A Comparison of the RTU Hardware RTOS with a Hardware/Software RTOS

Jaehwan Lee[+], Vincent J. Mooney III[++]
{jaehwan, mooney}@ece.gatech.edu, http://codesign.ece.gatech.edu
[++]Assistant Professor, [+]School of Electrical and Computer Engineering
[++]Adjunct Assistant Professor, College of Computing
Georgia Institute of Technology, Atlanta, GA, U.S.A.

Anders Daleby*, Karl Ingström*,
Tommy Klevin** and Lennart Lindh**
{ady99002, kim99001, klevin, llh}@mdh.se
*Mälardalens University, Västerås, Sweden
**Mälardalens University and RealFast, Västerås, Sweden

Outline

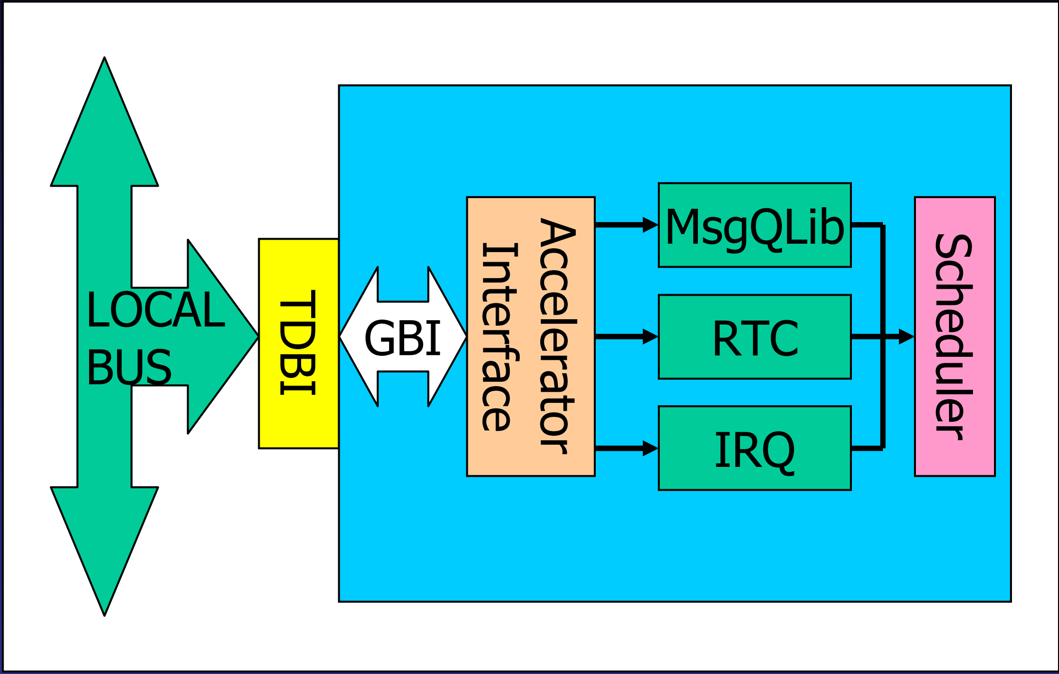
Introduction

- Trends in chip design
  - multiple processor cores in system-on-a-chip (SoC)
  - functionality moving from hardware to software
- Trends in RTOS design
  - traditional software RTOS
  - hardware/software RTOS
  - hardware RTOS
- Automatic configuration
  - for multiprocessor system-on-a-chip
  - to support current trends
  - to compare and contrast benefits of different RTOS options

Goal

- To help the user examine different system-on-a-chip (SoC) architectures utilizing a hardware and/or software RTOS

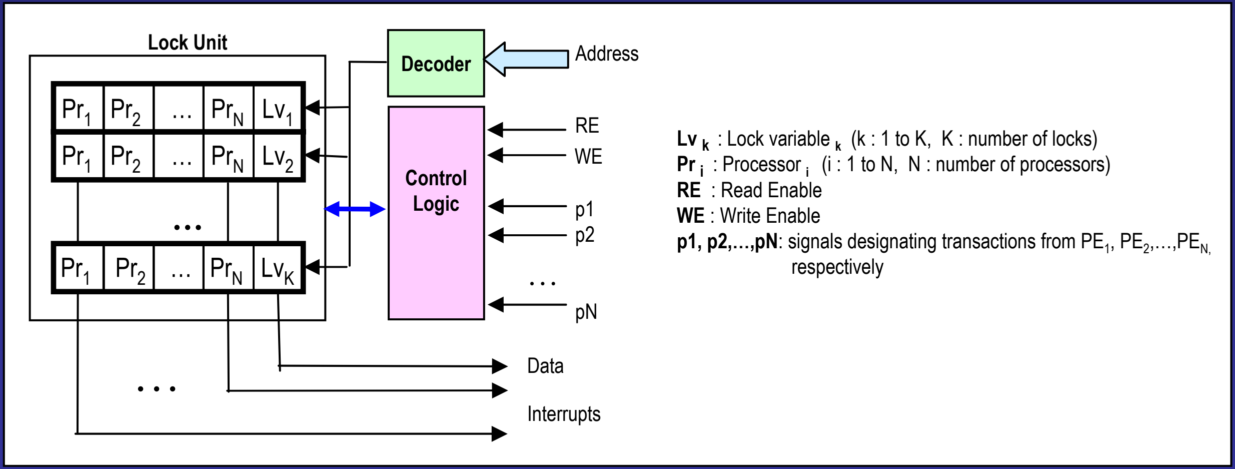
Motivation (1/5)



- An RTOS in hardware
  - Real-Time Unit (RTU)
    - scheduling
    - IPC
    - dynamic task creation
    - timers
  - Custom hw => upper bound on # tasks
  - Reconfigurable hw => can alter max. # tasks, max. # priorities
  - Prof. Lennart Lindh, Mälardalens U., Västerås, Sweden
  - RealFast, www.realfast.se

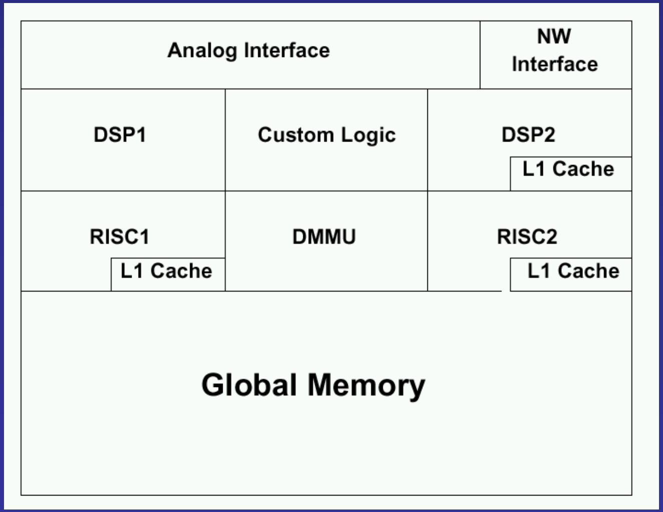
Motivation (2/5)

- System-on-a-Chip Lock Cache
  - A hardware mechanism that resolves the critical section (CS) interactions among PEs
  - Lock variables are moved into a separate "lock cache" outside of the memory
  - Improving the performance criteria in terms of lock latency, lock delay and bandwidth consumption


Lock Unit
Pr_1 Pr_2 … Pr_N Lv_1
Pr_1 Pr_2 … Pr_N Lv_2
…
Pr_1 Pr_2 … Pr_N Lv_K
Decoder
Address
Control Logic
RE
WE
p1
p2
…
pN
Data
Interrupts
Lv_k : Lock variable $_k$ (k : 1 to K, K : number of locks)
Pr_i : Processor $_i$ (i : 1 to N, N : number of processors)
RE : Read Enable
WE : Write Enable
p1, p2,…,pN: signals designating transactions from PE_1, PE_2,…,PE_N, respectively

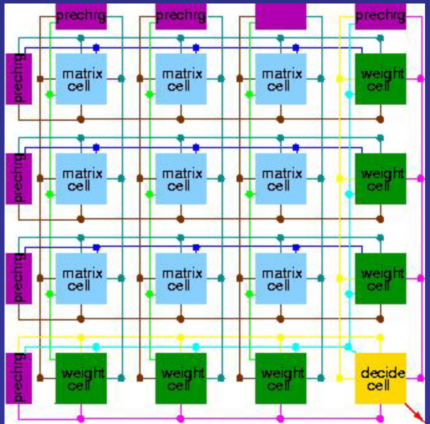
Motivation (3/5)

- SoCDMMU: System-on-a-Chip Dynamic Memory Management Unit
  - Provides fast, deterministic and yet dynamic memory management of a global on-chip memory
  - Achieves flexible, efficient memory utilization
  - Provides APIs for applications



Motivation (4/5)

- SoCDDU: System-on-a-Chip Deadlock Detection Unit
  - Performs a novel parallel hardware deadlock detection based on implementing deadlock searches on the resource allocation matrix in hardware
  - Provides a very fast deadlock detection at run-time with dedicated hardware performing simple bit-wise boolean operations
  - Reduces deadlock detection time by 99% as compared to software
  - Requires at most O(2*min(m,n)) iterations as opposed to O(m*n) required by all previously reported (sequential) software algorithms

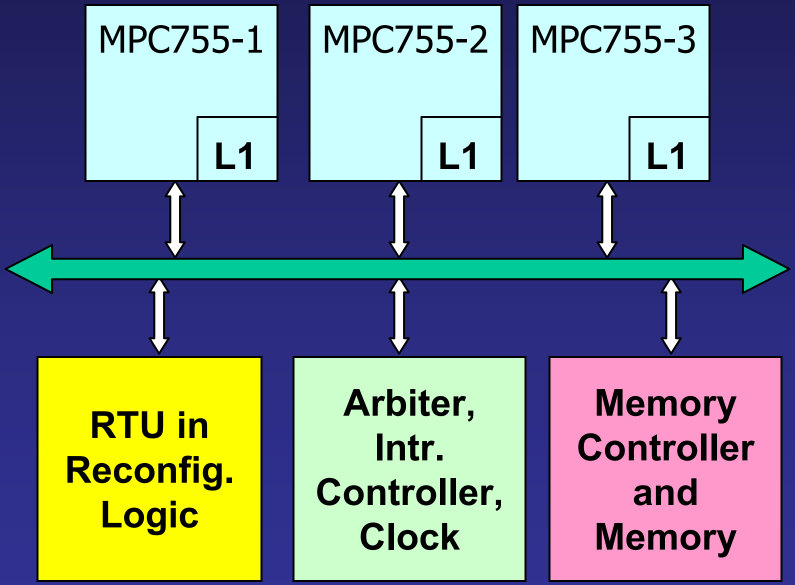


Motivation (5/5)

- δ framework
  - Enables automatic generation of different mixes of the HW/SW RTOS
  - Can be generalized to instantiate additional HW or SW RTOS components
- Configuration and Comparison
  - RTU Hardware RTOS
  - A HW/SW RTOS

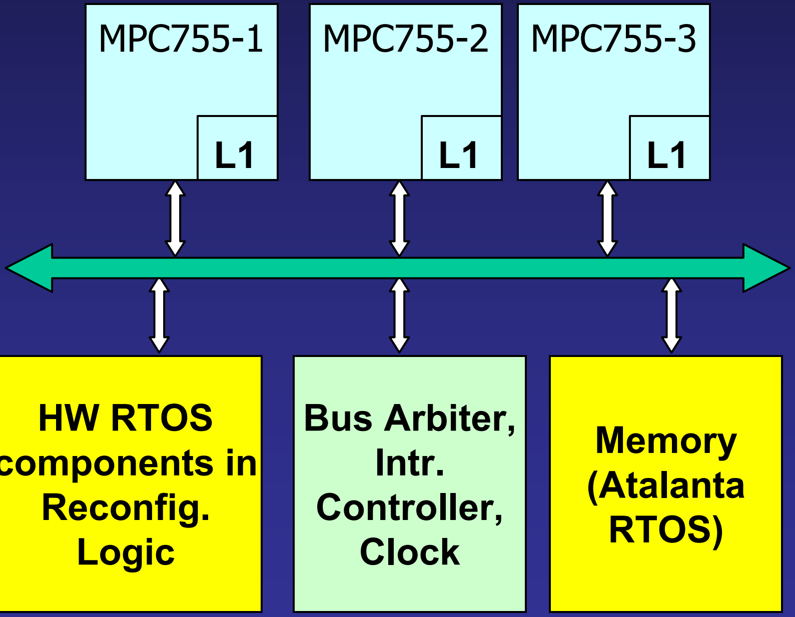
Methodology

- An SoC architecture with the RTU Hardware RTOS


MPC755-1
L1
MPC755-2
L1
MPC755-3
L1
RTU in Reconfig. Logic
Arbiter, Intr. Controller, Clock
Memory Controller and Memory

Methodology

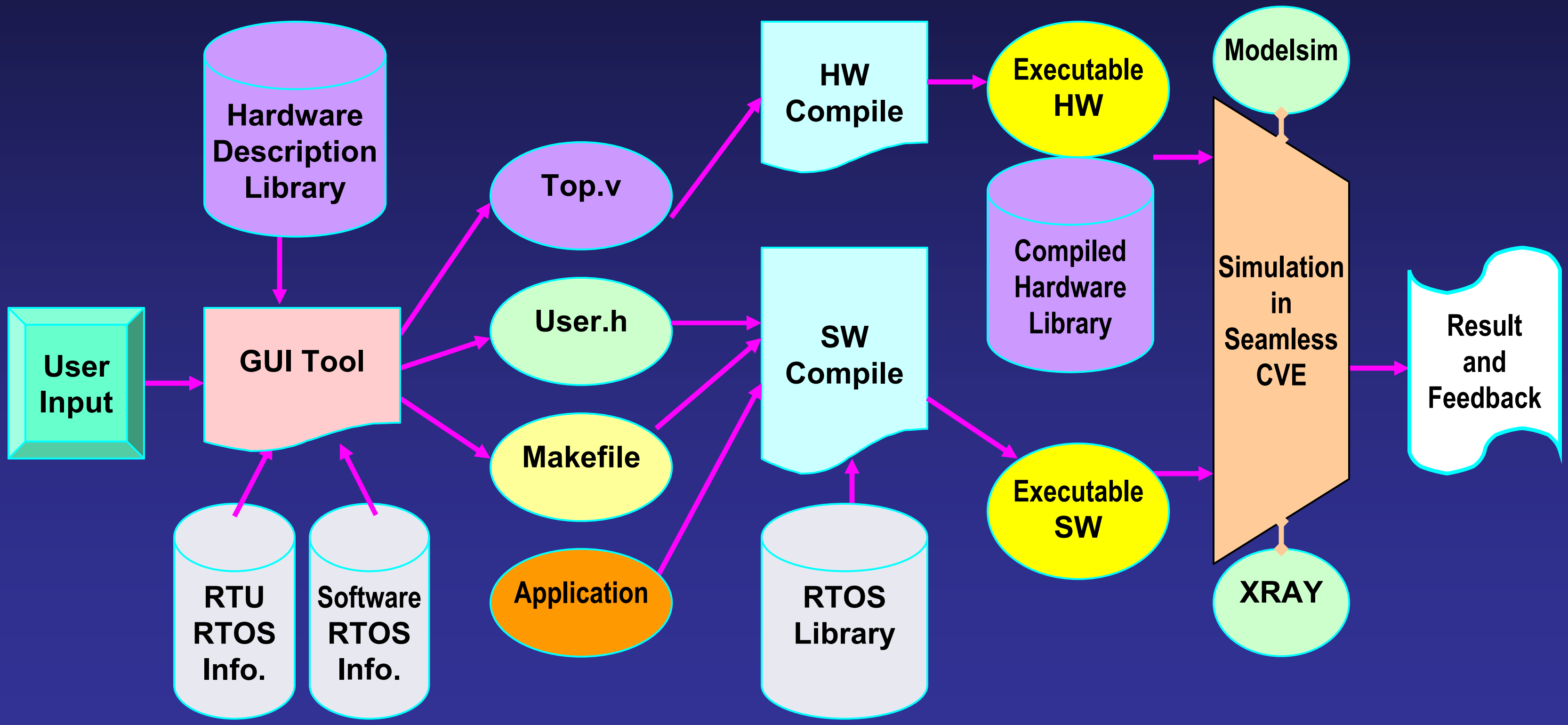
- An SoC architecture with a hardware/software RTOS



MPC755-1
L1
MPC755-2
L1
MPC755-3
L1
HW RTOS components in Reconfig. Logic
Bus Arbiter, Intr. Controller, Clock
Memory (Atalanta RTOS)

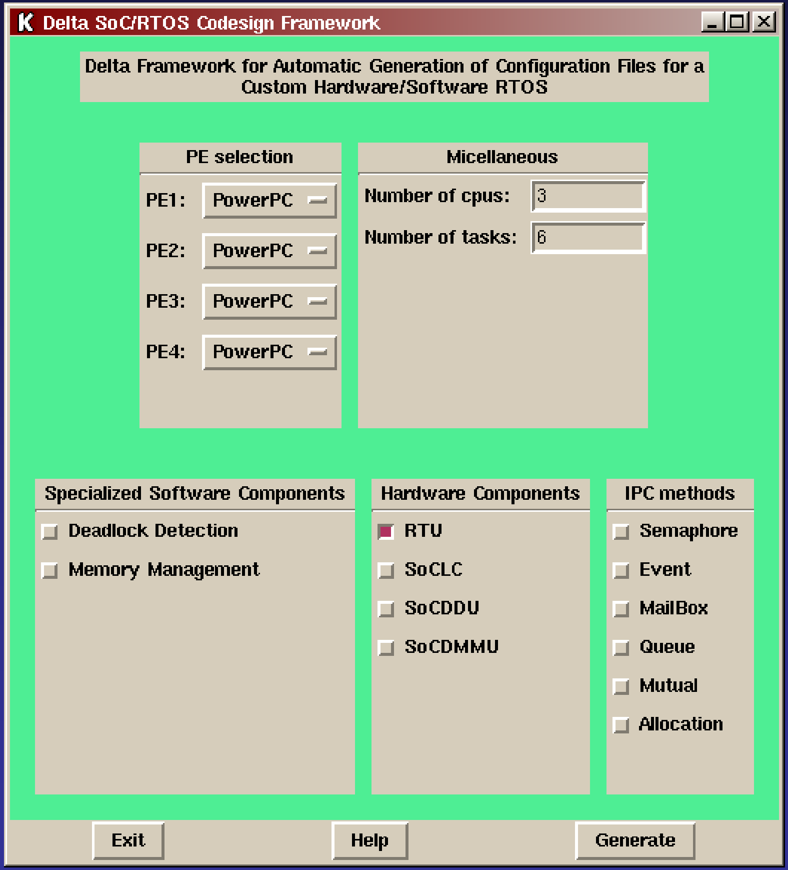
Methodology

- δ Framework



Methodology

- δ Framework – GUI



Delta SoC/RTOS Codesign Framework

Delta Framework for Automatic Generation of Configuration Files for a Custom Hardware/Software RTOS

PE selection

PE1:	PowerPC
PE2:	PowerPC
PE3:	PowerPC
PE4:	PowerPC

Micellaneous

Number of cpus:	3
Number of tasks:	6

Specialized Software Components

- ☐ Deadlock Detection
- ☐ Memory Management

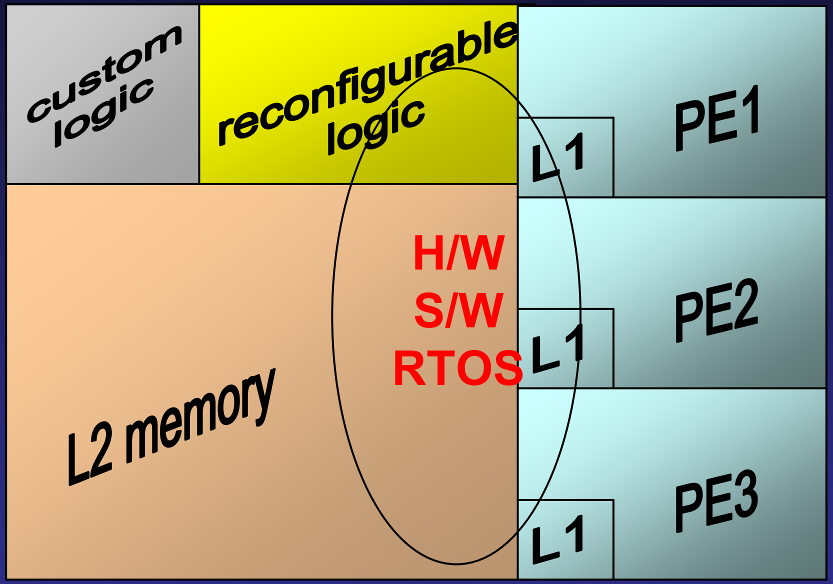
Hardware Components

- ☑ RTU
- ☐ SoCLC
- ☐ SoCDDU
- ☐ SoCDMMU

IPC methods

- ☐ Semaphore
- ☐ Event
- ☐ MailBox
- ☐ Queue
- ☐ Mutual
- ☐ Allocation

Our RTOS and Possible Target SoC



- A multiprocessor System-on-a-Chip (*Base* architecture)
- A multiprocessor RTOS
- Application(s) running on the SoC using the RTOS APIs

Our RTOS in Detail

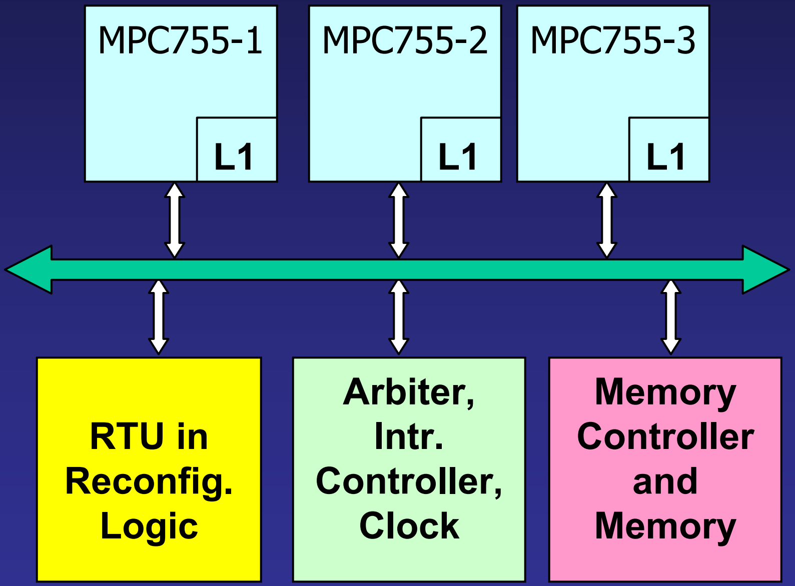
- Atalanta software RTOS
  - A multiprocessor SoC RTOS
    - the RTOS and device drivers are loaded into the L2 cache memory
  - All Processing Elements (PEs)
    - share the kernel code and data structures

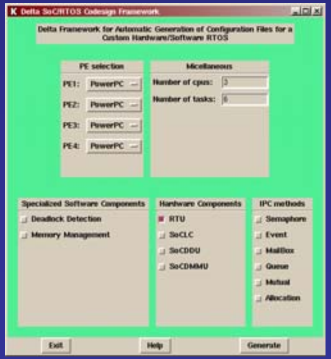
Selectable RTOS IP components

- Software (Atalanta RTOS)
  - Inter-Process Communication (IPC) components (semaphore, queue, event, mailbox, etc)
  - Memory management module (gmm)
  - Deadlock detection module (ddm)
- Hardware
  - RTU Hardware RTOS
  - SoC Lock Cache for fast IPC (SoCLC)
  - Dynamic Memory Management Unit (SoCDMMU)
  - Deadlock Detection Unit (SoCDDU)

Implementation

- RTU HW RTOS component integration method
  - Integrate user-selected HW RTOS components into the Base architecture
  - RTU selection from GUI
  - Start with architecture description
  - Generate a Verilog top file


MPC755-1
L1
MPC755-2
L1
MPC755-3
L1
RTU in Reconfig. Logic
Arbiter, Intr. Controller, Clock
Memory Controller and Memory


Delta SoC/RTOS Codesign Framework

Delta Framework for Automatic Generation of Configuration Files for a Custom Hardware/Software RTOS

PE selection

PE1: PowerPC

PE2: PowerPC

PE3: PowerPC

PE4: PowerPC

Micellaneous

Number of cpus: 3

Number of tasks: 6

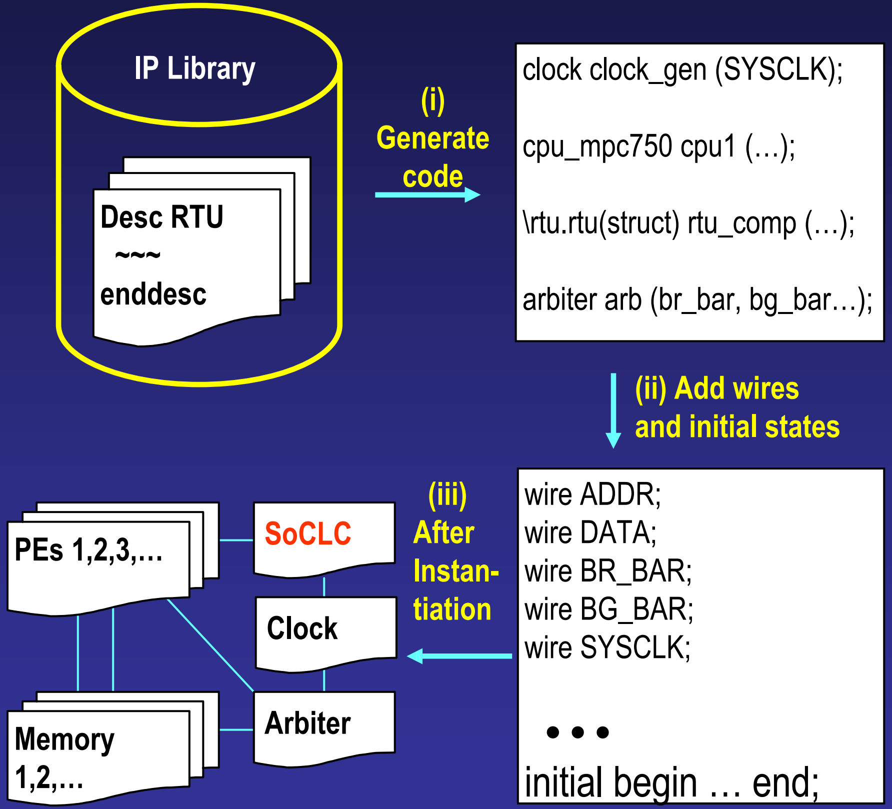
Specialized Software Components
- Deadlock Detection
- Memory Management

Hardware Components
- [✓ RTU]
- SoCLC
- SoCDDU
- SoCDMMU

IPC methods
- Semaphore
- Event
- MailBox
- Queue
- Mutual
- Allocation

Implementation

- Verilog top file generation example
  - Start with RTU description
  - Generate instantiation code
    - ✓ multiple instantiations of same unit if needed (e.g., PEs)
  - Add wires and initial statements

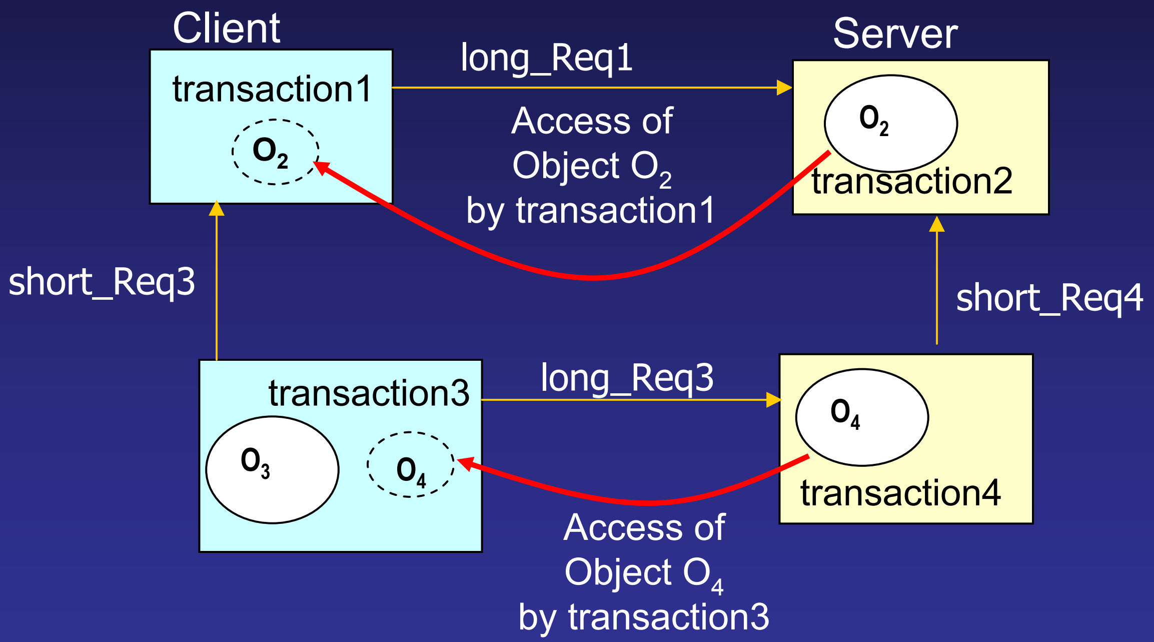


```
clock clock_gen (SYSCLK);
cpu_mpc750 cpu1 (…);
\rtu.rtu(struct) rtu_comp (…);
arbiter arb (br_bar, bg_bar…);
```

```
wire ADDR;
wire DATA;
wire BR_BAR;
wire BG_BAR;
wire SYSCLK;
• • •
initial begin … end;
```

Experimental Results (1/4)

- Application: Database transaction example [1]



[1] M. A. Olson, "Selecting and implementing an embedded database system," *IEEE Computer*, pp.27-34, September 2000.

Experimental Results (2/4)

- Comparison
  - A system with RTU hardware RTOS
  - A system with SoCLC hardware and software RTOS
  - A system with pure software RTOS

Total Execution Time		Pure SW *	With SoCLC	With RTU
6 tasks	(in cycles)	100398	71365	67038
	Speedup	0%	41%	50%
30 tasks	(in cycles)	379440	317916	279480
	Speedup	0%	19%	36%

* A semaphore is used in pure software and a hardware mechanism is used in SoCLC and RTU.

Experimental Results (3/4)

- The number of interactions

Times	6 tasks	30 tasks
Number of semaphore interactions	12	60
Number of context switches	3	30
Number of short locks	10	58

Experimental Results (4/4)

- The average number of cycles spent on communication, context switch and computation (6 task case)

cycles	Pure SW	With SoCLC	With RTU
communication	18944	3730	2075
context switch	3218	3231	2835
computation	8523	8577	8421

Hardware Area

Total area	SoCLC (64 short CS locks + 64 long CS locks)	RTU for 3 processors
TSMC 0.25 µm library from LEDA	7435 gates	About 250000 gates

Conclusion

- Comparison between RTU hardware RTOS and a hardware/software RTOS
- δ framework for automatic generation of configuration files for a custom HW/SW RTOS
- Future work
  - support for heterogeneous processors
  - support for multiple bus systems/structures