Design of a Hardware/Software RTOS for FPGAs with Processors

The δ Hardware/Software RTOS Generation Framework

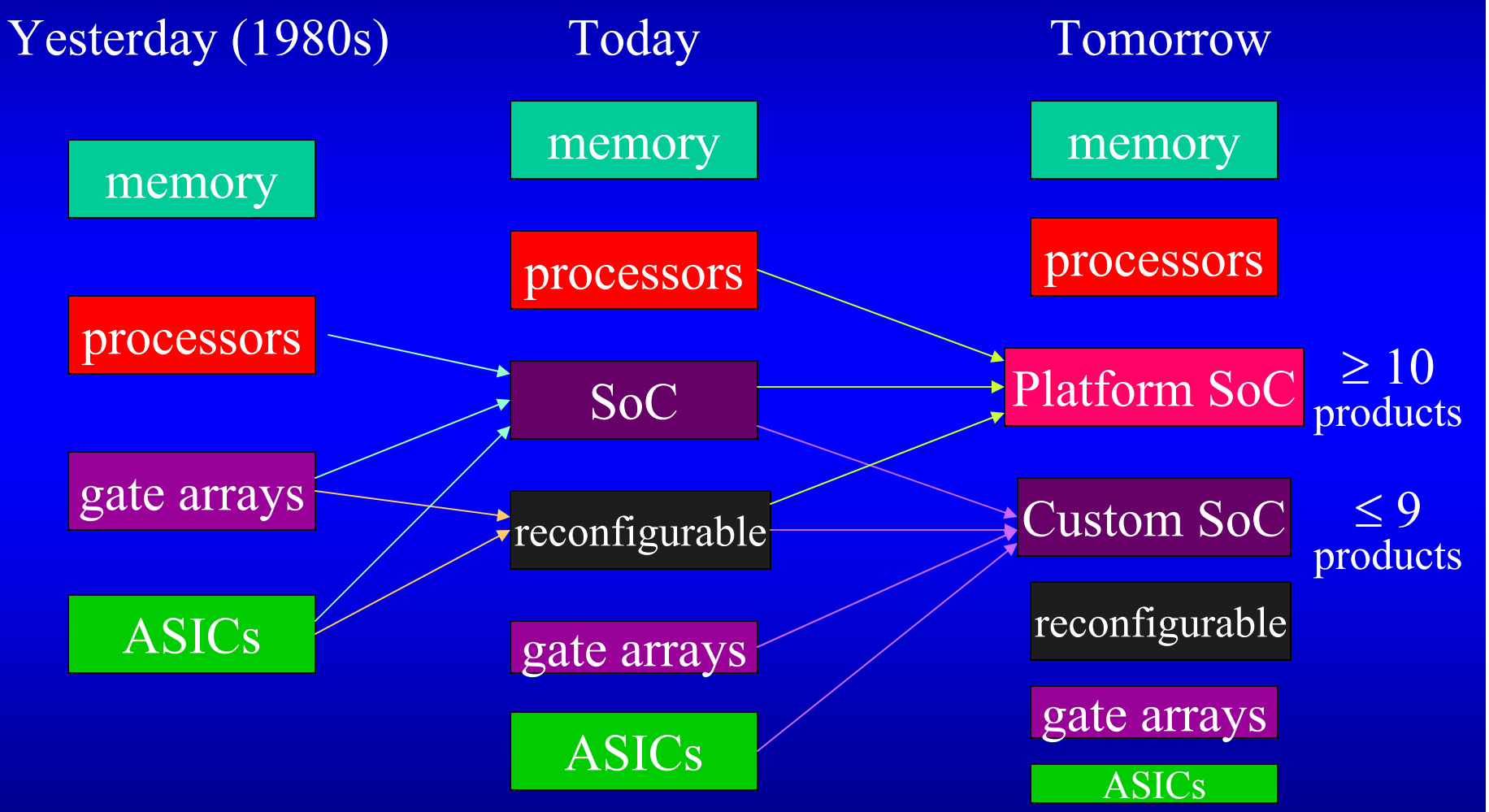
Vincent J. Mooney III
http://codesign.ece.gatech.edu
http://www.crest.gatech.edu

Associate Professor, School of Electrical and Computer Engineering
Adjunct Associate Professor, College of Computing
Associate Director, Center for Research on Embedded Systems and Technology
Georgia Institute of Technology
Atlanta, Georgia, USA

Outline

- Trends
- Vision: Hardware/Software Real-Time Operating System
- Custom RTOS Hardware IP Components
  - System-on-a-Chip Lock Cache (SoCLC)
  - SoC Deadlock Detection Unit (SoCDDU) and SoC Deadlock Avoidance Unit (SoCDAU)
- The δ Hardware/Software RTOS Generation Framework
  - Comparison with the RTU Hardware RTOS
- Conclusion

Digital Silicon CMOS VLSI Trends



Why Should SoC Customizers/Programmers Worry About RTOSes?

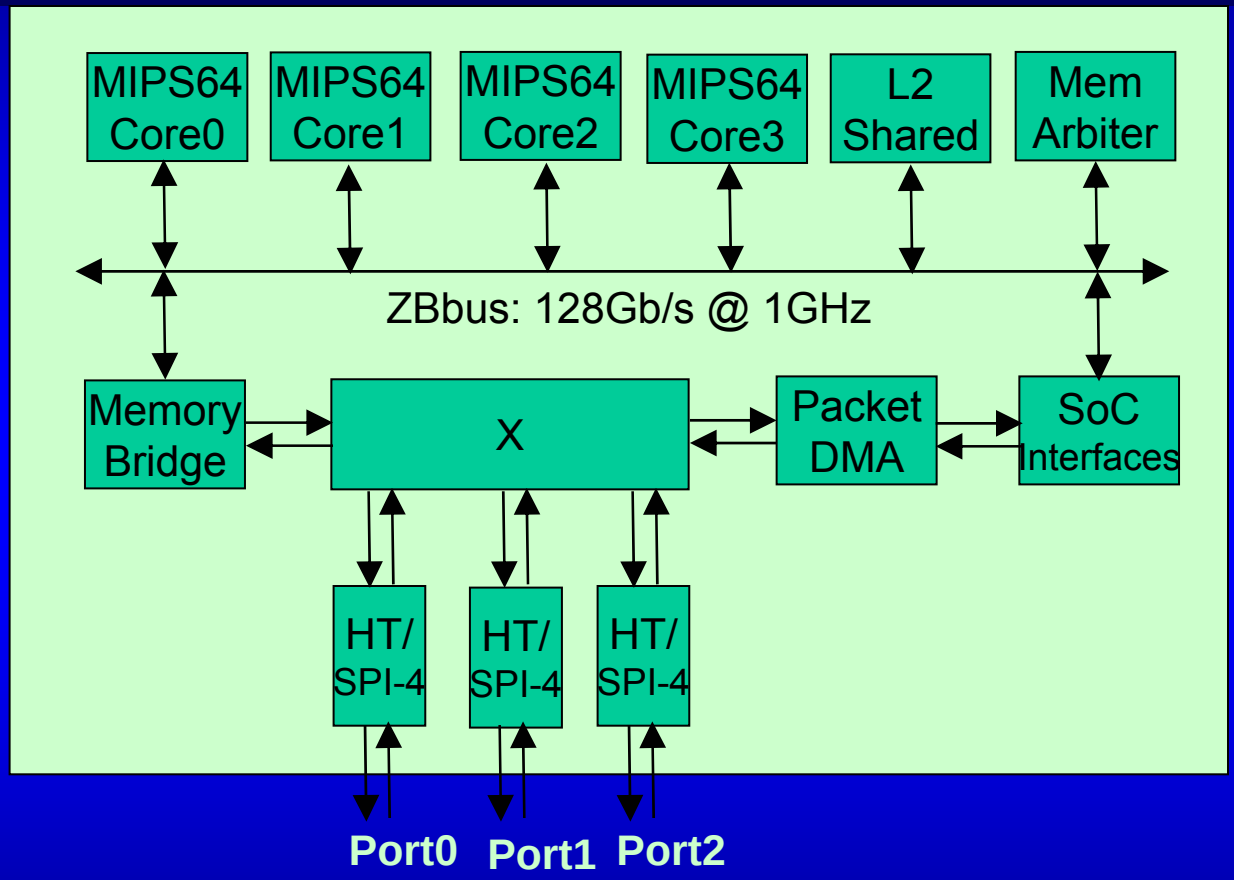
- Trend: logic design → RTL design → processors + custom RTL design → embedded software design!
- The RTOS is the state-of-the-art embedded software manager
- Prediction: the best embedded software designers will work closely with a custom RTOS design => the best SoC hardware/software designs will include tradeoffs in the RTOS

Vision: Dynamic Software/ Hardware RTOS Design

Key to System-on-a-Chip architecture optimization and customization

SoC Example: Broadcom BCM1400

- Four Processor Cores
  - MIPS64
  - 1GHz
  - 8-way 1MB Shared L2
- On-chip ZBbus
  - maintains coherency
  - proprietary
- Off-chip HT/SPI-4 19Gb/s

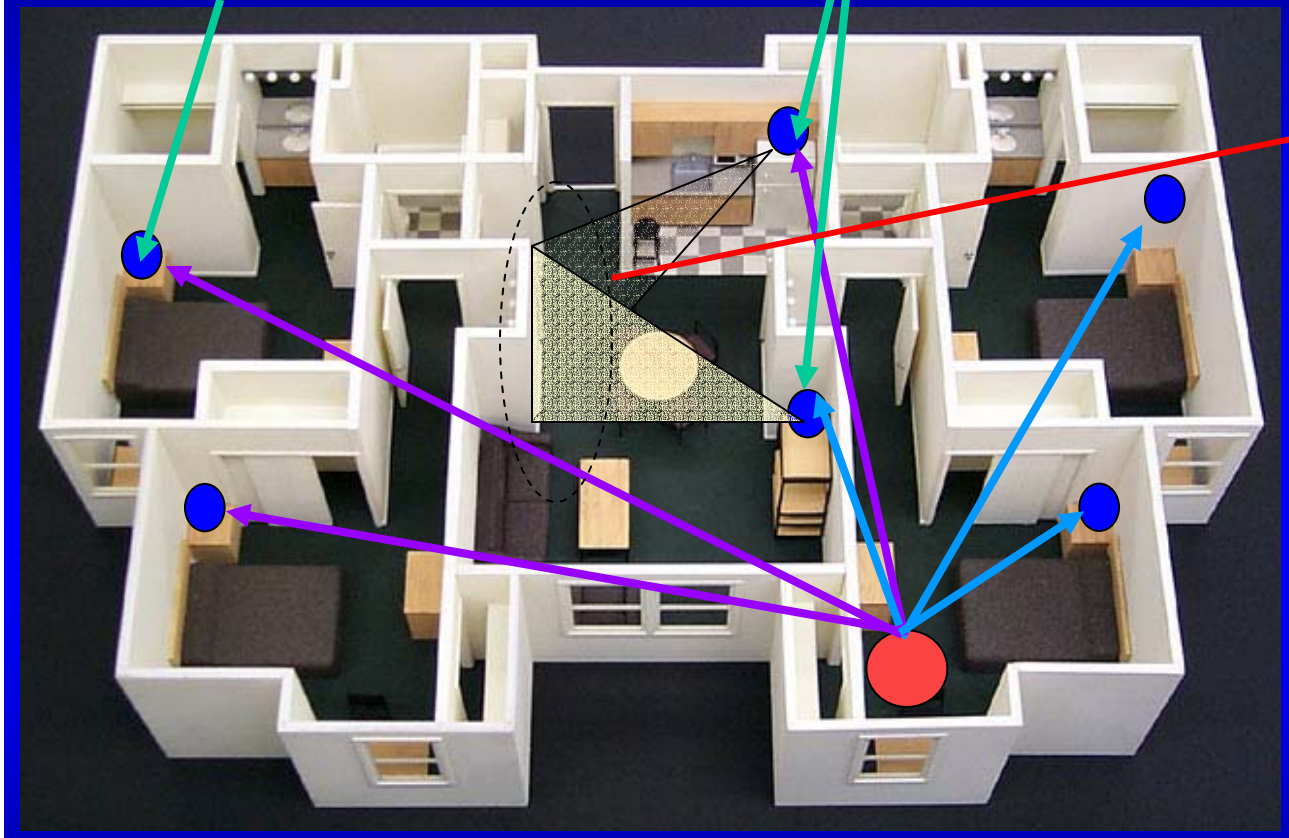


[Levy02] M. Levy, "Chip Combines Four 1GHz Cores," *Microprocessor Report*, pp. 12-14, October 2002.

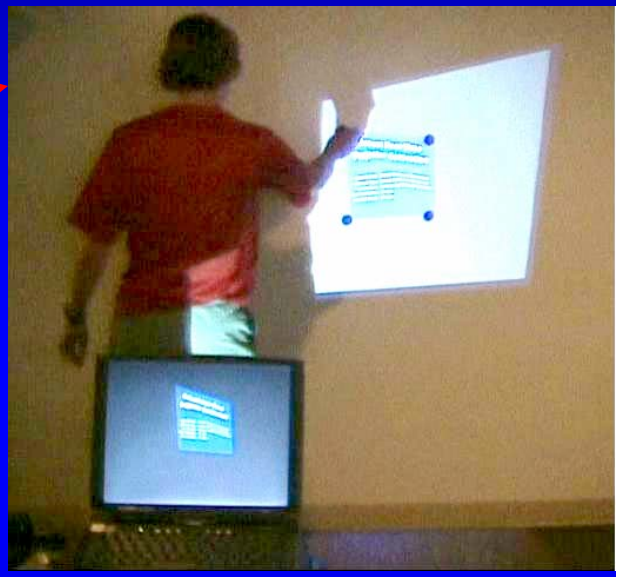
Motivational Example: Home 2005

Programmable PDA

Projectors



Projected Light Displays



SoC Device

Central Storage Unit (e.g., PC)

wireless link

wired link

Analogy

- Microprocessor design
  - Compiler
  - Computer architecture
- SoC design
  - Dynamic hw/sw RTOS
  - SoC architecture

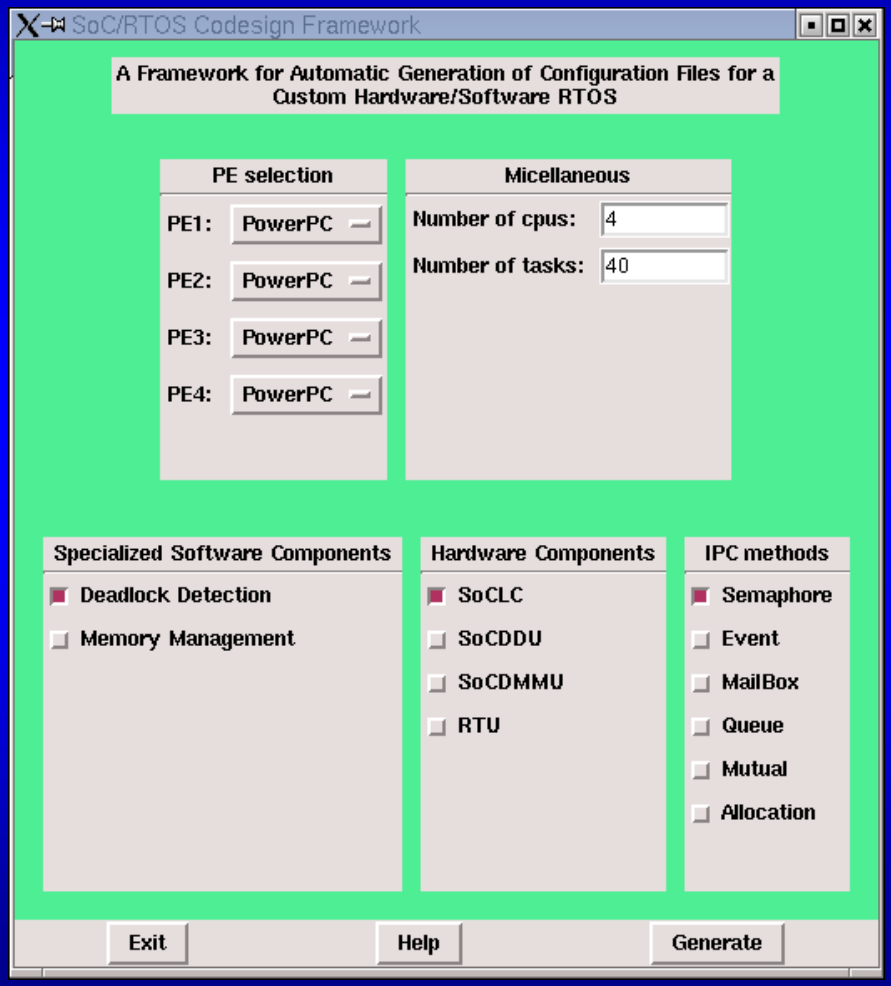
Building Blocks

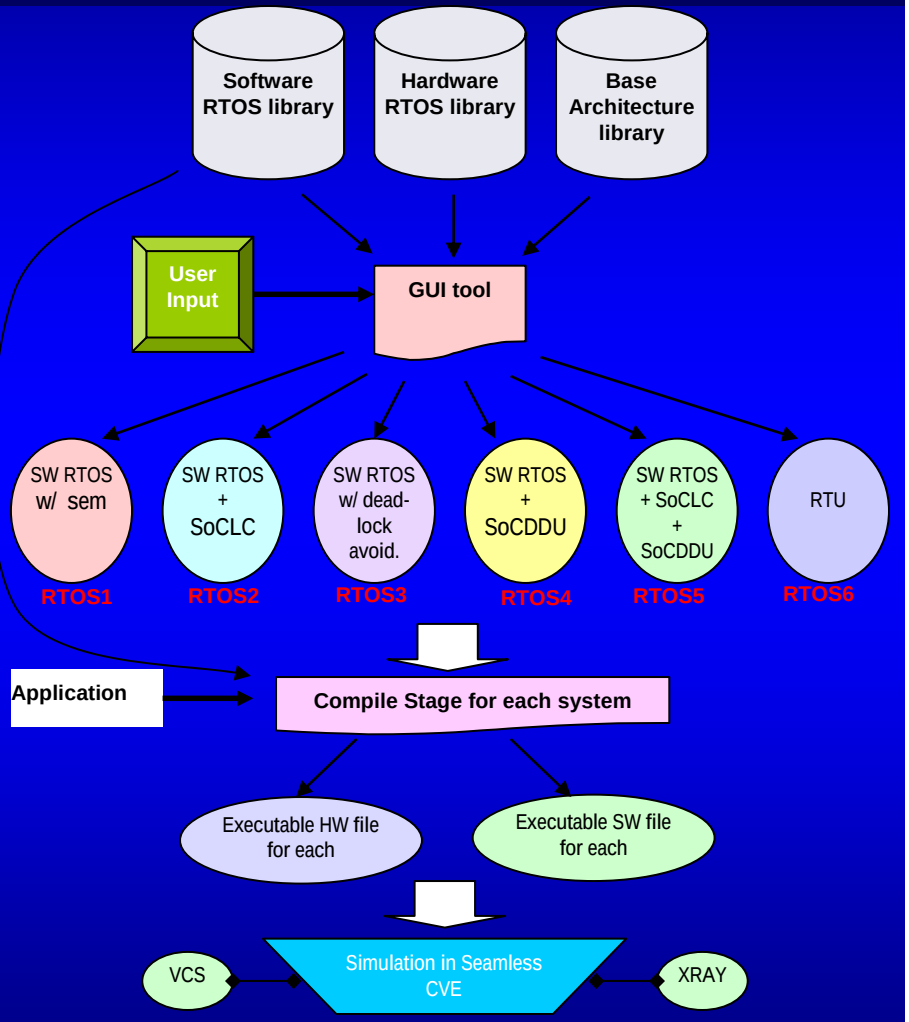
- SoC Programming Model
  - multi-threading, shared mem., message passing, control-data flow graph
- SoC Programming Environment
  - δ Hardware/Software RTOS
- Microprocessor Programming Model
  - C/C++/Java/other serial language
- Microprocessor Programming Environment
  - gcc, various

Approach

- δ Hw/Sw RTOS made up of library components
- Library component = predefined C code, assembly code or HDL code
- Similar to existing RTOS's, except for the HDL code
  - ex.: SoC Lock Cache in hardware
- RTOS HDL code can be automatically generated by a custom "IP Generator"
  - ex.: PARLAK SoC Lock Cache generator

The δ Hardware/Software RTOS Generation Framework



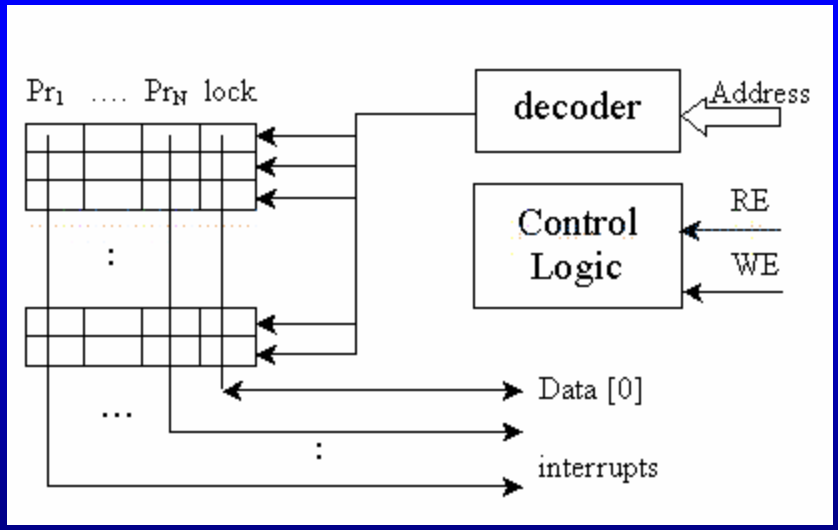


Outline

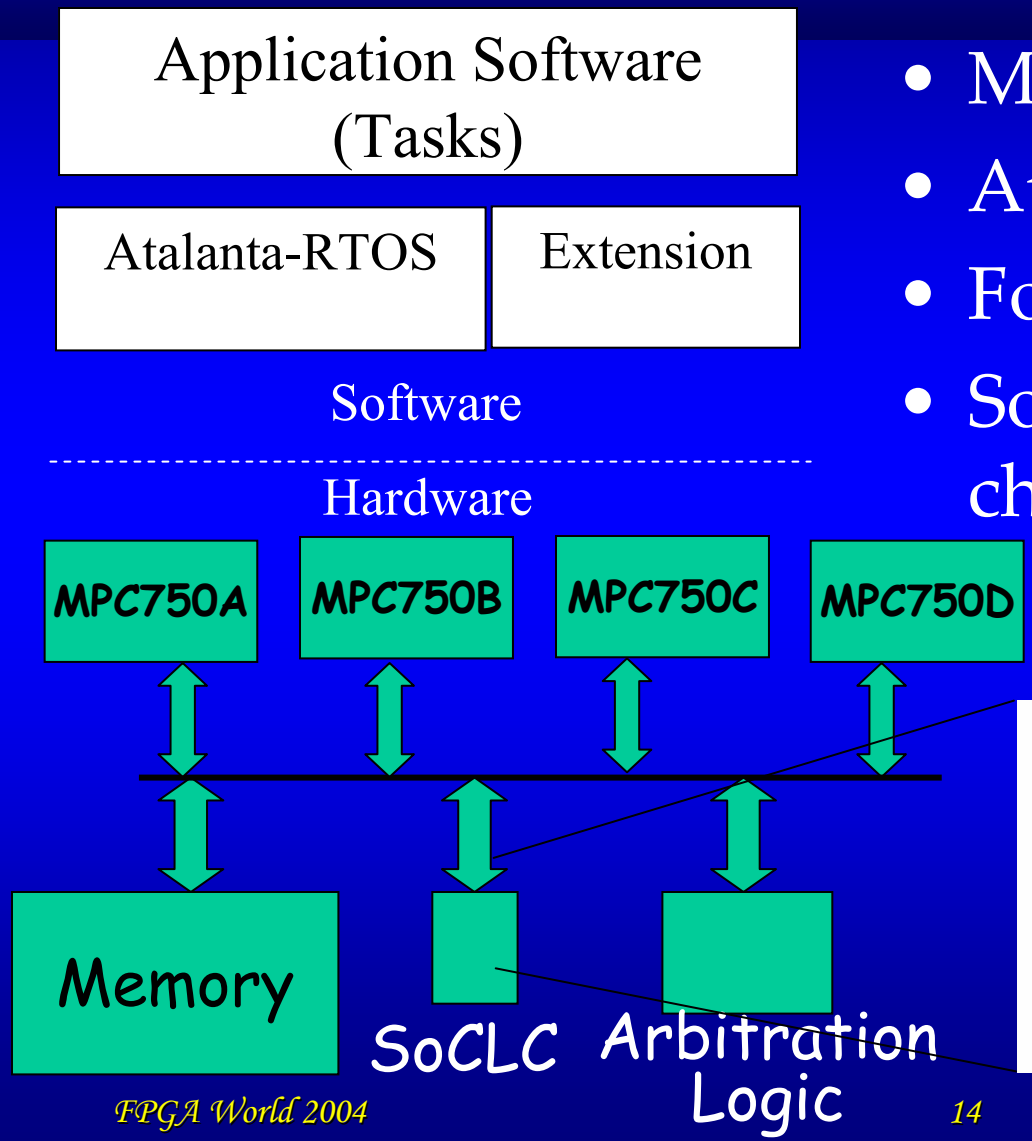
- Trends
- Vision: Hardware/Software Real-Time Operating System
- Custom RTOS Hardware IP Components
  - System-on-a-Chip Lock Cache (SoCLC)
  - SoC Deadlock Detection Unit (SoCDDU) and SoC Deadlock Avoidance Unit (SoCDAU)
- The δ Hardware/Software RTOS Generation Framework
  - Comparison with the RTU Hardware RTOS
- Conclusion

SoC 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
- Improves the performance criteria in terms of lock latency, lock delay and bandwidth consumption

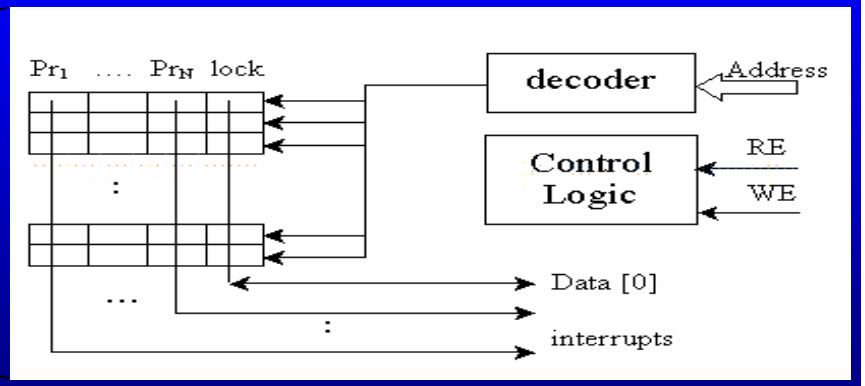


Software/Hardware Architecture



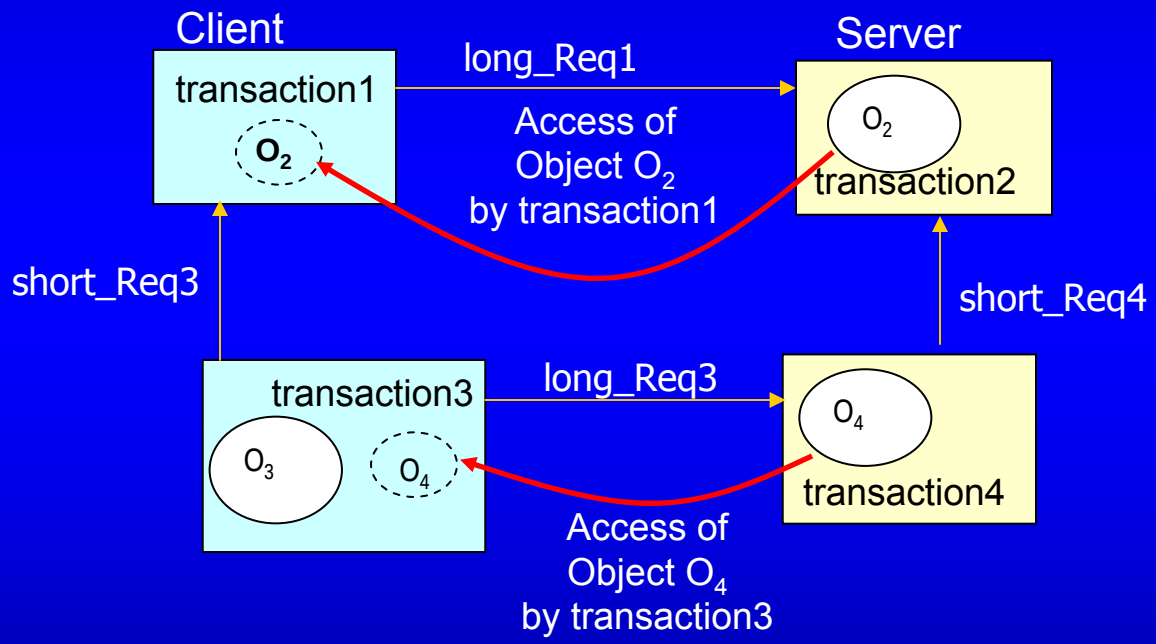
- Multiple application tasks
- Atalanta-RTOS
- Four MPC750s
- SoCLC provides lock synchronization among PEs

SoC Lock Cache



Experiment

- Example: Database transaction application [1]



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

Experimental Result

- Comparison with database application example [2]
  - RTOS1 with semaphores and spin-locks
  - RTOS2 with SoCLC, no SW semaphores or spin-locks

(clock cycles)	* Without SoCLC	With SoCLC	Speedup
Lock Latency	1200	908	1.32x
Lock Delay	47264	23590	2.00x
Execution Time	36.9M	29M	1.27x

* Semaphores for long critical sections (CSes) and spin-locks for short CSes are used instead of SoCLC.

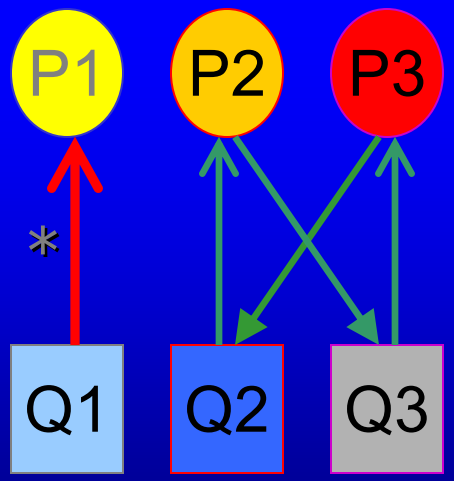
[2] B. S. Akgul, J. Lee and V. Mooney, "System-on-a-chip processor synchronization hardware unit with task preemption support," *CASES '01*, pp.149-157, November 2001.

Outline

- Trends
- Vision: Hardware/Software Real-Time Operating System
- Custom RTOS Hardware IP Components
  - System-on-a-Chip Lock Cache (SoCLC)
  - SoC Deadlock Detection Unit (SoCDDU) and SoC Deadlock Avoidance Unit (SoCDAU)
- The δ Hardware/Software RTOS Generation Framework
  - Comparison with the RTU Hardware RTOS
- Conclusion

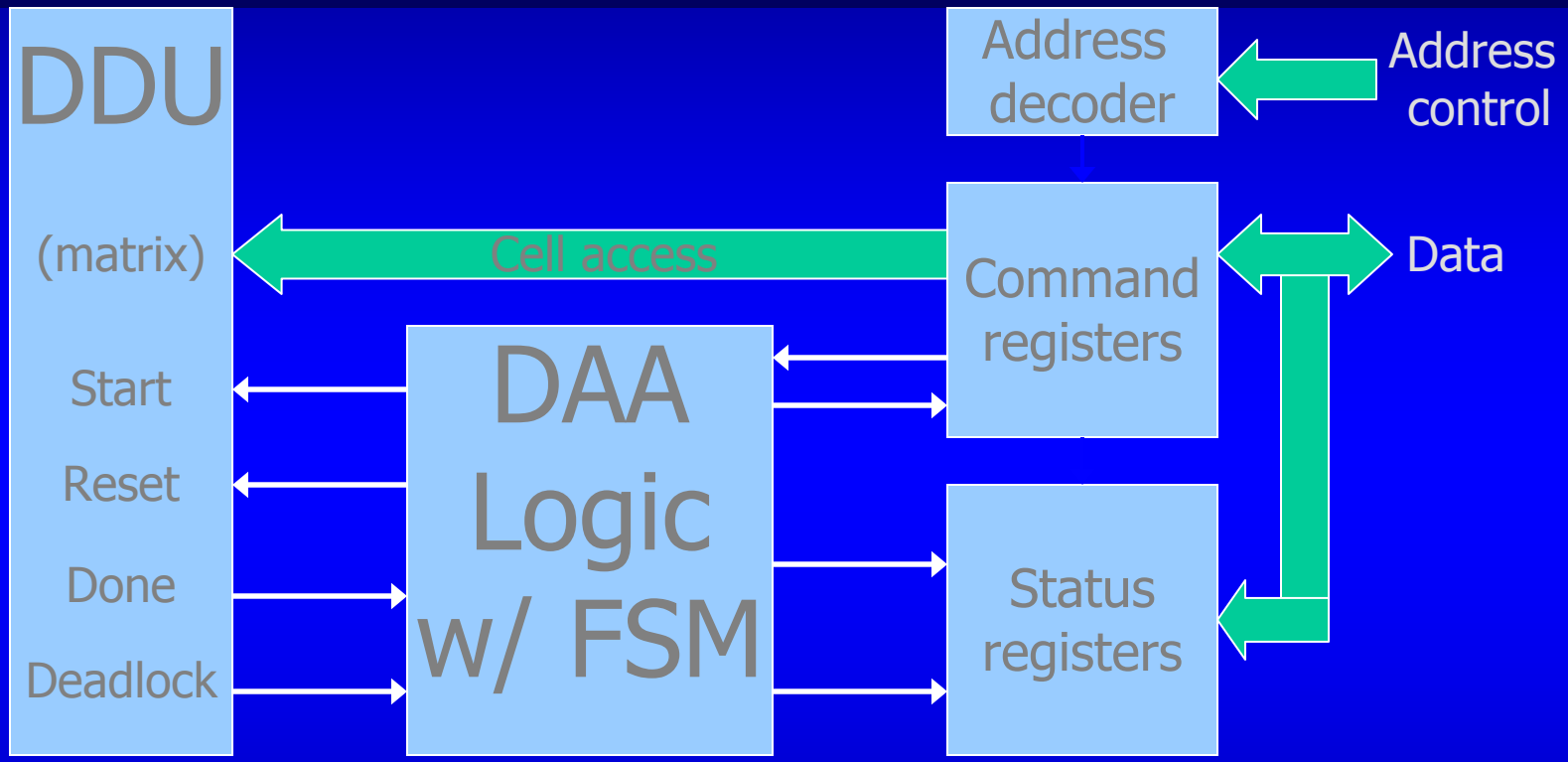
SoC Deadlock Detection Unit (SoCDDU)

- Matrix based parallel processing approach
- Removable edge reduction technique to reveal cycles (i.e., deadlock)
  - Removable edge*: not related to deadlock
- Simple bit-wise Boolean operations
  - Implementation easier
  - Operation faster, O(min(m,n))
  - 2~3 orders of magnitude faster than software
- Novelty from previous algorithms
  - Does NOT trace exact cycles
  - Does NOT require linked lists



P. Shiu, Y. Tan and V. Mooney, "A Novel Parallel Deadlock Detection Algorithm and Architecture," *9th International Workshop on Hardware/Software Codesign* (CODES'01), pp. 30-36, April 2001.

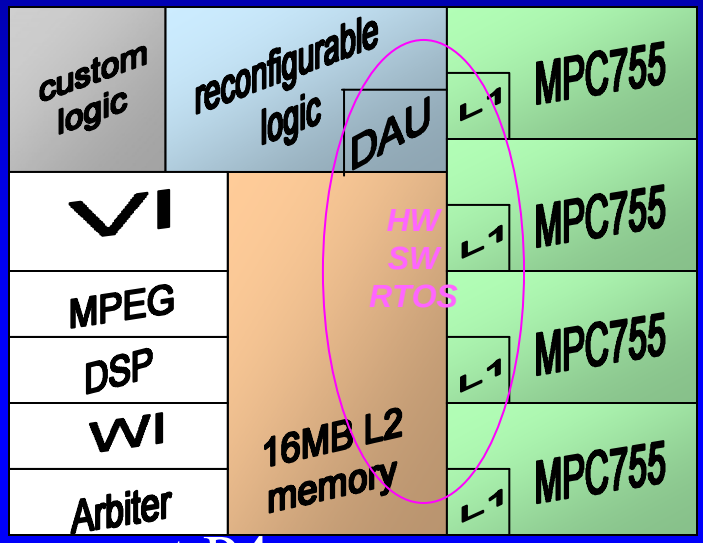
Architecture of the SoC Deadlock Avoidance Unit (SoCDAU)



DDU: Deadlock Detection Unit
FSM: finite state machine

SoCDDU/SoCDAU Experiment - System and Application

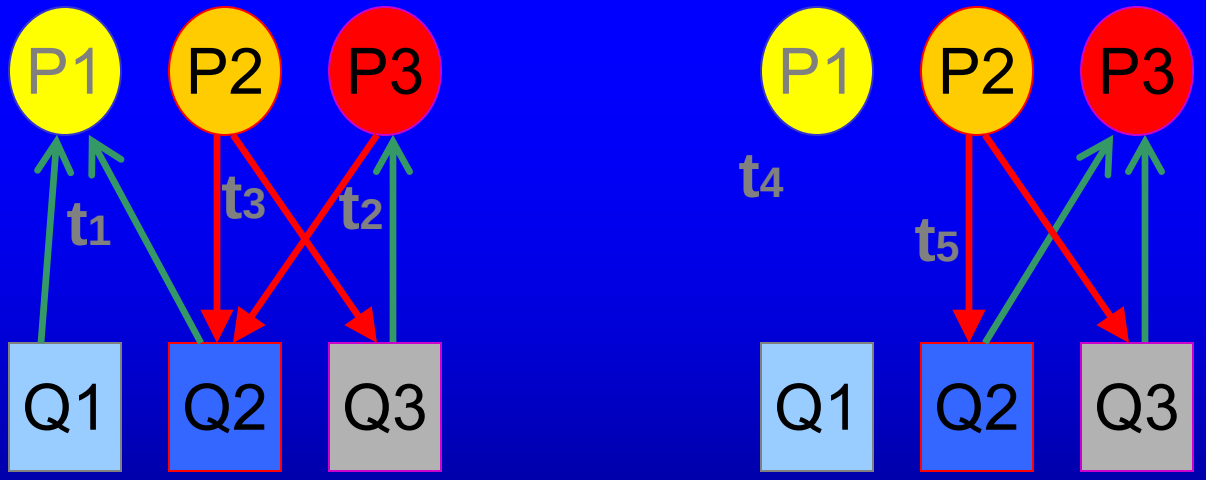
- Four resources
  - Q1: Video Interface (VI)
  - Q2: MPEG
  - Q3: DSP
  - Q4: Wireless Interface (WI)



- Each process requires two resources except P4
  - P1: processing a video stream (needs Q1 + Q2)
  - P2: separating/enhancing frames (needs Q2 + Q3)
  - P3: extracting special images (needs Q3 + Q1)
  - P4: transferring images (needs Q4)
- One active process for each processing element (PE)

Experimental Results of SoCDAU

- Deadlock avoidance simulation: Two kinds
  - SoCDAU: Synopsys VCS runs compiled Verilog code
  - DAA in software: MPC755 runs compiled C code in Seamless CVE
  - Example application invokes deadlock avoidance 12 times



Experimental Results of SoCDAU (cont'd)

- Deadlock avoidance simulation result
- Performance improvement
  - 99% algorithm execution time reduction
  - 37% reduction in an application execution time

Method of Deadlock Avoidance	Algorithm Exe. Time (cycles)	Normalized Exe. Time	Application Exe. Time (cycles)
SoCDAU Hardware	7 (average)	1X	34791
DAA in Software	2188 (average)	312X	47704

Synthesis Results of SoCDAU

- SoCDAU for 5 processes and 5 resources
- Qualcore Logic library with TSMC .25µm technology
- 0.01% of the total SoC area with four PEs and memory

Module Name	Total Area in terms of two-input NAND gates	Lines of Verilog HDL Code
SoCDDU 5x5 (inside SoCDAU)	364	203
DAA Logic	1472	344

TSMC: Taiwan Semiconductor Manufacturing Company
PE: Processing Element

Hardware Area

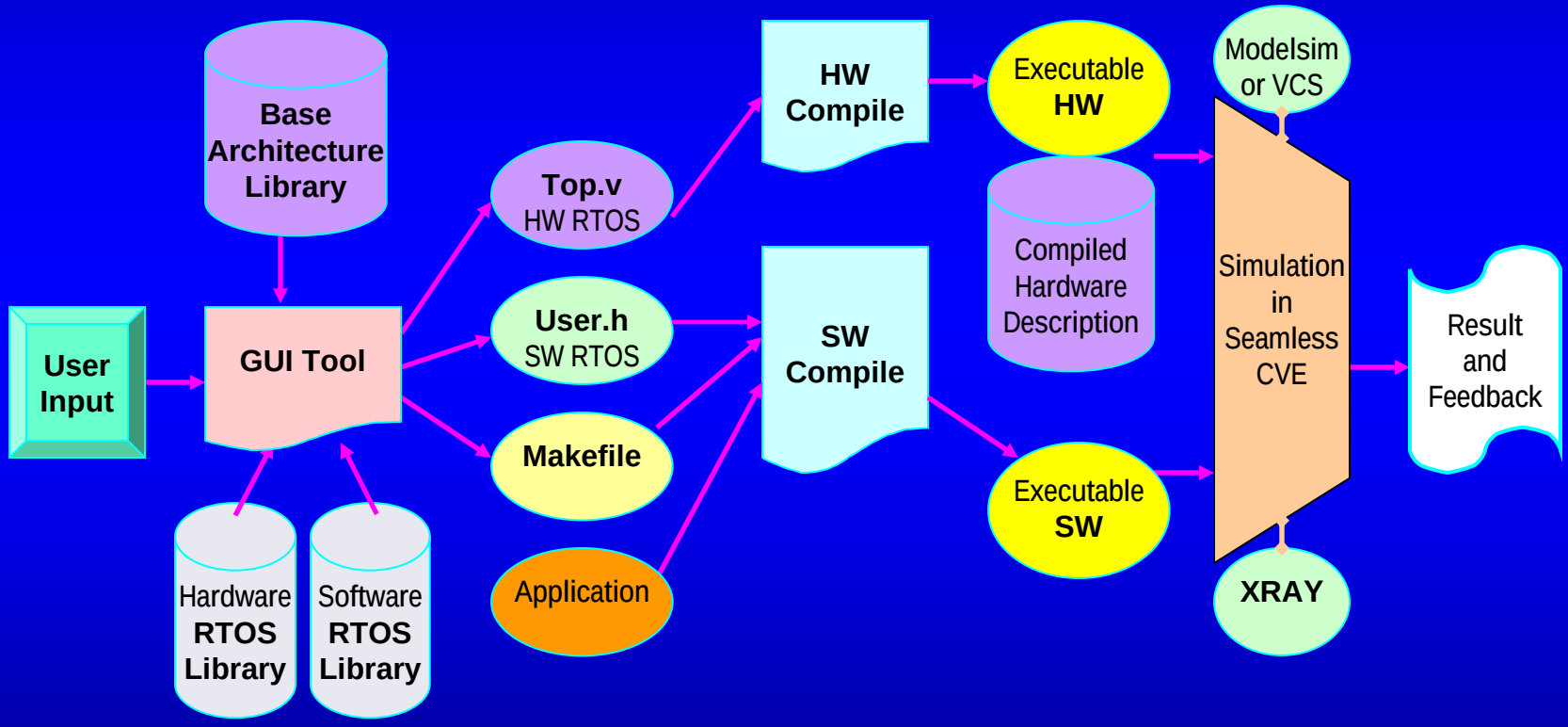
Total area in	SoCLC (For 64 short CS locks + 64 long CS locks)		SoCDDU (For 5 Processors x 5 Resources)
Semi-custom VLSI	7435 gates using TSMC 0.25µm standard cell library		364 gates using AMI 0.3µm standard cell library
Xilinx XC4000E 4003EPC84	# Seq. logic	532	10
	# Other gates	9036	559

Outline

- Trends
- Vision: Hardware/Software Real-Time Operating System
- Custom RTOS Hardware IP Components
  - System-on-a-Chip Lock Cache (SoCLC)
  - SoC Deadlock Detection Unit (SoCDDU) and SoC Deadlock Avoidance Unit (SoCDAU)
- The δ Hardware/Software RTOS Generation Framework
  - Comparison with the RTU Hardware RTOS
- Conclusion

δ Hardware/Software RTOS Generation Framework

and current simulation platform



δ Hardware/Software RTOS Generation Framework Goals

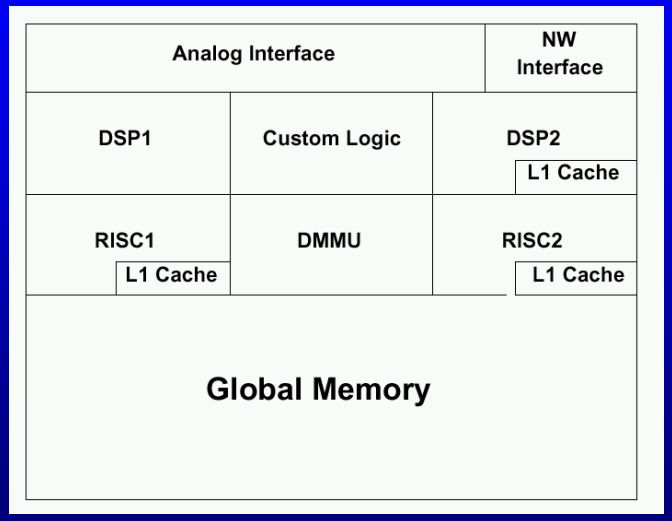
- To help the user examine which configuration is most suitable for the user's specific applications
- To help the user explore the RTOS design space before chip fabrication as well as after chip fabrication (in which case reconfigurable logic must be available on the chip)
- To help the user examine different System-on-a-Chip (SoC) architectures subject to a custom RTOS

Motivation (1/3)

- HW/SW RTOS partitioning approach
- Previous innovations in HW/SW RTOS components
  - System-on-a-Chip Lock Cache (SoCLC)
  - System-on-a-Chip Dynamic Memory Management Unit (SoCDMMU)
  - System-on-a-Chip Deadlock Detection Unit (SoCDDU) and Deadlock Avoidance Unit (SoCDAU)
- RTU Hardware RTOS

Motivation (2/3)

- 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 (3/3)

Constraints about using the previous HW/SW RTOS innovations

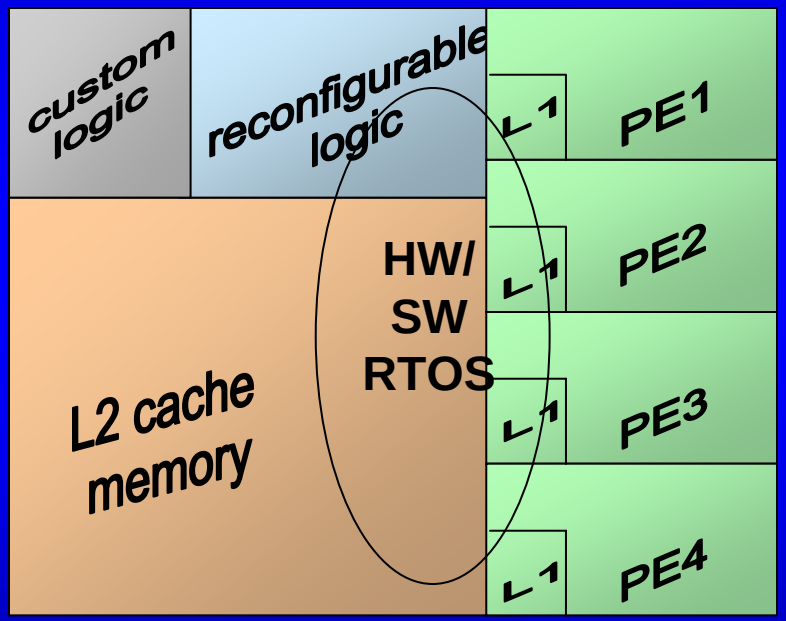
- Perhaps not enough chip space for all three of them
- All of them may not be necessary

⇒ The δ framework

- Enables automatic generation of different mixes of the previous innovations for different versions of a HW/SW RTOS
- Enables selection of the RTU hardware RTOS
- Can be generalized to instantiate additional HW or SW RTOS components

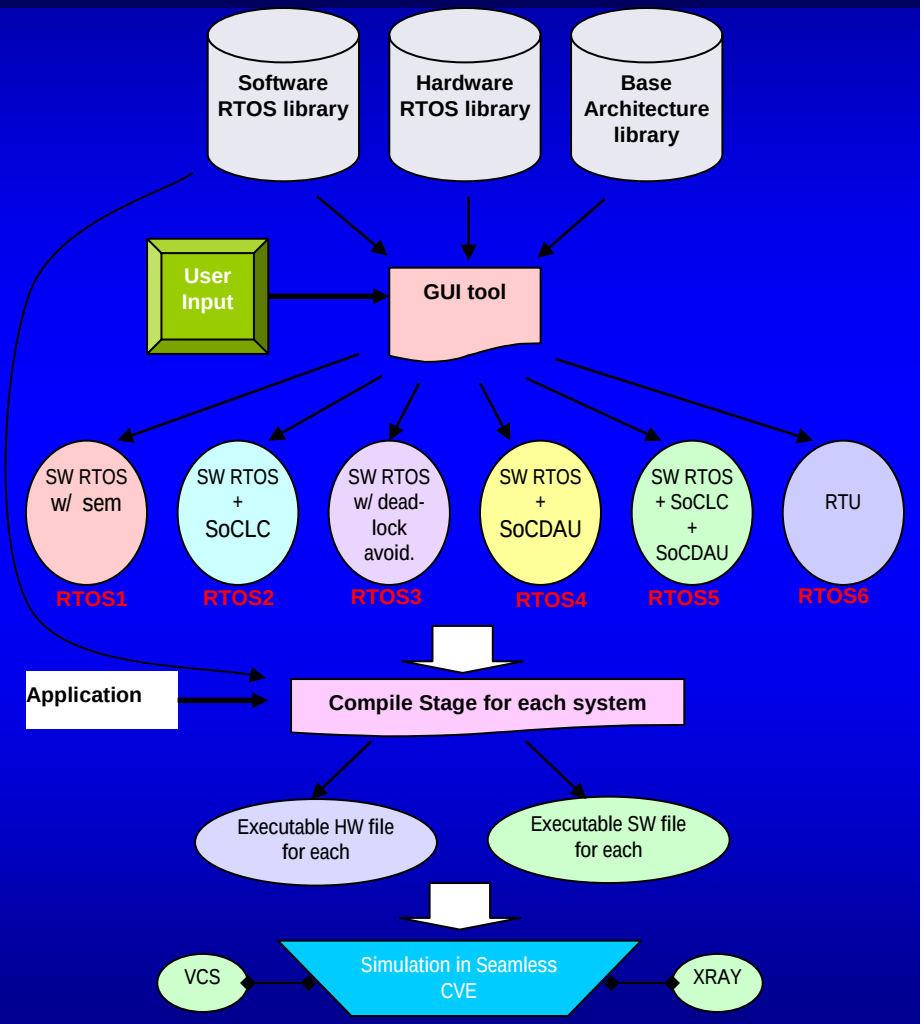
Our RTOS in Post-Fabricaton Scenario

- Application(s) run on the SoC using standard RTOS APIs
- 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
- Hardware RTOS components are downloaded into the reconfigurable logic

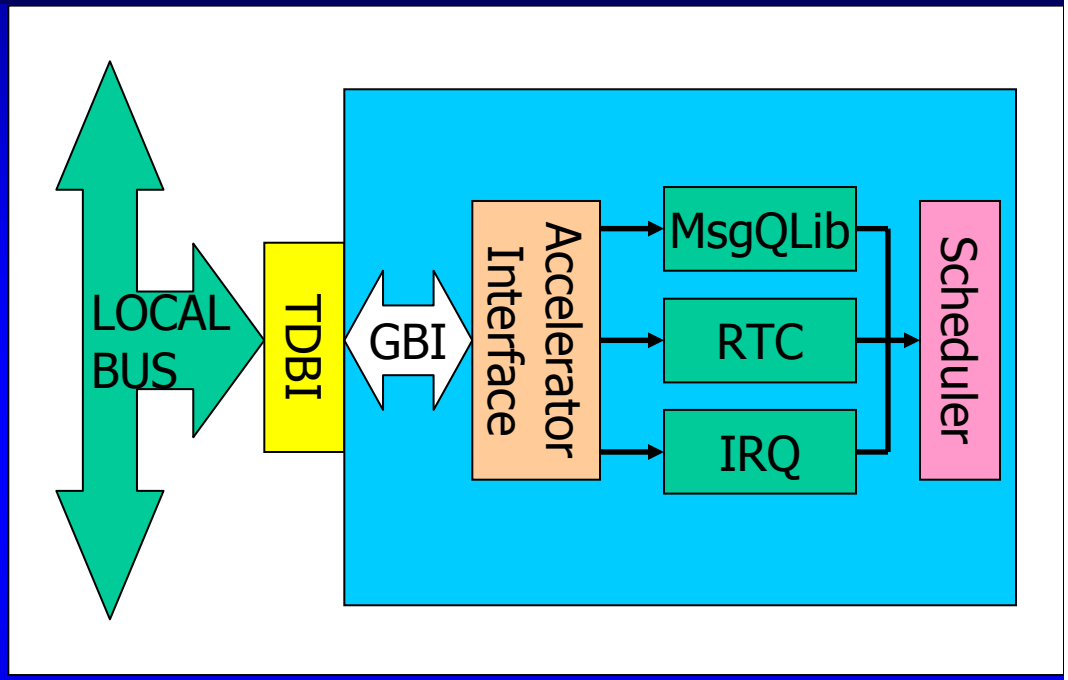


Experimental Setup

- Six custom RTOSes
  - With semaphores and spin-locks, no HW components in the RTOS
  - With SoCLC, no SW IPCs
  - With deadlock avoidance software, no HW RTOS components
  - With SoCDAU
  - With SoCLC and SoCDAU
  - With RTU
- Each with the *Base* architecture
- Each with application(s)
- Each executable in Seamless CVE
  - 4 MPC750 processors
  - Reconfigurable logic
  - Single bus



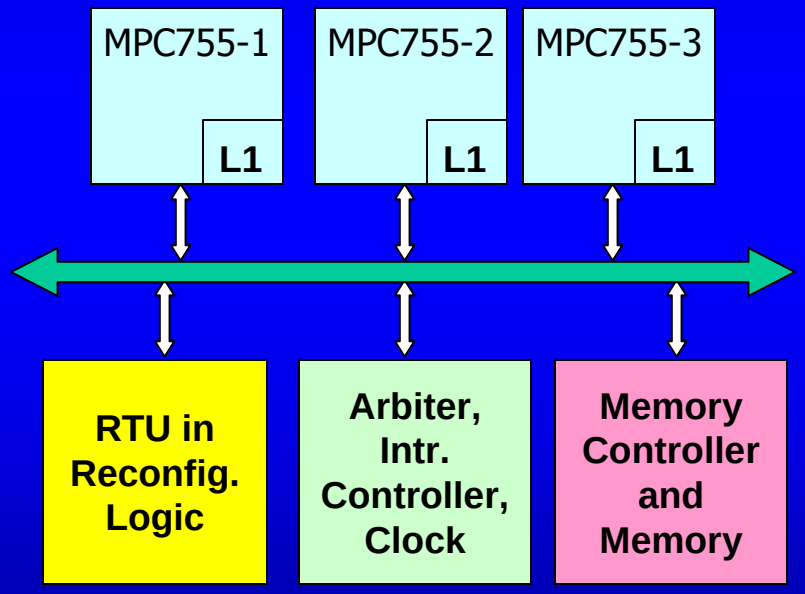
RTU Hardware RTOS



- 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

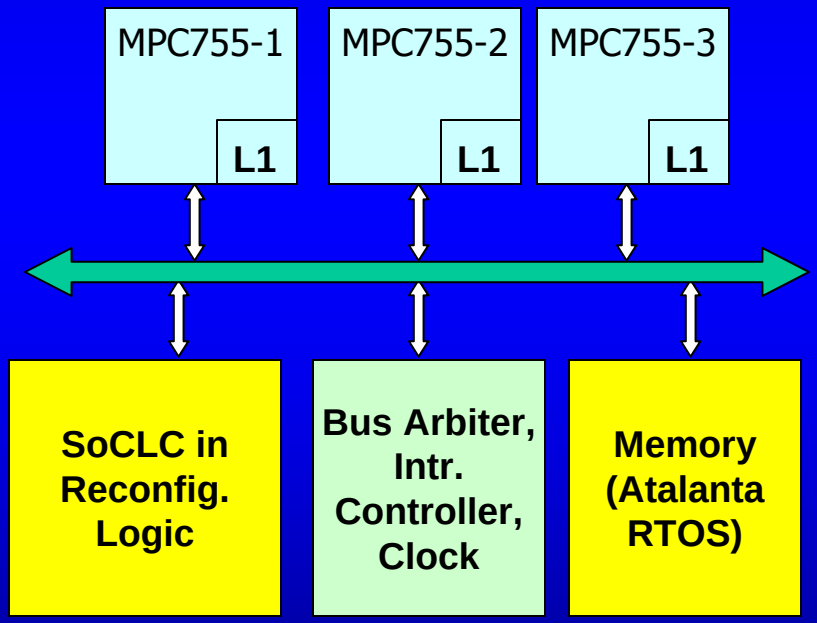
Methodology

- An SoC architecture with the RTU Hardware RTOS



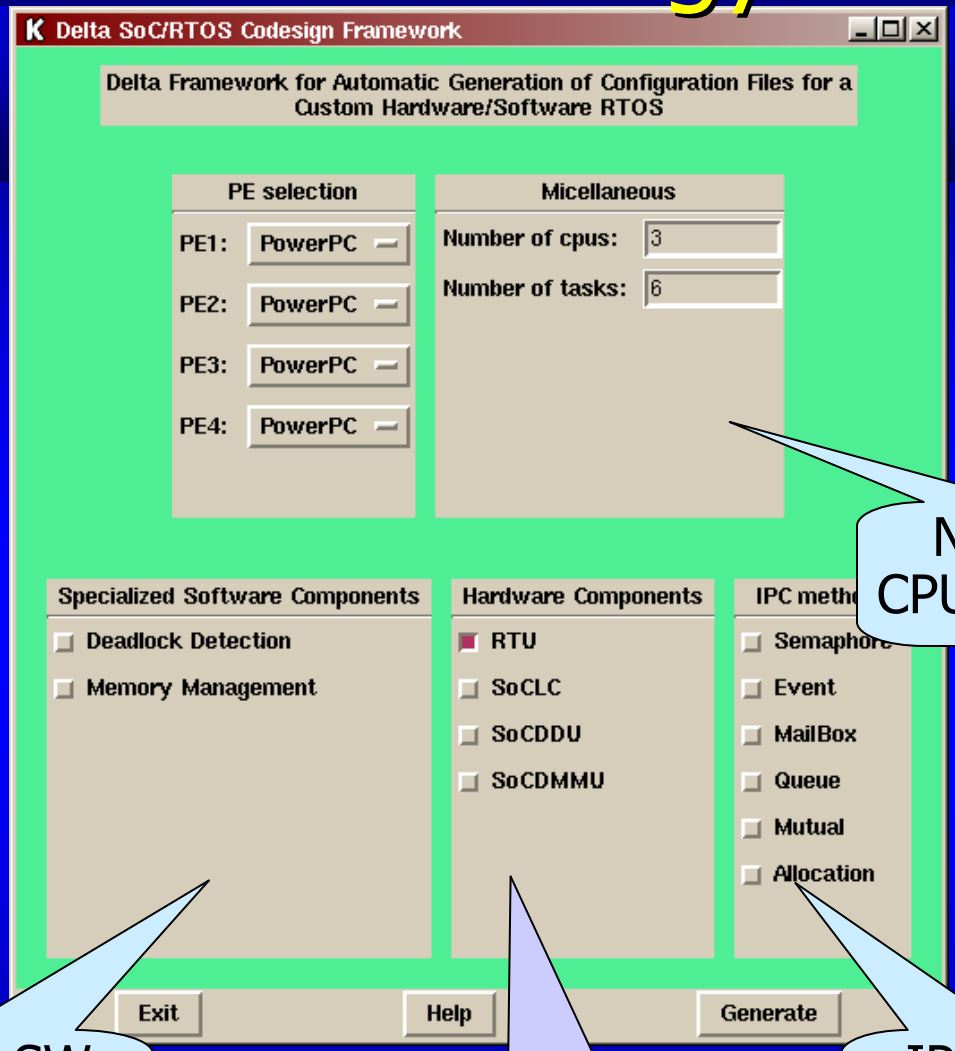
Methodology

- An SoC architecture with a hardware/software RTOS



Methodology

- δ Framework
  - GUI



Number of CPUs in system

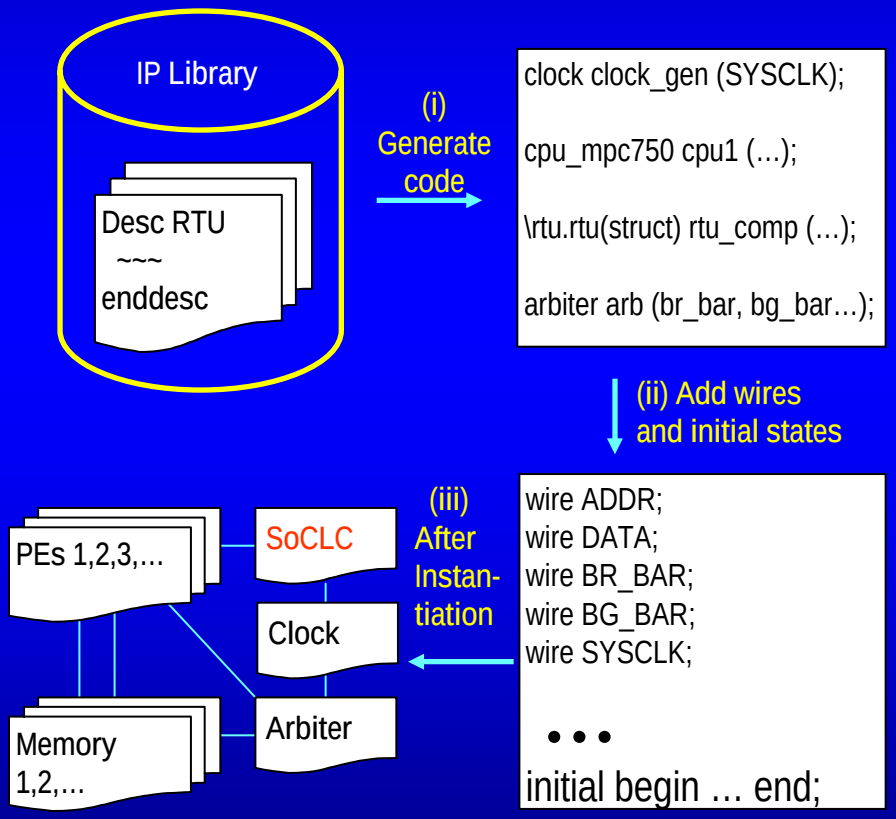
Specilized SW RTOS component

HW RTOS component

IPC module linking method

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



Experimental Results (1/3)

- 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
	Reduction	0%	29%	33%
30 tasks	(in cycles)	379440	317916	279480
	Reduction	0%	16%	26%

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

Experimental Results (2/3)

- 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 0.25µm library from LEDA	7435 gates	About 250000 gates

Conclusion

- A framework for automatic generation of a custom HW/SW RTOS
- Experimental results showing
  - a multiprocessor FPGA that utilizes the SoCDAU/SoCDDU has a 37% overall speedup of the application execution time over deadlock avoidance in software
  - speedups with the SoCLC, RTU
  - addition hw RTOS component in references: SoCDMMU
- Future work
  - support for heterogeneous processors
  - support for multiple bus systems/structures