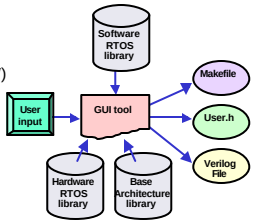
Framework for Automatic Generation of Bus Systems and a Hw/Sw RTOS for Multiprocessor SoC

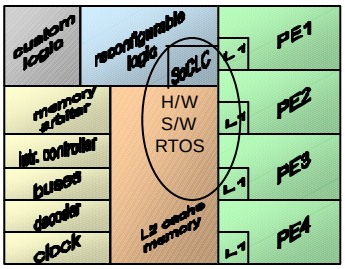
Bilge E. S. Akgul, Jaehwan Lee, Kyeong K. Ryu, Mohamed Shalan, Eung Shin, Taeweon Suh, Di-Shi Sun, Vincent J. Mooney and Douglas Blough
School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, U.S.A.
http://codesign.ece.gatech.edu/

Motivation

- Implement a graphical user interface (GUI) tool
- Generate configuration files used to make a custom RTOS
  - A custom RTOS may contain HW (as well as SW) components
- Compile both hardware and software with an application
- Simulate the system to see the result



Multiprocessor system-on-a-chip (SoC)
A multiprocessor RTOS: **AtalantaRTOS**
An application on top of the system



GOALS:

- To help the user examine which configuration is most suitable for his or her specific applications
- To help the user explore the RTOS design space after chip fabrication as well as before chip fabrication
- To help the user examine different SoC architectures subject to a custom RTOS

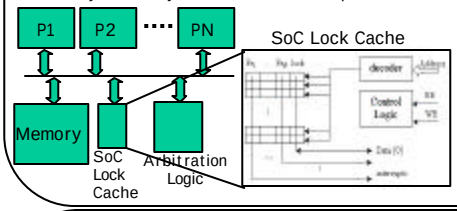
Innovations

SoCLC: System-on-a-Chip Lock Cache

A hardware mechanism that resolves the critical section 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

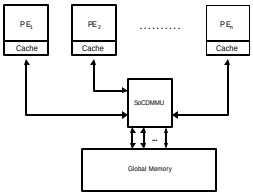


SoCDMMU: System-on-a-Chip Dynamic Memory Management Unit

Provides fast, deterministic and yet dynamic memory management of a global on-chip memory

Achieves flexible, efficient memory utilization

Provides APIs for applications



SoCDDU: System-on-a-Chip Deadlock Detection Unit

Performs a novel parallel hardware deadlock detection based on searches on the resource allocation matrix

Provides a very fast deadlock detection at run-time performing simple bit-wise boolean operations

Reduces deadlock detection time by 99% as compared to software

What is missing?

Perhaps not enough chip space for all three of them

All of them may not be necessary

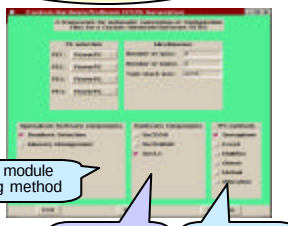
Solution: Our framework

Enables automatic generation of different mixes of HW/SW versions with the three previous innovations

Implementation

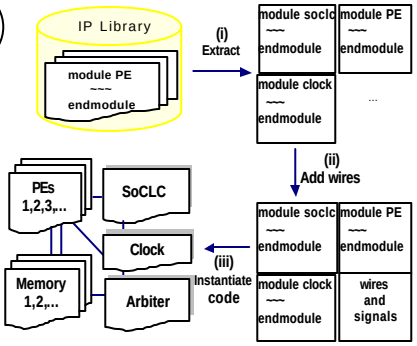
Selectable SW/HW IP components

CPU schedulers (priority, round-robin)
Inter-Process Communication components
Memory management module (gmm)
Deadlock detection module (ddm)
SoCLC
SoCDDU
SoCDMMU



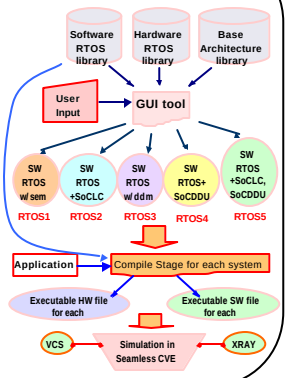
SW module linking method
HW integration method
IPC module linking method

Verilog header file generation example



With semaphores and spin-locks, no HW RTOS
With SoCLC, no SW IPCs
With deadlock detection software, no HW RTOS
With SoCDDU, no SW IPCs
With SoCLC and SoCDDU
Each with the *Base* architecture
Each with an application
Each executable in the Seamless CVE

Five custom RTOSs



Experimental Results

SoCLC

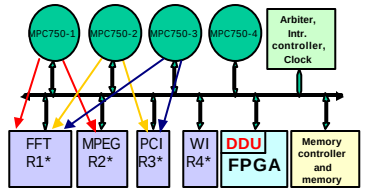
- Comparison with database application example [1]
- RTOS1 with semaphores and spin-locks
- RTOS2 with SoCLC, no SW semaphores or spin-locks

(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.

[1] 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, Nov. 2001.

SoCDDU



Method of Deadlock Detection	Software Algorithm	SoCDDU	Speedup
Detection Time Δ (clock cycles)	16928	**2**	8463x
Execution time up to deadlock detection	61131	44205	**1.38x**

* Note: Resources are not actual modules but dummies containing timers

[2] S. Morgan, "Jini to the rescue," *IEEE Spectrum*, 37(4), pp 44-49, April 2000.
[3] P. H. Shiu, Y. Tan and V. Mooney, "A novel parallel deadlock detection algorithm and architecture," *CODES '01*, pp.30-36, April 2001.

Hardware Chip Area

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

Conclusion

A framework for automatic generation of configuration files for a custom HW/SW RTOS

A Novel HW header file generation methodology

Experimental results show: the configured systems are correct

A framework used to explore the RTOS design space.

More Results and d-Hardware/Software RTOS publications are available on web: http://codesign.ece.gatech.edu/