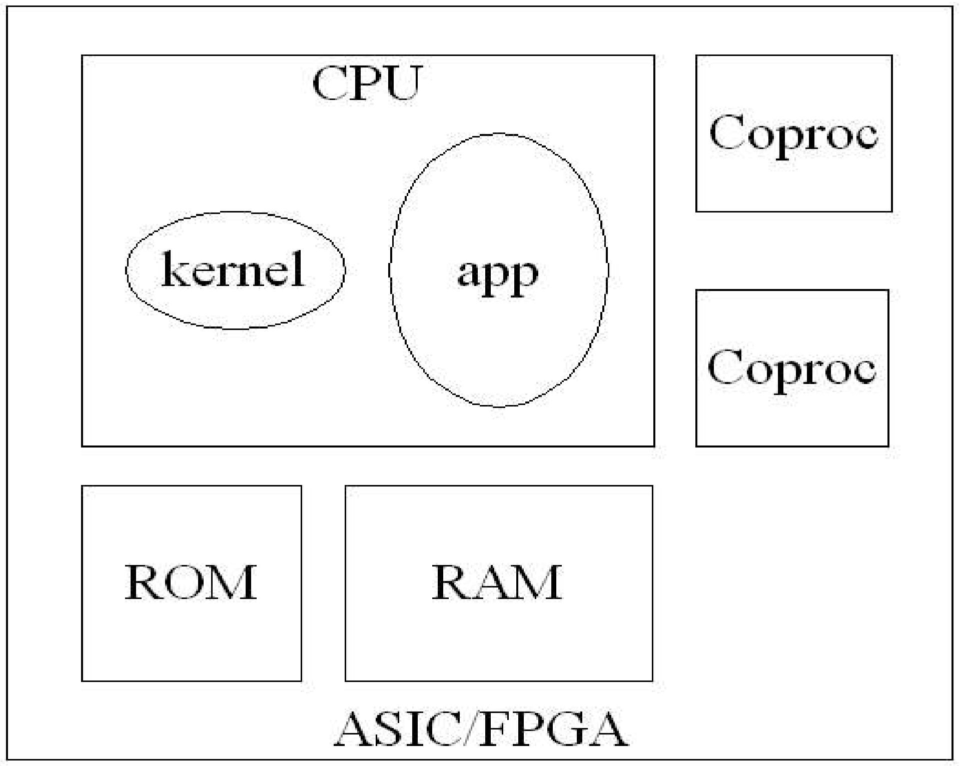
A Hardware/Software Kernel for System on Chip Designs

Andrew Morton and Wayne M. Loucks
Electrical and Computer Engineering
University of Waterloo
Waterloo, Ontario, Canada

*work supported in part by Natural Sciences and Engineering Research Council of Canada and the Graduate Studies Office, University of Waterloo

System on Chip (SoC)



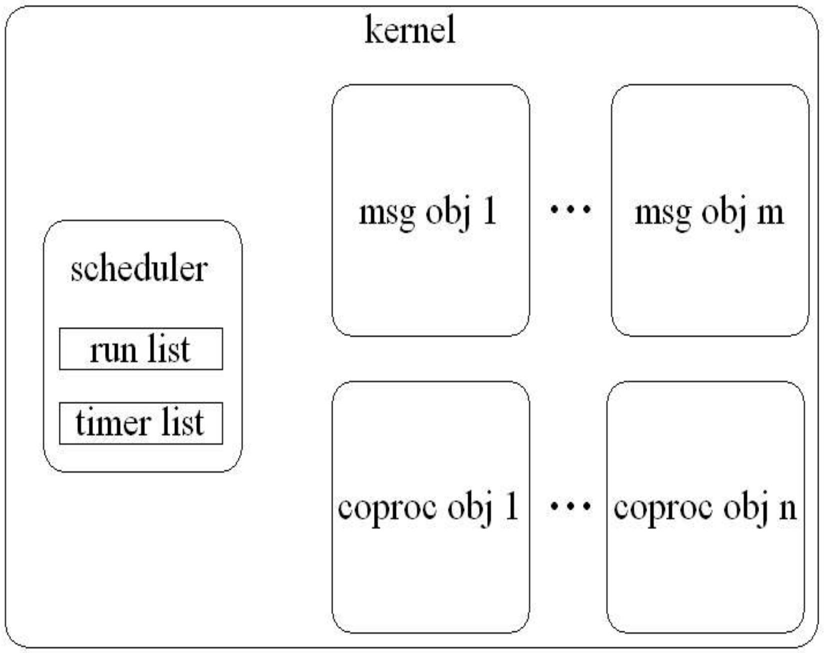
Kernel Coprocessors

- STRON-I, UFμK
  - scheduler*
  - interrupt handler
  - timer
  - semaphore
  - periodic task start

- δ Framework
  - lock cache
  - deadlock detection
  - dynamic memory manager
- 1/2 performance gain vs 1/25 gate count

* fixed-priority preemptive

cs1 Kernel

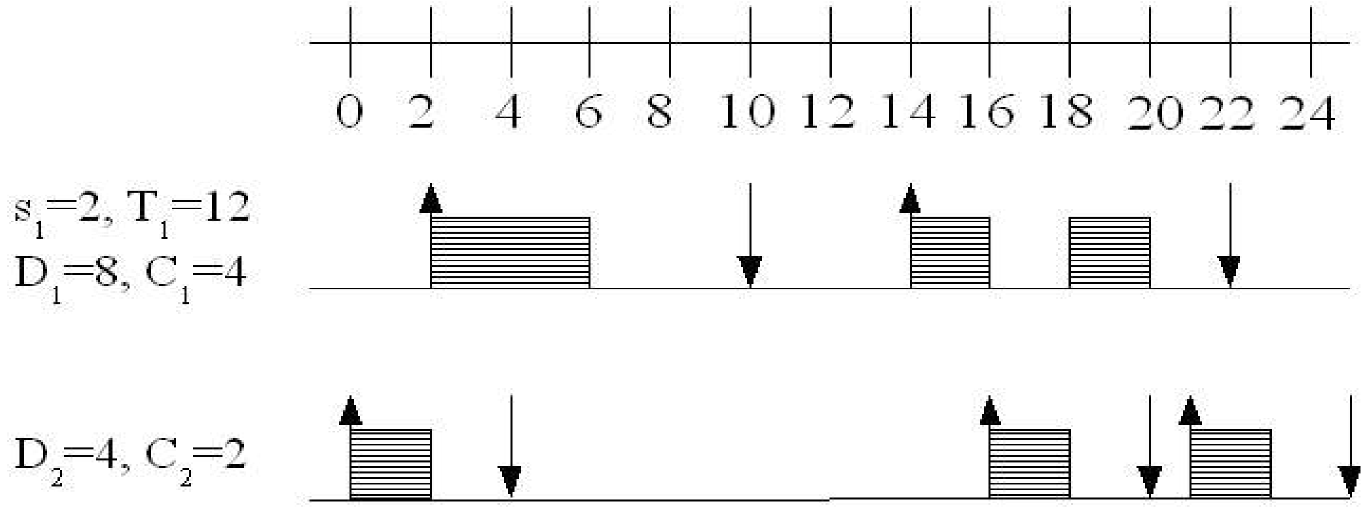


- uni-processor
- EDF scheduler
- message queues
- application coprocessor manager

Preemptive EDF Scheduler

- Earliest Deadline First
  - Of all ready tasks, the task with the earliest deadline is executed first. If another task arrives with an earlier deadline, it will preempt the currently executing task. This is an optimal scheduling policy.
  - Periodic tasks: released at regular interval
    - start(s), period(T), deadline(D)
    - release r = s + mT, m = 0,1,...
    - relative deadline d = r + D
  - Aperiodic tasks: released by event
    - release r = arrival of event
    - relative deadline = r + D

EDF Example

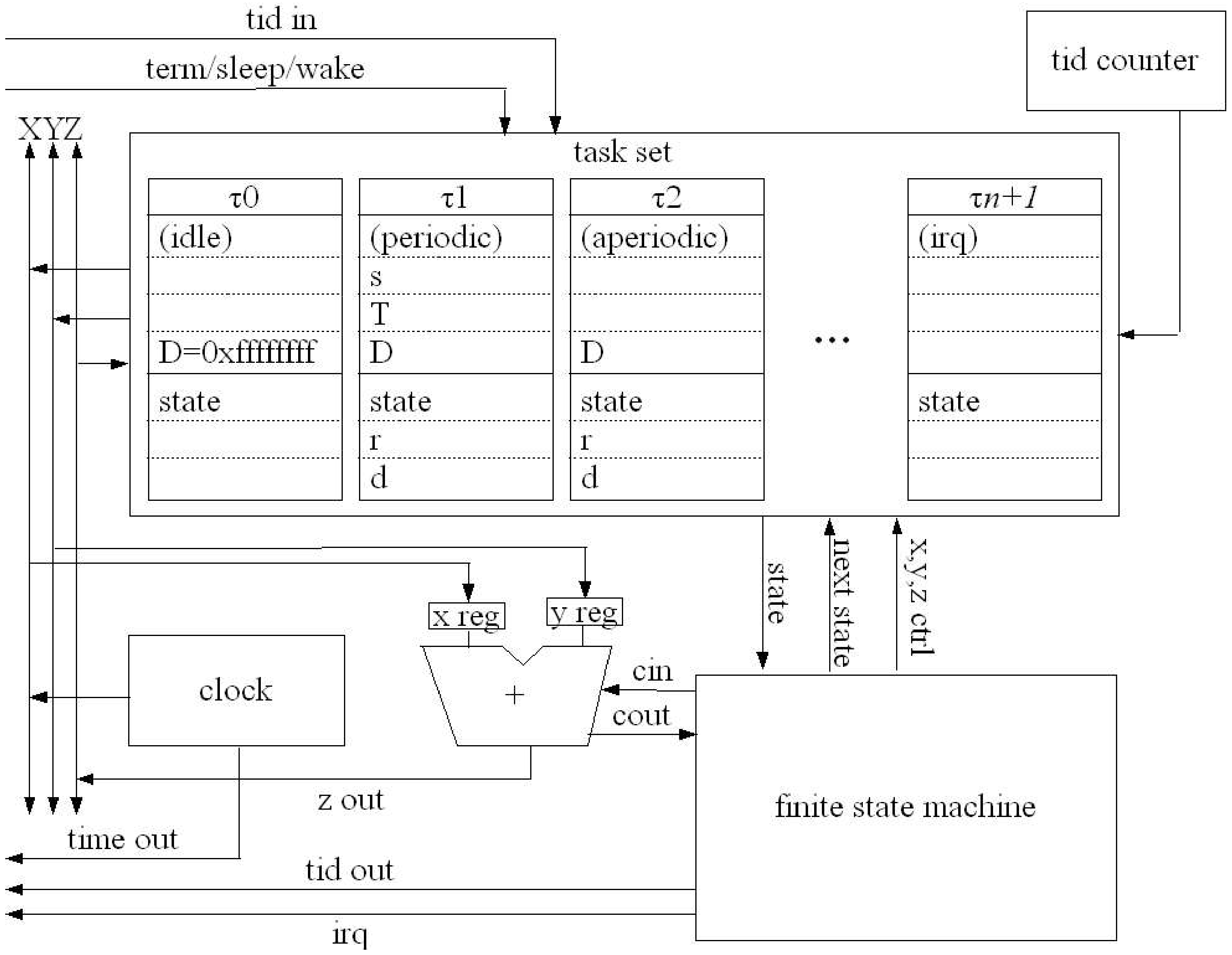


- r_1= 2, 14, 26, ...
- r_2= 0, 16, 21

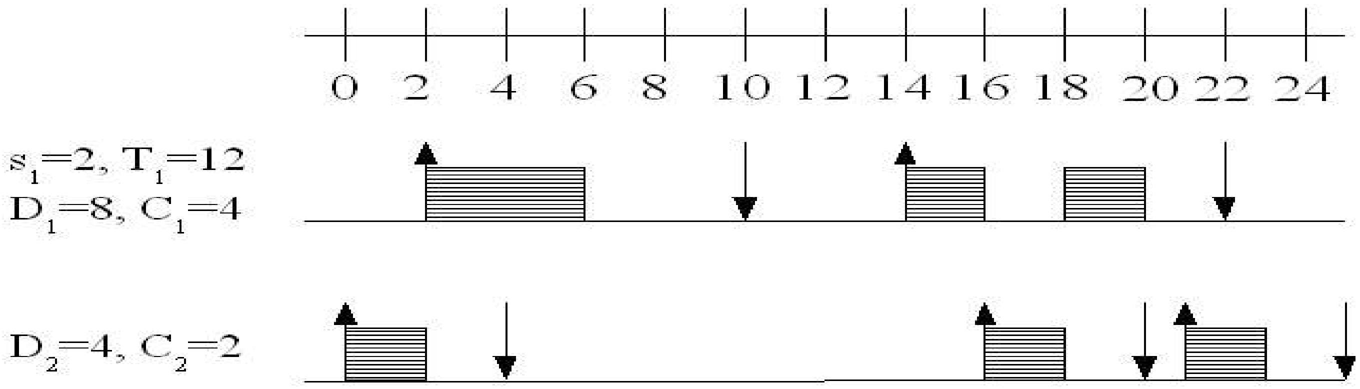
EDF Coprocessor Design

- Coprocessor contains list of tasks with parameters
  - Periodic: start(s), period(T), deadline(D)
  - Aperiodic: deadline(D)
- Ready tasks bid in round-robin order for earliest deadline.
- Idle task initiates bidding with $D=\infty$
- If a task has earlier deadline, it bids
- Task with earliest deadline wins
  - If winning task changes, coprocessor interrupts CPU

Coprocessor Structure



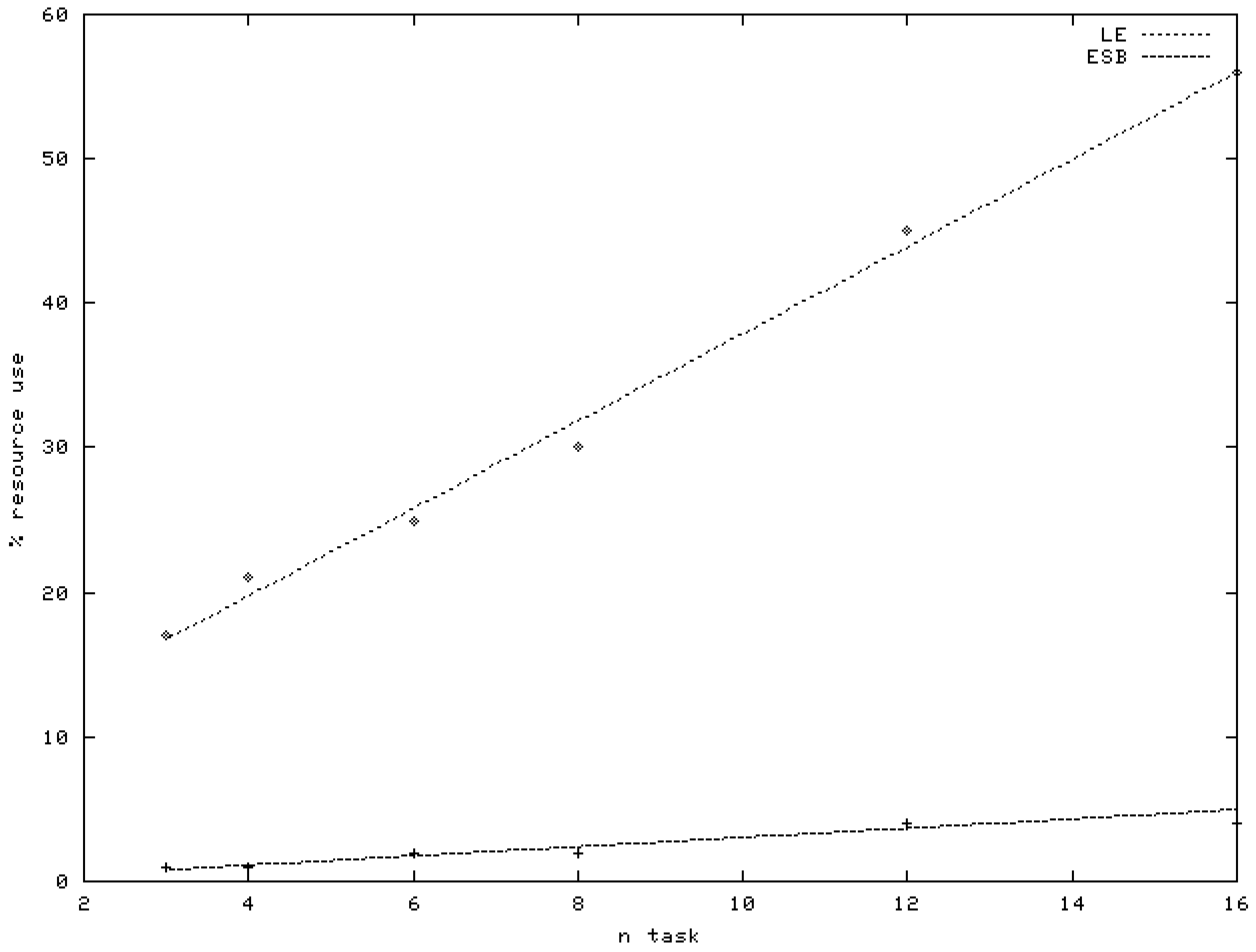
Scheduling Example



		t=0	t=2	t=6	t=8	t=14	t=16	t=18
task 0 : idle D=∞	state vars bid?	alive d=∞ yes	alive d=∞ yes	alive d=∞ yes	alive d=∞ yes	alive d=∞ yes	alive d=∞ yes	alive d=∞ yes
task 1 : periodic s=2,T=12,D=8	state vars bid?	dead r=2 no	alive r=2,d=10 yes	dead r=14 no	dead r=14 no	alive r=14,d=22 yes	alive r=14,d=22 yes	alive r=14,d=22 yes
task 2 : aperiodic D=4	state vars bid?	alive r=0,d=4 yes	dead r=? no	dead r=? no	dead r=? no	dead r=? no	alive r=16,d=20 yes	dead r=? no
task 3 : irq tid=?	tid d action	_ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _

EDF Coprocessor Size

on Altera 20K200E FPGA



EDF Coprocessor Performance

1. Response time (maximum scheduler delay)
2. Processor overhead

	Worst-case Execution Cycles*
cs1	$1166 + 136*\lceil \log_2 n \rceil + 101P + 100*\lceil \log_2 n \rceil *P$
cs2 EDF coproc	1096 $32 + 20n$

* assumes 1 application coprocessor

Performance Example

- Example:
  - n=8, P=3
  - clock=33MHz

1. Response time
   - cs1 (without EDF coprocessor) = 84 µs
   - cs2 (with EDF coprocessor) = 39 µs

Performance Example

2. Processor overhead

- Task processor utilization

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

- Assumption: each task invokes kernel twice (upon release and termination)
- Kernel processor utilization

$$U_{kernel} = 2C_{kernel} * \sum_{i=1}^{n} \frac{1}{T_i}$$

Kernel Coprocessor vs Application Coprocessor

- Benefit-Cost Ratio

$$\frac{\Delta U}{\text{Size}}$$

- Case Study [ML03]

Coprocessor	Δt	ΔU	Size (%)	ΔU/Size
EDF	0.06138μs	0.1270	30	0.4235
cosine	1.337μs	0.2164	46	0.4704

[ML03] A. Morton and W. M Loucks. Real-time kernel support for coprocessors: Empirical study of an SoPC. In *Proceedings of the Embedded Systems and Applications Conference*, pages 10-14, 2003.

Hardware/Software Partitioning

- This kernel partitioning part of overall hardware/software partitioning strategy:
  - Partition the *application and kernel* between hardware and software such that hardware size not violated and application can be feasibly scheduled by the preemptive EDF policy.
  - Application consists of a set of tasks represented by control flow graphs and scheduling parameters.
  - Kernel also represented by control flow graph.
  - ΔU computed differently between application and kernel tasks.

Future Work

- Investigate EDF coprocessor for multi-processor systems.
- Apply hardware/software partitioning to the Java Virtual Machine
  - Deterministic behaviour: garbage collector coprocessor
  - Processor overhead: byte code translation coprocessor