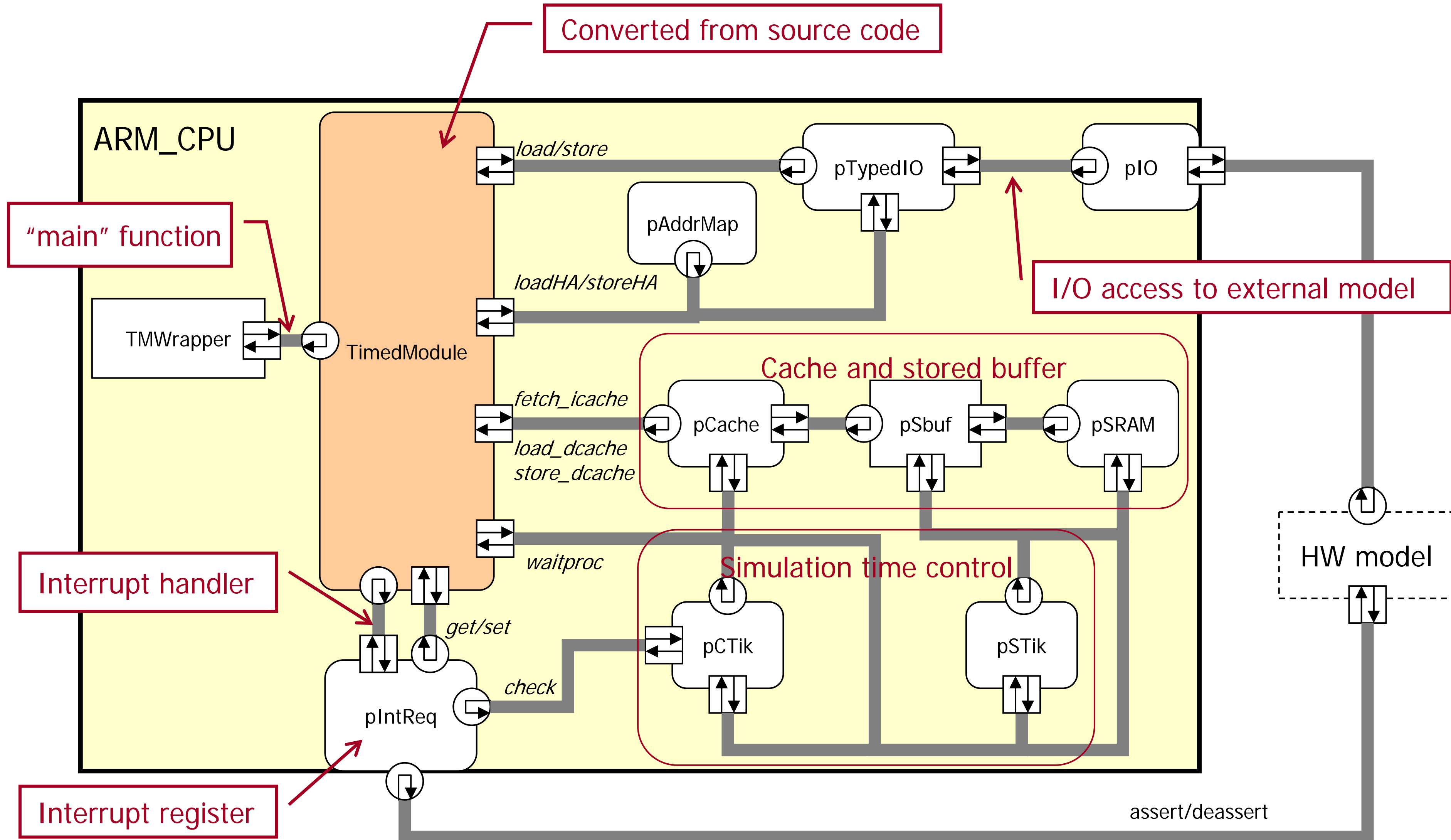
1. "Instruction Timing back-annotation"
   - Embeds delay functions which simulates target CPU instruction cycles
   - Embeds probe functions which emulate CPU cache
   - CPU configuration settings (CPU core, clock frequency, cache size etc.) can be designed at the code conversion
2. I/O access conversion
   - Detects I/O access and interrupt handler in software code, and automatically converts them into method call of SystemC channel

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

Virtual CPU model in SystemC

InterDesign Technologies



FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

Advantages of Timing Back-annotation scheme

InterDesign Technologies

a. Boosting simulation speed

- Most portion of C source code is just compiled & executed with host environment (gcc on Linux, VC++ on Windows etc.)

b. Keeping cycle accuracy

- Timing back-annotation converts un-timed software code into timed model
- Cycle approximate accuracy in few percentage error rate

c. Generating virtual CPU models in pure SystemC

- Easily connected to external SystemC models, simulators , etc.
- Free of simulation environment (Linux, windows, ...)

(C) 2006, InterDesign Technologies, Inc.

C2SystemC converter supported platform

CPU

- ARM
  - ARM7TDMI, ARM7TDMI-S ARM7EJ-S
  - ARM920T, ARM922T ARM940T
  - ARM926EJ-S, ARM946E-S, ARM966E-S
- Toshiba TX49H3
- MIPS 5kf (planned)

Cross compiler

- ARM : arm-elf-gcc, RealView (by ARM)
- Toshiba TX49, MIPS 5kf : MULTI (by Green Hills Software)

SystemC simulator

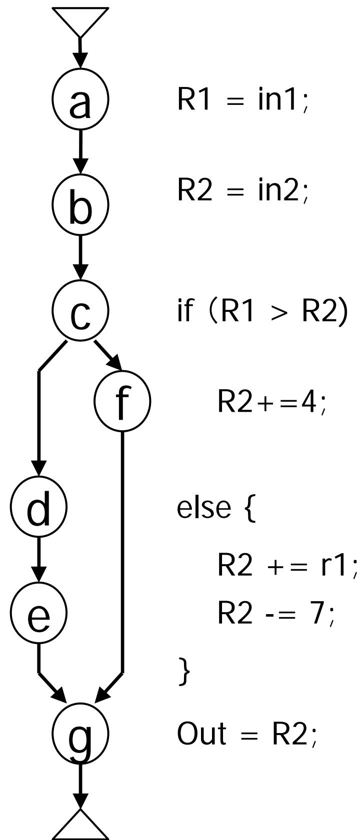
- OSCI SystemC Simulator 2.0.1 & 2.1 v1

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

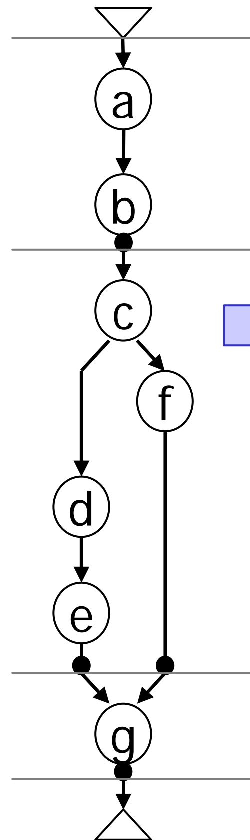
Instruction timing back-annotation

C source code




```
R1 = in1;
R2 = in2;
if (R1 > R2)
    R2+=4;
else {
    R2 += r1;
    R2 -= 7;
}
Out = R2;
```

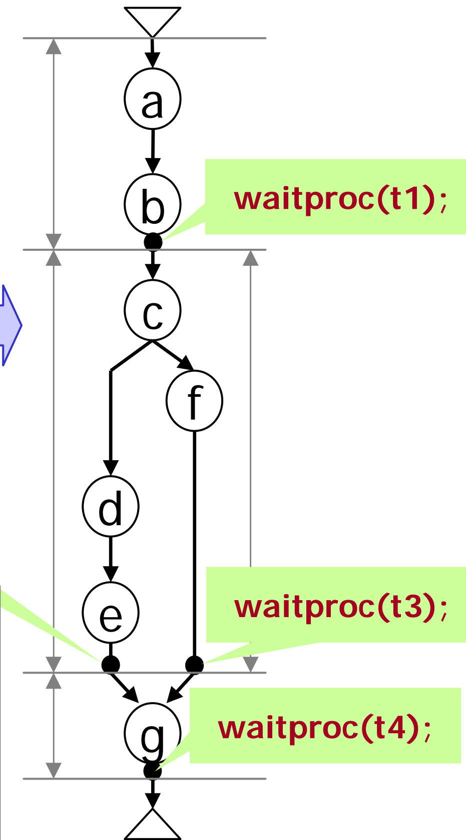
Decompose into basic blocks



Conversion

- Cross-compiling
- Examines assembly code, compute delay T in each BB.
- Embeds waitproc(T) in each basic blocks

Timed model (SystemC)



Simulation

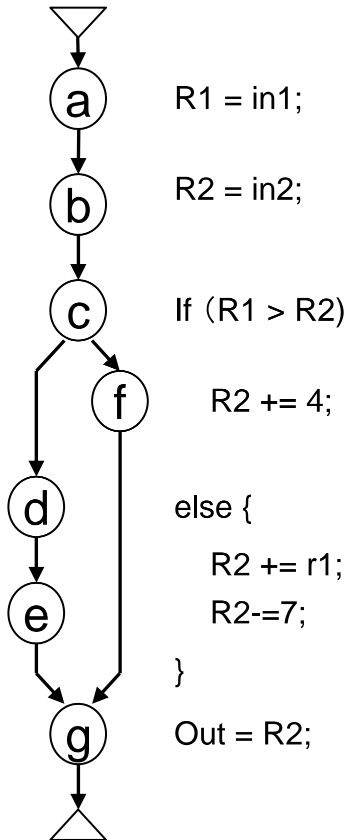
- waitproc(T) summed up instruction cycles, and eventually calls SystemC wait( ) at the designated threshold

=> reducing synchronization counts

(C) 2006, InterDesign Technologies, Inc.

Instruction cache emulation

C source code



R1 = in1;

R2 = in2;

If (R1 > R2)

R2 += 4;

else {

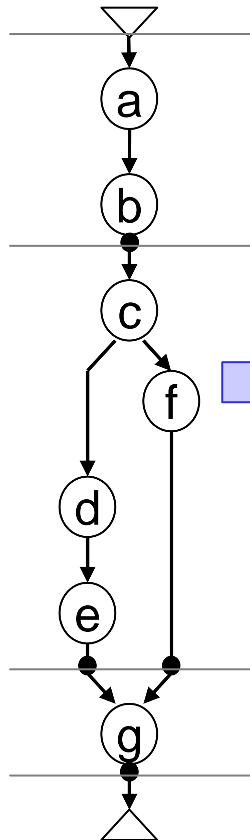
R2 += r1;

R2-=7;

}

Out = R2;

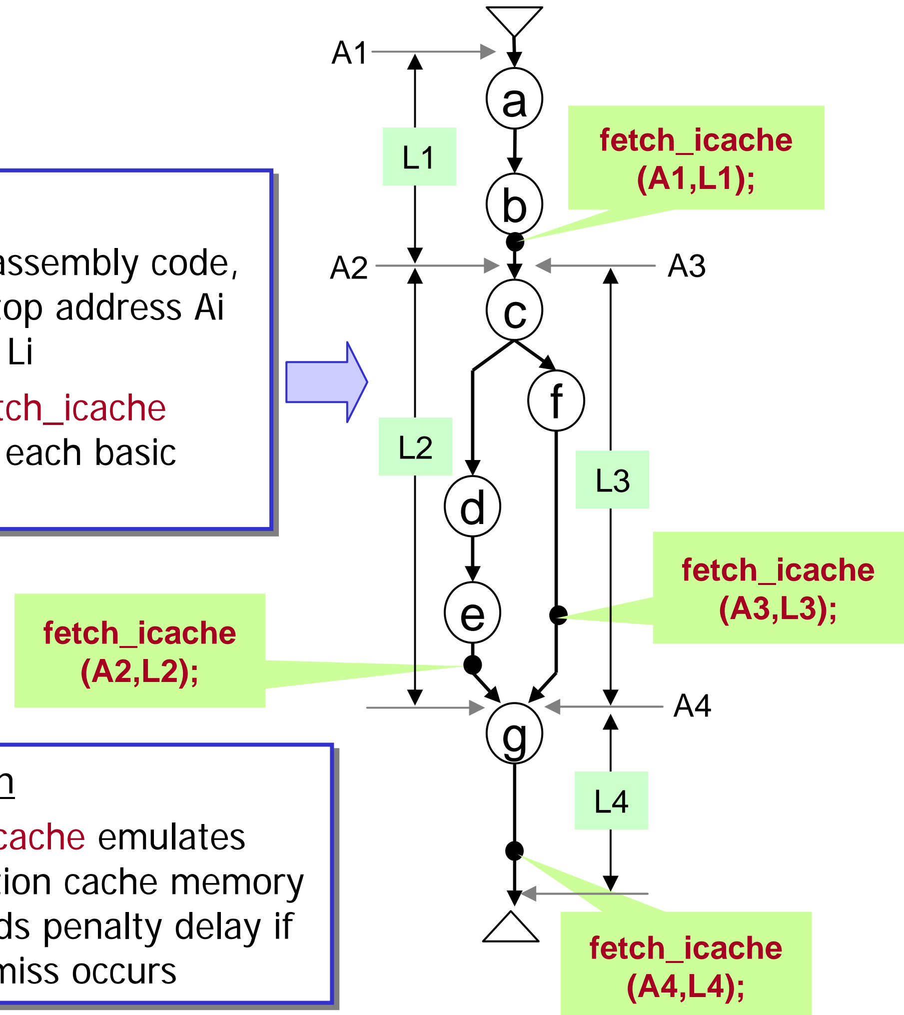
Decompose into basic blocks



Conversion

- Examines assembly code, computes top address Ai and length Li
- Embeds fetch_icache function in each basic blocks

Timed model (SystemC)



Simulation

- fetch_icache emulates instruction cache memory and adds penalty delay if cache miss occurs

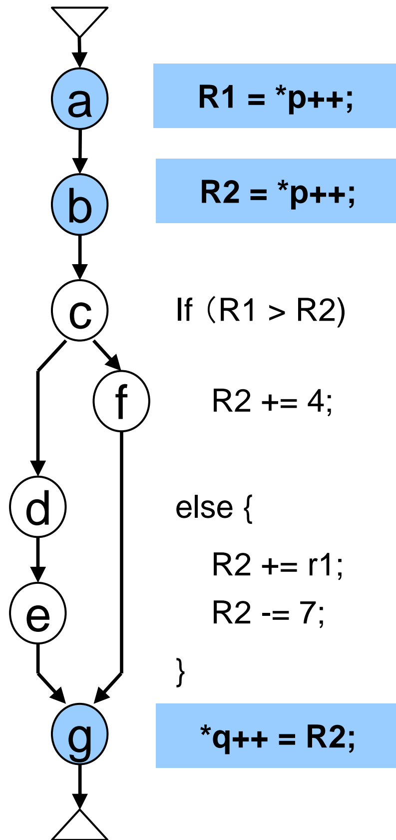
FastVeri
FAST hardware/software co-VERIfication framework

InterDesign Technologies

(C) 2006, InterDesign Technologies, Inc.

Data cache emulation

C source code



R1 = *p++;

R2 = *p++;

If (R1 > R2)

R2 += 4;

else {

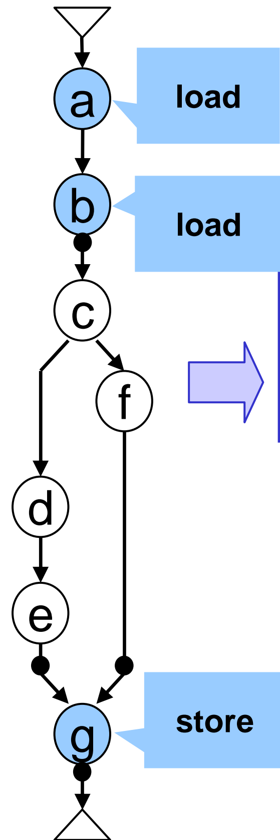
R2 += r1;

R2 -= 7;

}

*q++ = R2;

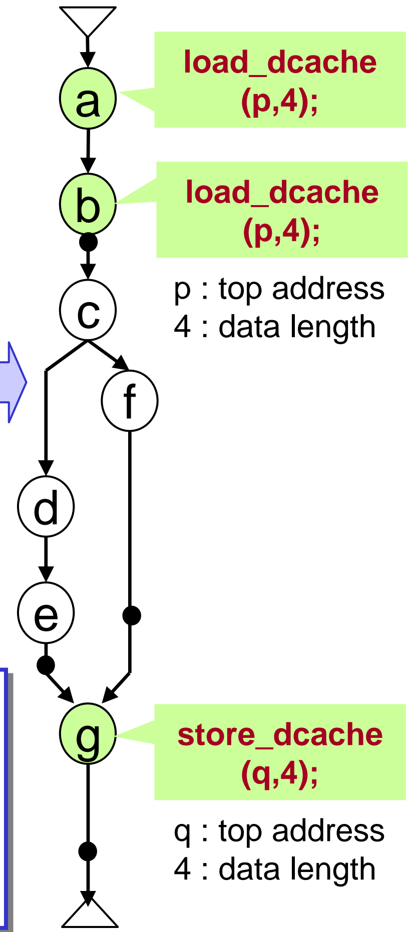
Assembly code



Conversion

- Detects load / store in assembly
- Embeds load_dcache( ) and store_dcache( ) functions

Timed model (SystemC)



Simulation

- load_dcache( ) and store_dcache( ) emulates data cache memory and add penalty delay if cache miss occurs.

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

I/O access conversion

- Detects I/O accesses in software code
  - Converts them into method calls of SystemC channel which interfaces between CPU model and external models
- C2SystemC converter supports external I/O access in embedded software as
  - I/O accessing with volatile pointer
  - I/O accessing with memory section attribute
- Care about host / target addressing

(C) 2006, InterDesign Technologies, Inc.

I/O accessing with volatile pointer

InterDesign Technologies

C

```
typedef struct {
    unsigned int Period;
    unsigned int Intclear;
} Timer_reg_t;

#define Timer ((volatile Timer_reg_t *)0x10000000)

void foo( ) {
    Timer->Period = timer_period;       // this line accesses register
}
```

C2SystemC conversion

SystemC

```
void TimedModule::foo( ) {
    io->storeUI(
        (unsigned int) &((volatile Timer_reg_t *) 0x10000000)->Period,
        this->timer_period);
}
```

storeUI( ) calls "store" method to external I/O model with target address

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

I/O accessing with memory section attribute

C

```
typedef struct {
    unsigned int Period;
    unsigned int Intclear;
} Timer_reg_t;

Timer_reg_t Timer __attribute__((section(".timer")));
// user provides target address by section map

void foo( ) {
    Timer.Period = timer_period;     // this line accesses register
}
```

C2SystemC conversion

SystemC

```
Timer_reg_t Timer;
// converter puts host and target address of "Timer" into address table

void TimedModule::foo( ) {
    io->storeHAUI((unsigned int) &this->Timer.Period, this->timer_period);
}
```

storeHAUI( ) checks 1st arg (host address) in the table.
If address is in the table, it calls "store" method to external I/O model with target address.

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

Additional notes related to I/O access conversion

"I/O accesssing with pointer" needs hint

- Converter changes "accessing to user spcified pointer" into store/load method (method checks address)
- "Static analysis of pointer access" is future work

I/O accessing of pointer typed value

- Value also needs host/target address conversion

C

```
typedef struct {
    unsigned int num;    unsigned char* dst;    unsigned char* src;
} dmac_t;

#define MEMSIZE 1024

dmac_t Dmac __attribute__((section(".dmac")));
unsigned char mem[MEMSIZE] __attribute__((section(".extmem"), aligned(0x1000)));

void foo( ) {
    Dmac.src = mem;
    Dmac.dst = &mem[MEMSIZE / 2];
}
```

storeHAP( ) converts 2nd arg (host address value) into target address

SystemC

```
io->storeHAP((unsigned int) &this->Dmac.src, this->mem);
io->storeHAP((unsigned int) &this->Dmac.dst, &this->mem[1024 / 2]);
```

FastVeri FAST hardware/software co-VERIfication framework

Performance of SystemC simulation

Design : MPEG4 decoder
Target CPU : ARM946E
Simulation host: OSCI 2.0.1, Cygwin on 3.4 GHz Pentium4

	Model A (All on software)		Model B (IDCT on HW)		Model C (M4D on HW)	
	Execution time [cycles]	Timing error rate [%]	Execution time [cycles]	Error rate [%]	Execution time [cycles]	Error rate [%]
Evaluation Board	110,998,126	-	98,747,078	-	69,154,754	-
AVT model *a)	110,710,334	-0.26	103,007,752	4.31	66,954,693	-3.18
PVT model *b)			102,048,426	3.34	67,508,298	-2.38

IDCT : Inverse Discrete Cosine Transform
M4D : MPEG4 Decoder core

		Simulation speed [million cycles per second]		
		Model A	Model B	Model C
TCM *1)	AVT model *a)	259.7	247.4	210.3
	PVT model *b)		198.9	120.3
	ratio	-	1.24	1.74
Cache *2)	AVT model *a)	34.6	36.6	56.4
	PVT model *b)		35.1	45.0
	ratio	-	1.04	1.26

HW modeling style :
*a) AVT (Architects View + Timing)
*b) PVT (Programmers View + Timing)

CPU configuration :
*1) TCM : Uses TCM (Tightly Coupled Memory)
*2) Cache : Uses 8KB inst. cache, 8KB data cache

* These experimental data is reported by STARC

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

Comparison with RTL simulation

InterDesign Technologies

Design : MPEG4 decoder
Target CPU : ARM946E

		Host environment	Simulation time	Speed ratio
Verilog RTL	RTL Verilog model of ARM946E core and M4D HW	HDL simulator on Solaris on 1.2 GHz UltraSPARC-III	201,600 sec.	1
SystemC	Model C : ARM946E core (by FastVeri) and M4D PVT model	OSCI 2.0.1, Cygwin on 3.4 GHz Pentium4	1.5 sec.	130,000

M4D : MPEG4 Decoder core

CPU configuration :
Uses 8KB inst. cache, 8KB data cache

HW modeling style :
PVT (Programmers View with Timing)

* These experimental data is reported by STARC

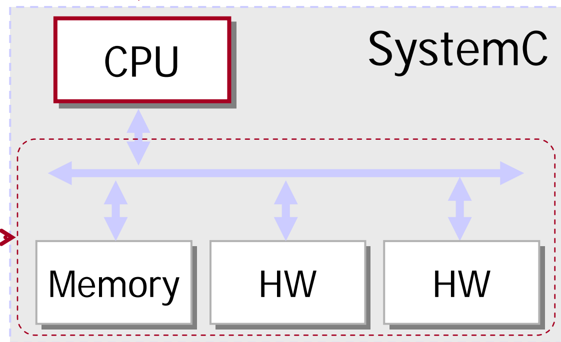
FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

FastVeri tool chain

Software source code (ANSI C)

C2SystemC converter



Using OSCI or 3rd vender's SystemC simulator

HW/SW co-verification

Simulation

Simultaneous code debugging of software source code in C and hardware model in SystemC

Performance analysis

Performance analyzer

Profiling and analyzing software execution on target CPU core

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.

Conclusion

InterDesign Technologies

- Virtual CPU model with timing back-annotated SW code
  - Drastically speeding up HW/SW co-simulation
  - Achieves cycle approximate accuracy
- Typical usage
  - Design space exploration
  - Verify HW/SW design with virtual platform simulation

FastVeri
FAST hardware/software co-VERIfication framework

(C) 2006, InterDesign Technologies, Inc.