임베디드 시스템의 성능 및 오류 검증

YUNHEUNG PAEK
SOC OPTIMIZATIONS AND RESTRUCTURING RESEARCH GROUP
SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
SEOUL NATIONAL UNIVERSITY

2009

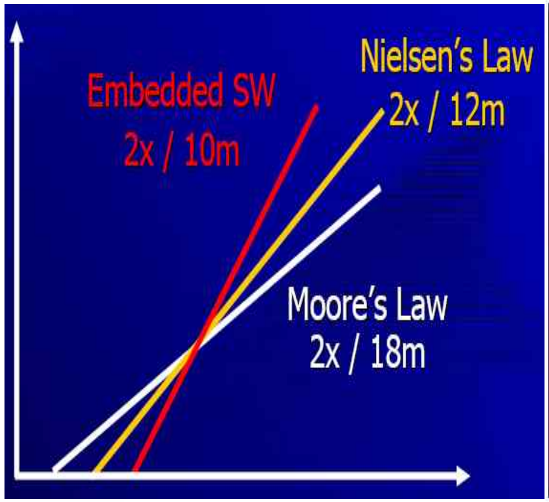
교육과학기술부
국가지정연구실
National Research Lab

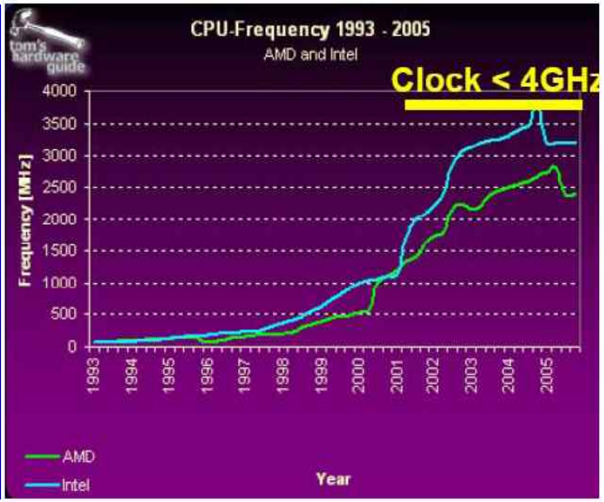
Contents

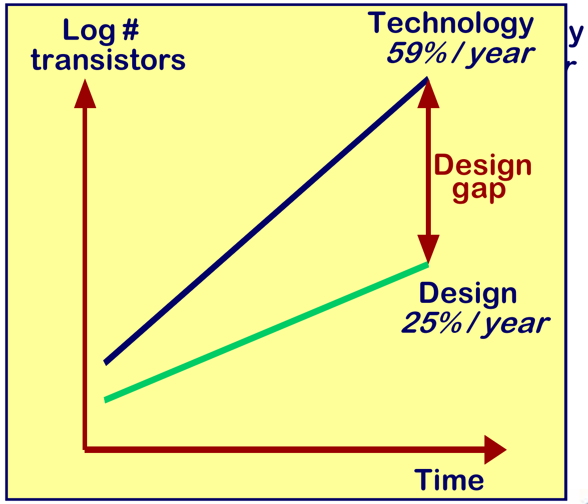
MPSoC ERA

- 응용프로그램의 복잡도 증가 vs. H/W 성능 증가
  - 단일 CPU만으로 고성능 응용프로그램의 수행 제한적
- 집적도 한계와 기하급수적 전력소모 증가
  - 단일 CPU 성능 향상 한계 도달
- CPU대안인 ASIC설계기술 발전 속도
  - 집적도 향상으로 NRE 비용 상승 & Time-to-market 저하

해결방안 ➔ MPSoC







Parallel Programming as a hot issue

- From PLDI `09 accepted paper list

PLDI 2009 Home Page - Mozilla Firefox

파일(F) 편집(E) 보기(V) 바로 가기(S) 북마크(B) 도구(T) 도움말(H)

http://www-plan.cs.colorado.edu/~pldi09/

자주 찾은 사이트 Firefox 시작 하기 최신 뉴스 보기

Home

Mon, June 15
- Tutorials
- PLAS workshop
- SRC Reception

Tue-Thu, June 16-18
- PLDI Technical Program (3 full days!)
- FIT Session (Tue)
- PLDI'09 Outing, O'Callaghan Davenport Hotel (Tue)
- SRC Presentations (Thu)

Fri-Sat, June 19-20
- LCTES
- ISMM

Registration
Accommodations
Local Orientation
Tourism Information

by Ohad Shacham (Tel Aviv University), Martin Vechev (IBM Research), and Eran Yahav (IBM Research)

- *PetaBricks: A Language and Compiler for Algorithmic Choice*
  by Jason Ansel, Cy Chan, Yee Lok Wong, Qin Zhao, Alan Edelman and Saman Amarasinghe, MIT
- *GC Assertions: Using the Garbage Collector To Check Heap Properties*
  by Edward Aftandilian and Samuel Guyer, Tufts University
- *Verifiable Composition of Deterministic Grammars*
  by August Schwerdfeger and Eric Van Wyk, University of Minnesota
- *FastTrack: Efficient and Precise Dynamic Race Detection*
  by Cormac Flanagan (University of California, Santa Cruz) and Stephen Freund (Williams College)
- *Effective Sampling for Lightweight Data-Race Detection*
  by Daniel Marino (UCLA), Madanlal Musuvathi (Microsoft Research), and Satish Narayanasamy (University of Michigan, Ann Arbor)
- *Semantics-Aware Trace Analysis*
  by Kevin Hoffman, Patrick Eugster, and Suresh Jagannathan, Purdue University
- *Lightweight Annotations for Controlling Sharing in Concurrent Data Structures*
  by Zachary Anderson (University of California, Berkeley), David Gay (Intel Research, Berkeley), and Mayur Naik (Intel Research, Berkeley)

5

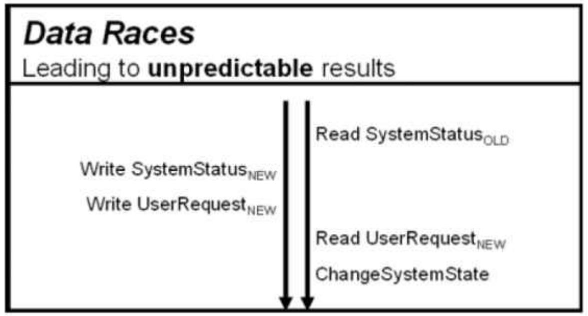
멀티 프로세서 디버깅

6

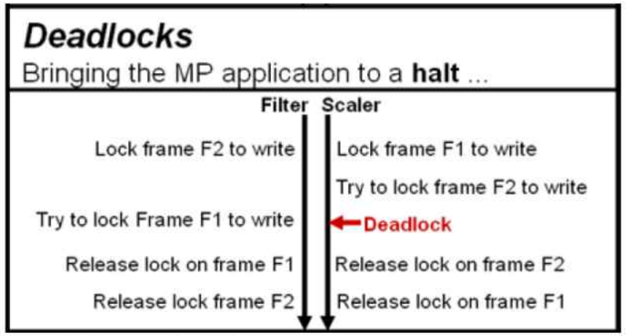
Necessity of an MPSoC source level debugger

MPSoC nightmare

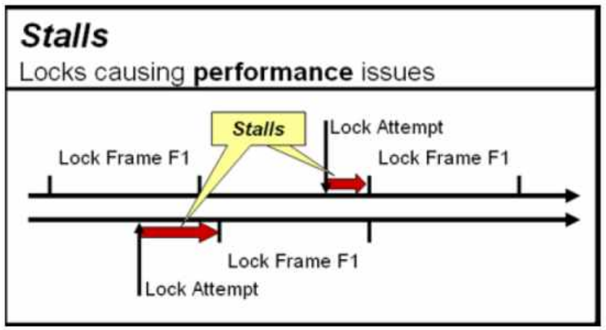
- MPSoC nightmares – runtime errors are important



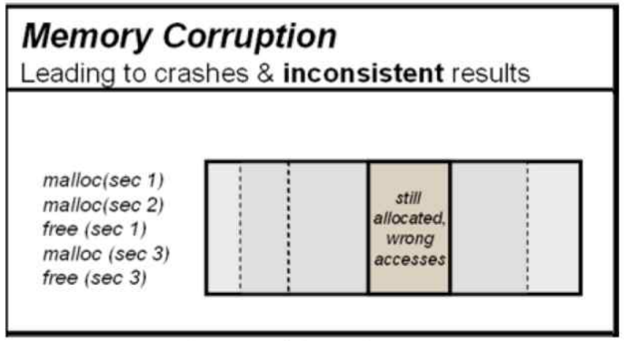
3: Data races are an example of MP debug nightmares.



4: Deadlocks are another MP debug nightmare.



6: Engineers may have a MP debug nightmare when dealing with stalls.



5: Memory corruption also is a MP debug nightmare.

Source Level Debugger

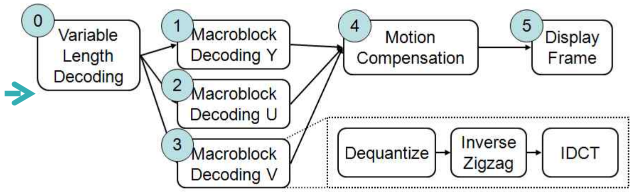
- Programmer의 semantic과 저수준의 error를 연결

```
SomeObject  *objPtr = NULL;
if ( a certain condition )
   objPtr = new Object;
else
  ;  // in this case, NULL pointer

objPtr -> do_some_job();

...
...
```

```
(gdb) print objPtr
0
```

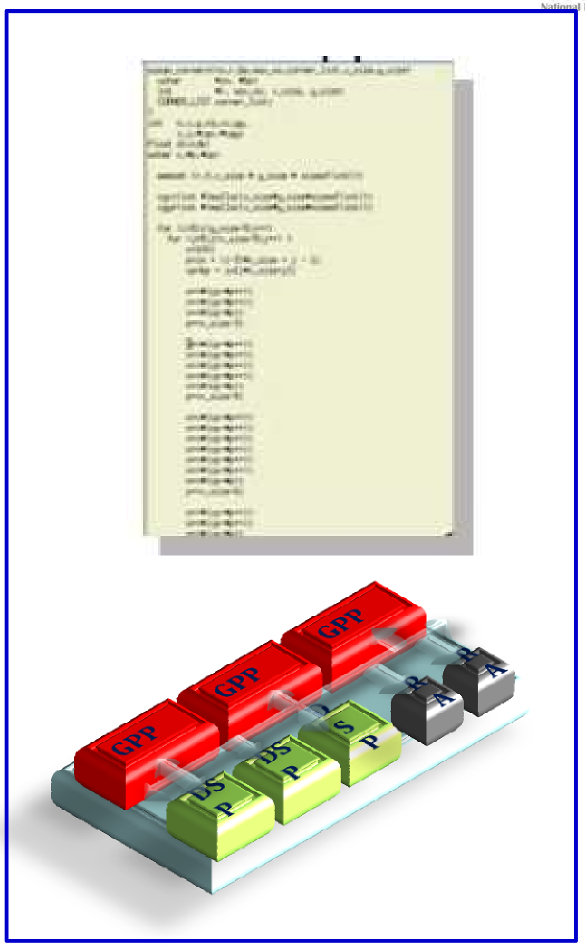
Even if…

- Tools are not so perfect



We need a
Source level debugger
for MPSoC

CAD tool

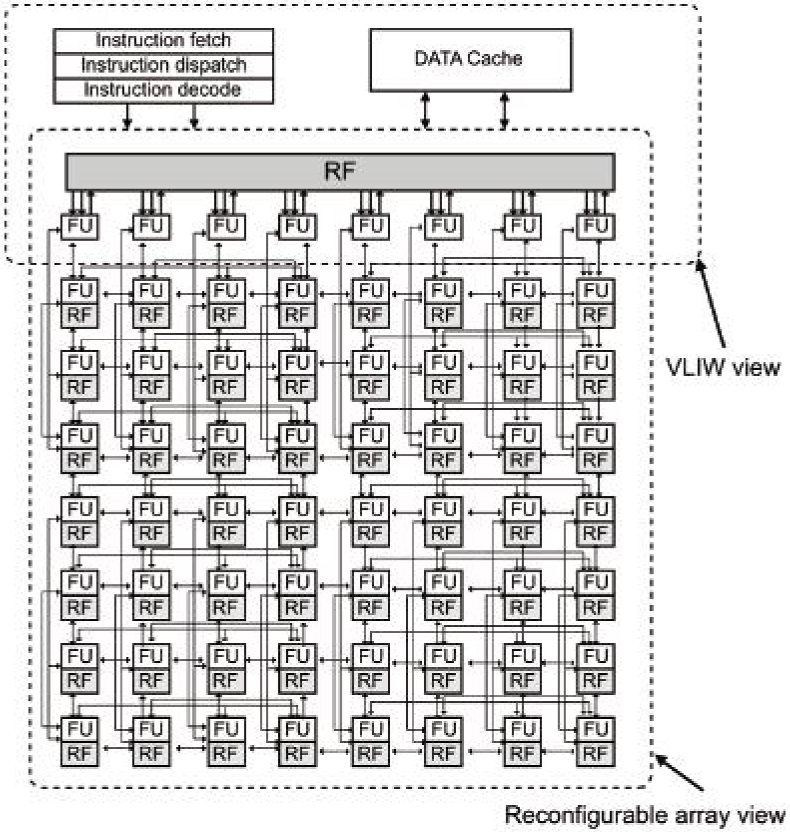


Automatically Synthesized code

What we have done...

Development of a re-targetable source level debugger for ADRES, with S.A.I.T

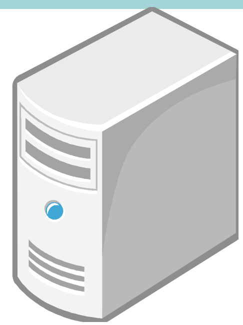
Reconfigurable Processor(ADRES)



개요 : Re-targetable Debugger

Independent part of a debugger :

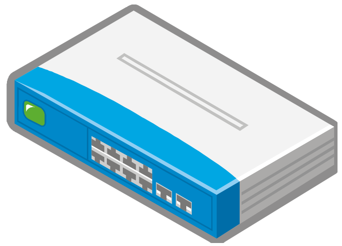
Source Level Symbol Table,
Logical breakpoint manager,
Target abstraction layer, …



Target Dependent codes for x86 Windows:

….

….

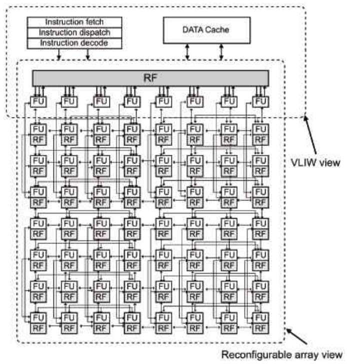


Target Dependent codes for MIPS :

….

….

….



Target Dependent codes for Adres :

….

….

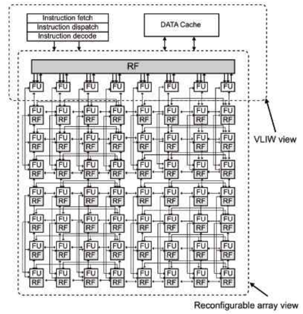
….

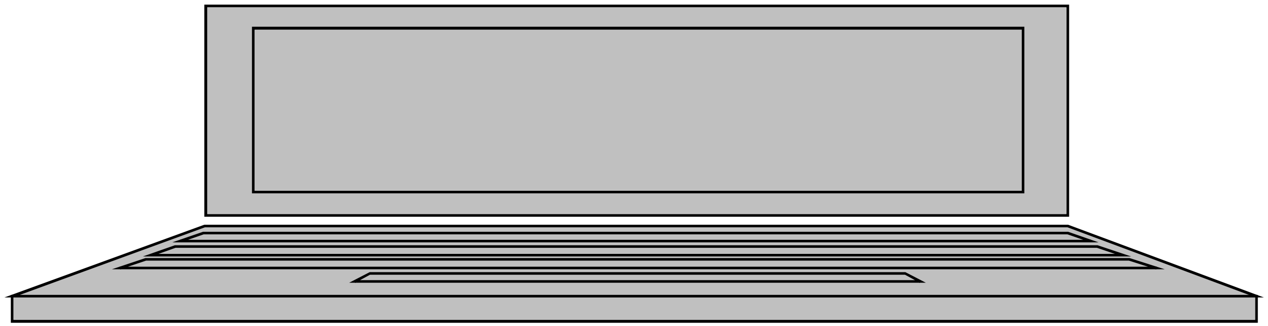
전체 그림 - Debugger 구조

Frontend : Visual Studio 2008

Debugger Engine : SNU 구현

PSIM : processor simulator





Personal Computer, x86 windows

Data inspection

- Data inspection
  - Operands
    - Scope
      - Static global, local variables
      - External global variables
      - Auto local variables
      - Function parameters
      - Nested scope supported
    - Data type
      - Arrays, struct variables, pointers
      - Union variables not yet.

Data inspection – cont'd

- Operation
  - Almost every operations but function call
    - Arrow operator (->) and assign operator(=) included
    - Array index operator([]), dot operator included
    - Dereference operator(*), Address operator(&) included
    - Basic binary, unary operators
      - +, -, <<, >>, /, %, etc included
  - Long and complex C expression supported
    - Ex) N[3] -> next . array_field[2] . second_field

Control execution

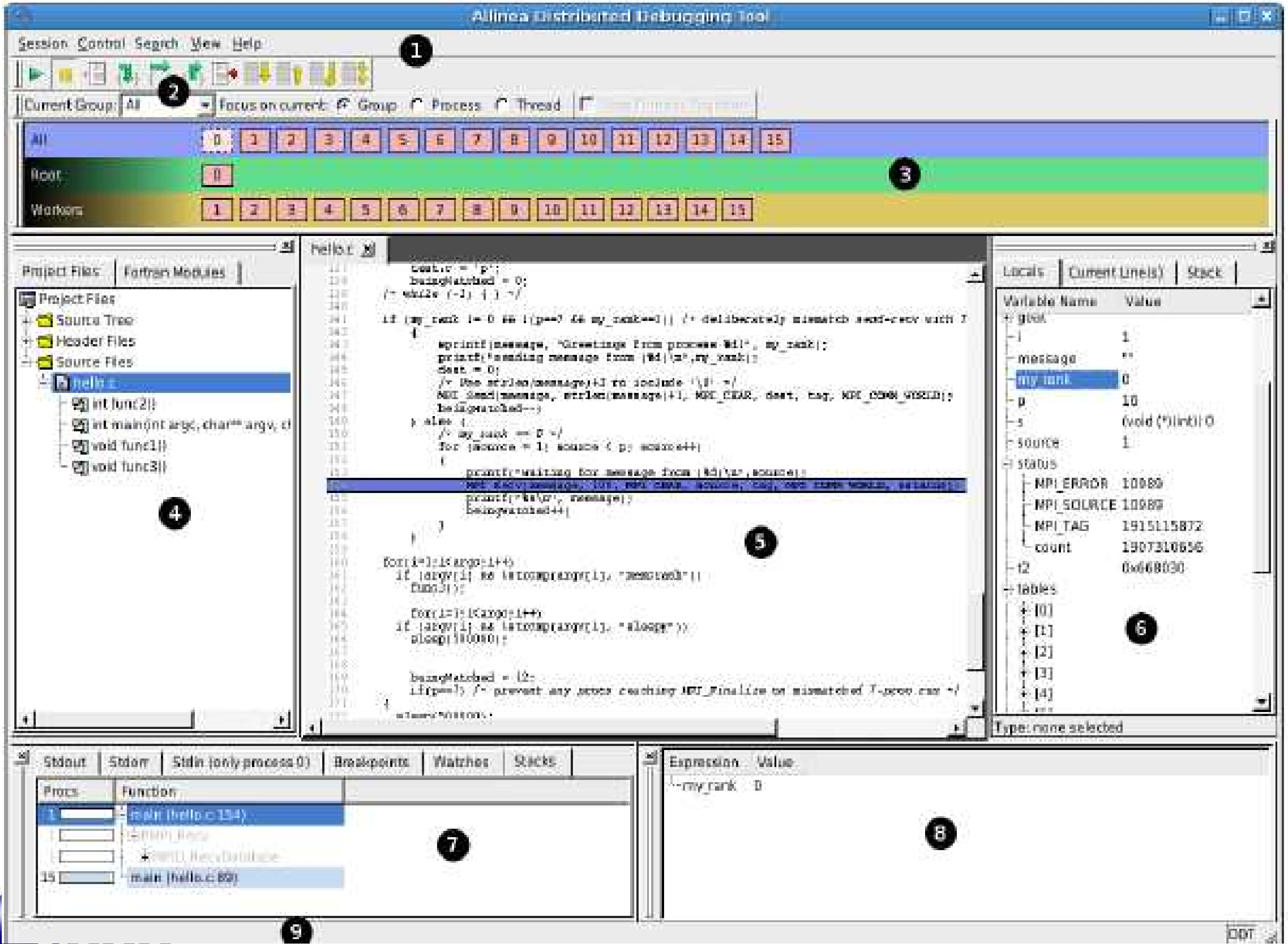
- Control execution
  - Removing/Setting breakpoints
  - Step in/over/out
  - Run, continue execution

17

연구 주제 확장

MPSoC debugger, Runtime error detection

Multi-process debugger example - DDT



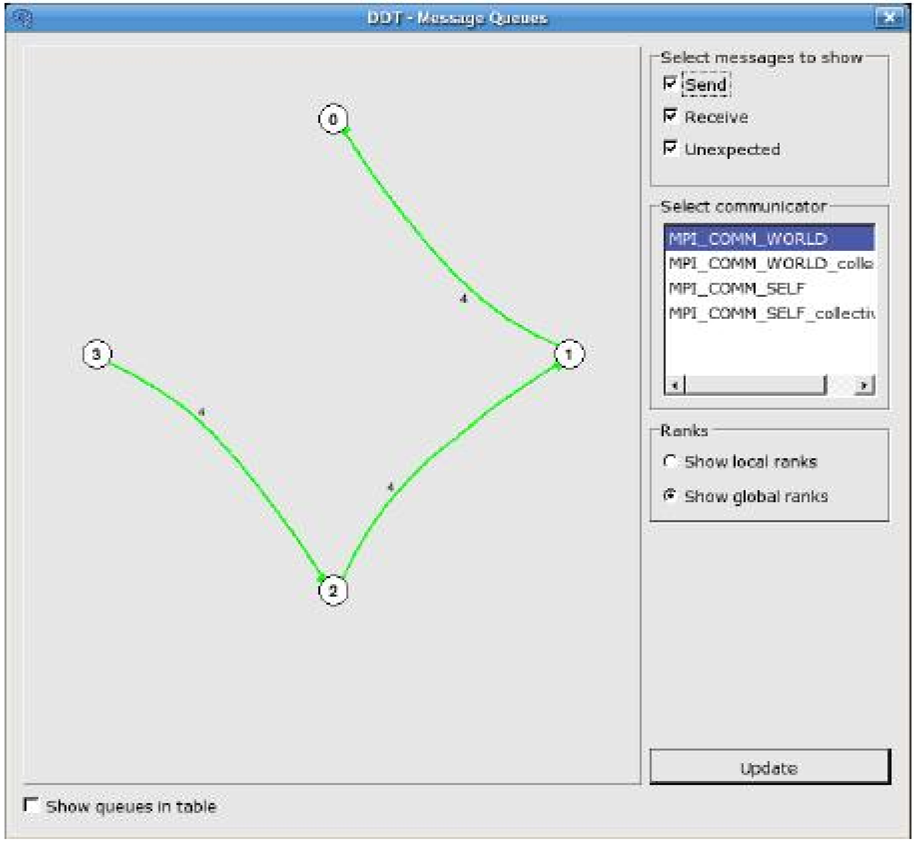
- 1. menu bar
- 2. process controls
- 3. process groups
- 4. project navigator
- 5. source code
- 6. variables and stack
- 7. parallel stack, IO, Breakpoints
- 8. evaluate window
- 9 status bar

Conditional breakpoints for multiple processes

Stdout | Stderr | Stdin (only process 0) | Breakpoints | Watches | Stacks

	Processes	Threads	File	Line	Function	Condition	Full path
☑	All	all	hello.c	91			/var/nick/code/ddt/examples
☑	All	all		0	malloc		
☑	All	all	hello.c	105		x > 0	/var/nick/code/ddt/examples
☑	Workers	all	hello.c	110			/var/nick/code/ddt/examples

DDT – Message Queues



Basically

- MPSoC debugger looks like DDT

- Including runtime error detection modules such as
  - Race condition detection
  - Data race detection
  - Deadlock detection
  - etc

22

임베디드 소프트웨어/하드웨어 검증

Problems

- Use different languages for modeling and implementation
  - Cannot verify the desired functions directly
- Hardware designers have to restart the design process by capturing the designs using the HDLs
  - May have unmatched problems
- Require many experts in system architecture for the partition of software and hardware parts
  - The partition may not be the optimal solution
- Hardware and software integration is often painful
  - Hardware and software cannot work together
  - Co-verification of hardware and software is inefficient

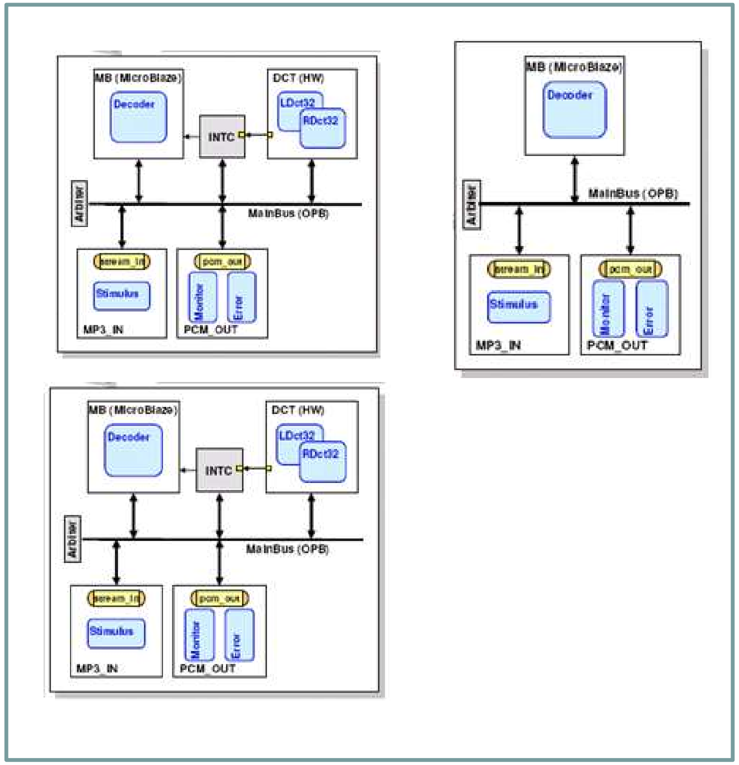
- Long design time and high-cost
- Verification and Debugging is painful

Needs of New Methodology

- [1] Kurt Keutzer, et. al. "System-Level Design: Orthogonalization of Concerns and Platform-Based Design," IEEE TCAD, 19(12), December 2000.

"we believe that the lack of appropriate methodology and tool support for modeling of concurrency in its various forms is an essential limiting factor in the use of both RTL and commonly used programming languages to express design complexity"

Meet-in-the-middle methodology



Set of architectures(platform)

mapping

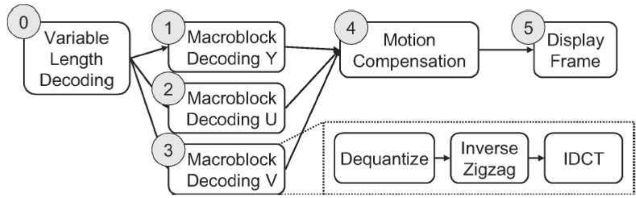
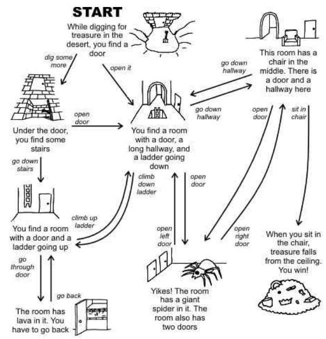


Fig. 3. Task specification example: H.263 decoder.

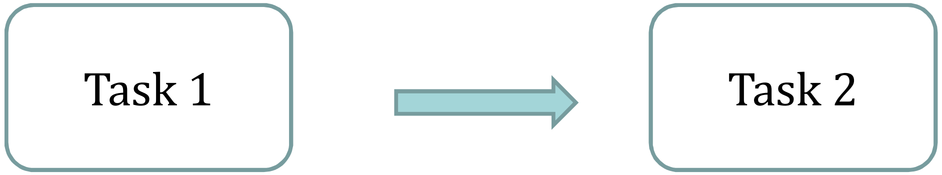


Implementation-independent application specifications

Timing in real-time systems

- Specify-Exploration-Refinement design methodology
  - Verification is very important
  - Timing is key factor for verification between specification before refinement and that after refinemen
- With SER methodology, high level estimation of execution time is important
  - To improve performance
  - To reduce runtime errors related with timing

Wrong estimation

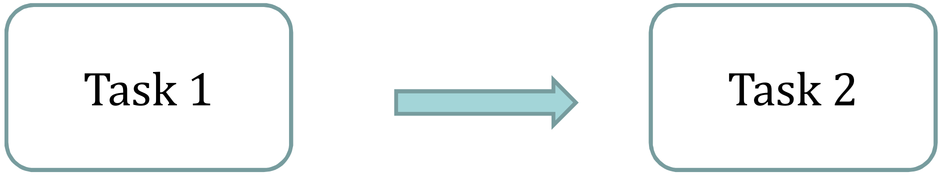


	Estimation	Actual execution time
Task 1	100	50
Task 2	50	50

Deadline = 120

Since the estimation was wrong, we need one more processor to meet the deadline

Wrong estimation



	Estimation	Actual execution time
Task 1	50	100
Task 2	50	50

Deadline = 120

Since the estimation was wrong, verification fails or unexpected runtime error occurs

Co-work with Aachen

- They developed high level estimation scheme and publish the paper, "Multiprocessor performance estimation using hy brid simulation", Lei Gao, et al
  - http://portal.acm.org/citation.cfm?id=1391469.1391552

- We have the environment that we can compile source code into ISA of a certain virtual machine, simulate and profile
  - By comparing these two method, and complementing, we expect getting more precise high level estimation method

30

결론

멀티 프로세서 디버깅

- MPSoC programming
  - It becomes more important to detect or avoid runtime errors
  - Connecting low level error and user-level semantic is getting crucial
  - Necessity of a source level debugger increases drastically

임베디드 소프트웨어/하드웨어 검증

- High level performance estimation is the key for
  - Design space exploration
  - Improve performance of the synthesized system
  - Avoiding runtime errors, especially related to timing in real-time systems