

# Instruction-Set Customization for Real-time Embedded Systems



Huynh Phung Huynh and Tulika Mitra  
School of Computing  
National University of Singapore

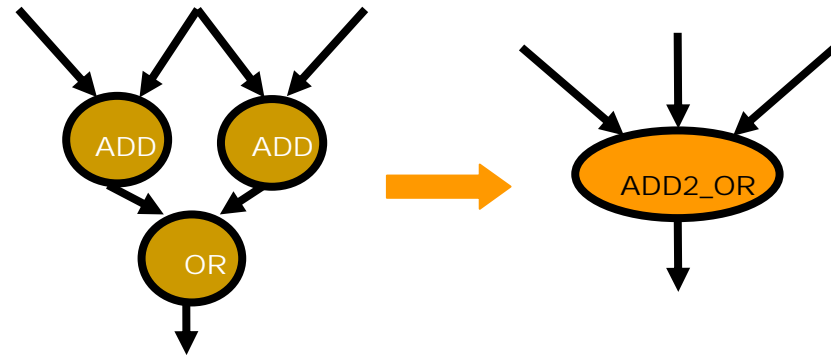
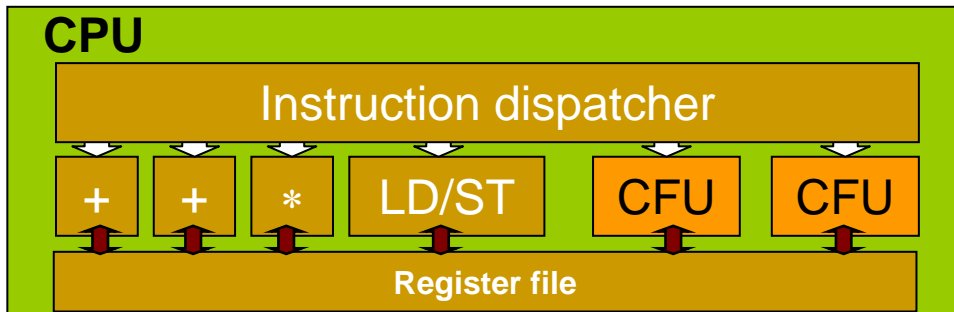
# Outline

---

- Instruction-set customization
- Motivation
- Related work
- Dynamic programming solution for EDF
- Branch and Bound solution for RMS
- Experimental results
- Conclusion

# Instruction-set extensible processors

## □ Typical architecture



## □ Examples of commercially available Instruction-set extensible processors



xtensa



Nios II



S5000

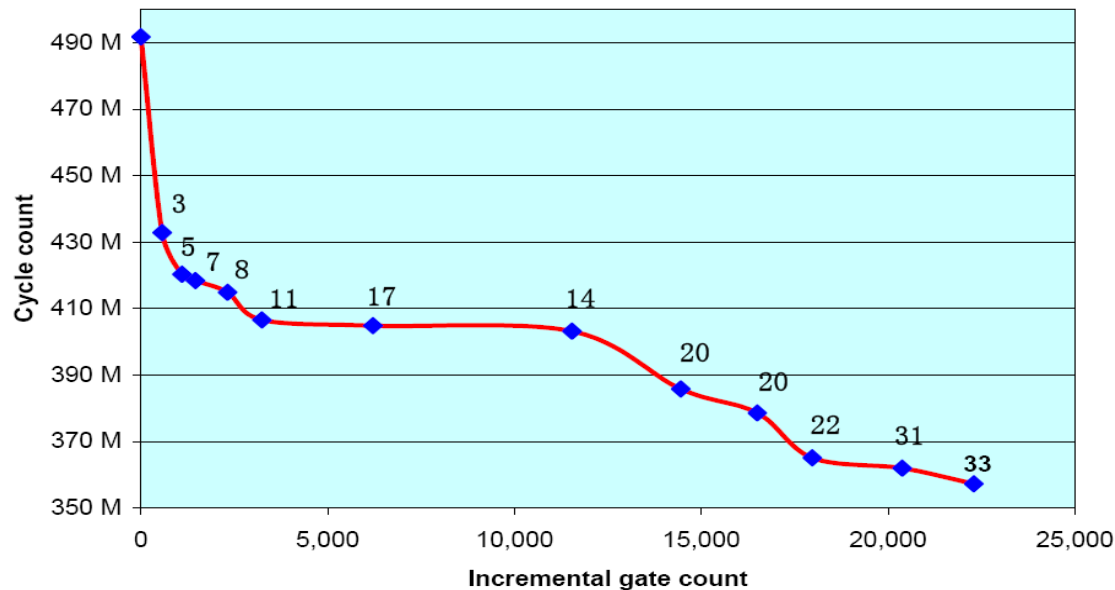
# Motivation

---

- ❑ We explore instruction-set customization in the context of multi-tasking real-time embedded systems.
- ❑ Custom instructions may reduce the processor utilization for a task set which:
  - Enables a task set that was originally unschedulable to become schedulable.
  - Opens up the possibility to execute non-real-time (best-effort) tasks alongside real-time tasks.
  - Can exploit voltage scaling opportunities to lower the operating frequency/voltage of the processor => reduces energy consumption

# Instruction-Set Customization of a Real-time Embedded System

- ❑ Each task has a set of custom instructions configurations different in performance and silicon area tradeoff.



- ❑ **Objective:** Select an appropriate configuration for each task such that the task set is schedulable and the total utilization  $U$  is minimized under RMS or EDF scheduling policy.

# Related Work

---

- ❑ Custom instruction generation: Atasu et al. 2003, N. Clark et al. 2003, P. Yu et al. 2005.
- ❑ Custom instruction selection: Arnold et al. 2001, J. Cong et al. 2004, N. Clark et al. 2003, J. Lee et al. 2002.
- ❑ Customization of task graphs on heterogeneous multiprocessor but does not consider real-time constraints: F. Sun et al. 2005.
- ❑ Customization of a real-time task such that worst -case execution time is minimized: P. Yu et al. 2005

# Dynamic Programming for EDF

---

$U_i(A)$  be the minimum total utilization of tasks  $T_1 \dots T_i$  under an area budget  $A$

$$U_i(A) = \min_{\substack{j=1 \dots n_i \\ area_{i,j} \leq A}} \left( \frac{cycle_{i,j}}{P_i} + U_{i-1}(A - area_{i,j}) \right)$$

$$U_1(A) = \min_{\substack{j=1 \dots n_1 \\ area_{1,j} \leq A}} \left( \frac{cycle_{1,j}}{P_1} \right)$$

# Branch and Bound for RMS

---

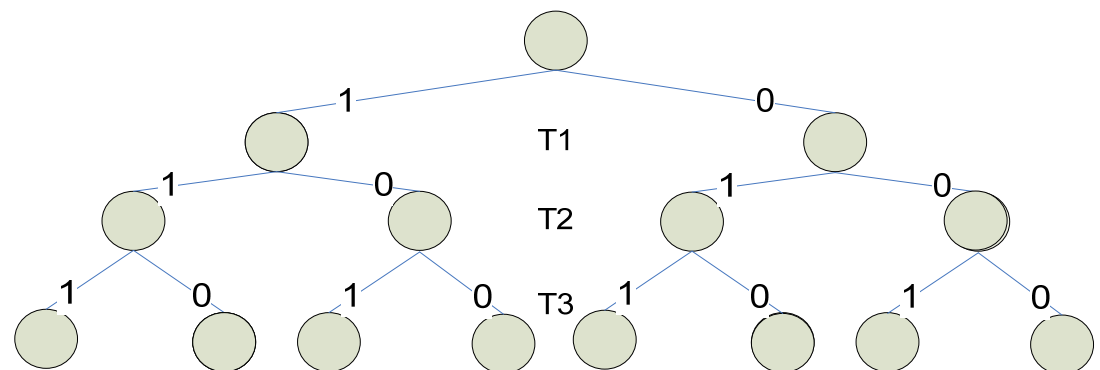
- There exist task sets with  $U \leq 1$  that are not schedulable under RMS
- It is possible to have two configurations  $p$  and  $p'$  such that  $U(p) < U(p')$  but  $p'$  meets all the task deadlines whereas  $p$  does not
- $\Rightarrow$  we can no longer minimize only the total utilization without checking the feasibility of the schedule.
- We use an exact schedulability test for RMS scheduling

# Branch and Bound Search Tree

---

- ❑ Each level  $i$  in the branch-and-bound search tree corresponds to the choice of configuration for the task  $T_i$ .
- ❑ Each node at level  $i$  corresponds to a partial solution which includes the selected configurations of the tasks  $T_1$  to  $T_i$ .

- ❑ Leaf node is a complete solution



# Bounding Function

---

- The minimum utilization achieved so far at any leaf node is saved as a bound MinU.
- The lower bound (LB):

$$LB = \sum_{k=1}^i U_k + \sum_{j=i+1}^N \min U_j$$

$T_k$ : task is enhanced with custom instructions

$T_j$ : task is not enhanced with custom instructions



# Selection Heuristic

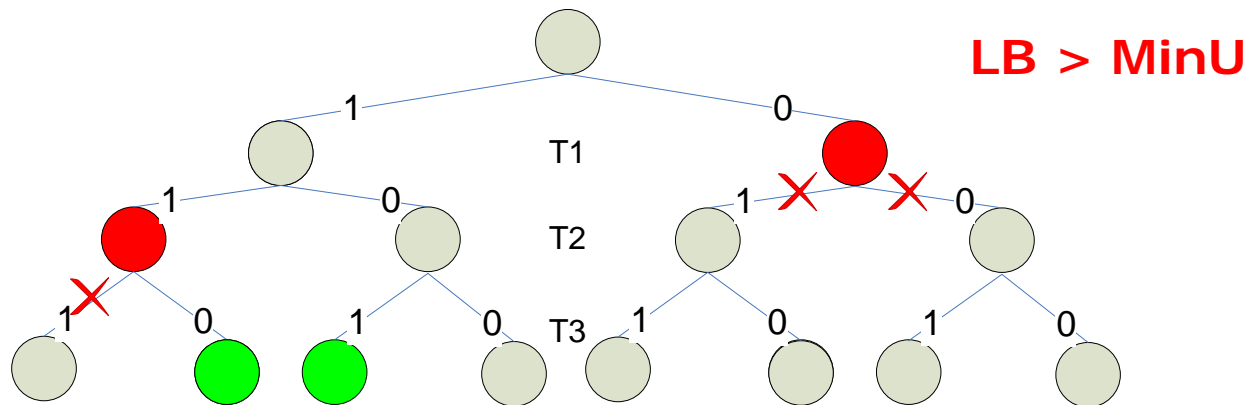
---

- Selecting the custom instruction configuration for each task is in the order of decreasing priority of tasks in RMS scheduling:
  - Any lower priority task cannot preempt the higher priority tasks.
  - At level  $i$  of the search tree we only need to check the schedulability of task  $T_i$ .
- At any level, the configuration with the minimum execution time is considered first.



# Effective Pruning

- ❑ The area constraint is violated at any node.
- ❑ Lower bound  $LB \geq \text{MinU}$

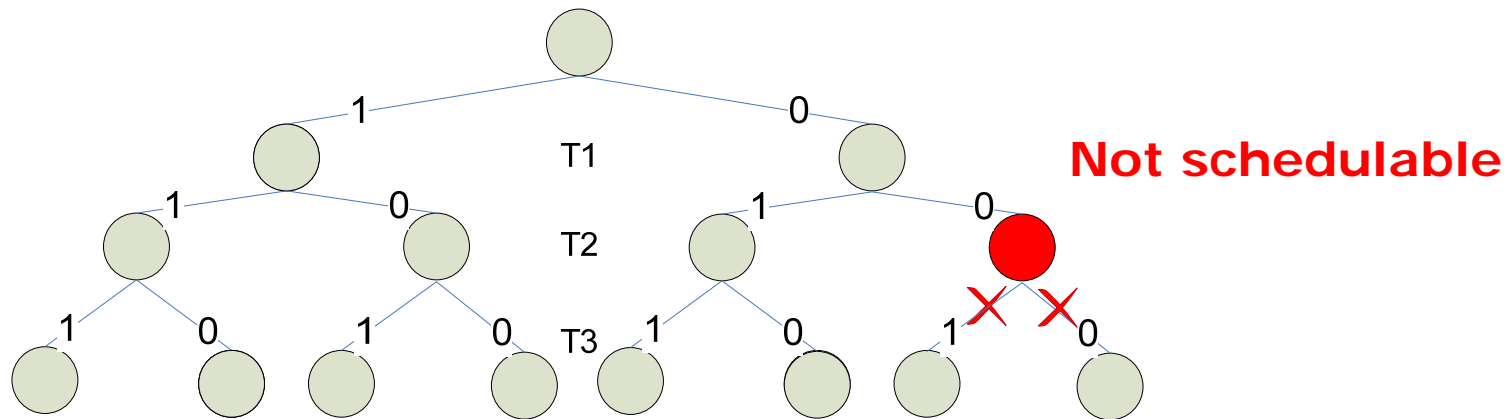


$$LB = 2.5/5 + 2.5/9 + 2/10 = 0.978$$

$$\text{MinU} = 0.889$$

# Effective Pruning

- ❑ The area constraint is violated at any node.
- ❑ Lower bound  $LB \geq \text{MinU}$
- ❑ Task fails to meet its deadline



# Experimental Methodology

---

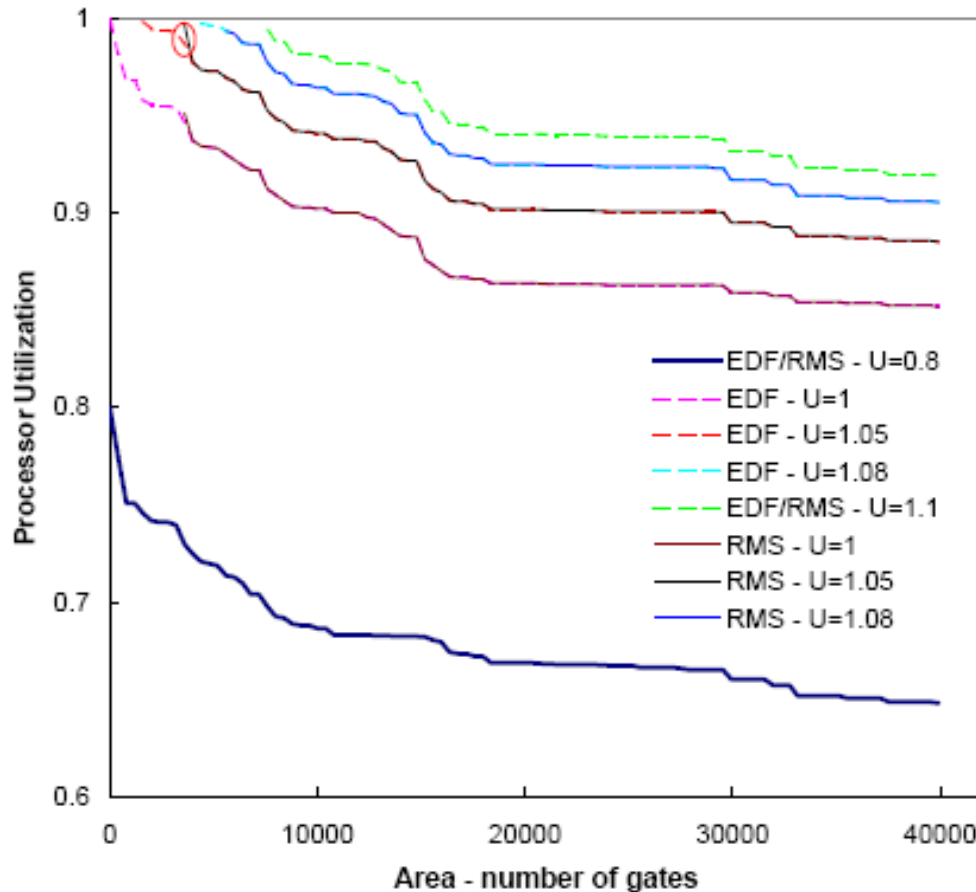
## ❑ 6 task sets

| Task set | Benchmarks                                   |
|----------|--|
| 1        | crc32, sha, jpeg_decoder, blowfish           |
| 2        | blowfish, adpcm_decoder, crc32, jpeg_encoder |
| 3        | adpcm_encoder, blowfish, jpeg_decoder, crc32 |
| 4        | sha, susan, crc32, g721_encoder              |
| 5        | adpcm_decoder, jpeg_decoder, crc32, blowfish |
| 6        | crc32, sha, blowfish, susan                  |

❑ 4 input utilization factors  $U = 0.80, 1.00, 1.05, 1.08$

❑ Hardware area constraint is from 0 to Max\_Area at an interval of  $0.01 * \text{Max\_Area}$  (summation of maximum area required of each task in the task set).

# Utilization versus Area for Task Set 3



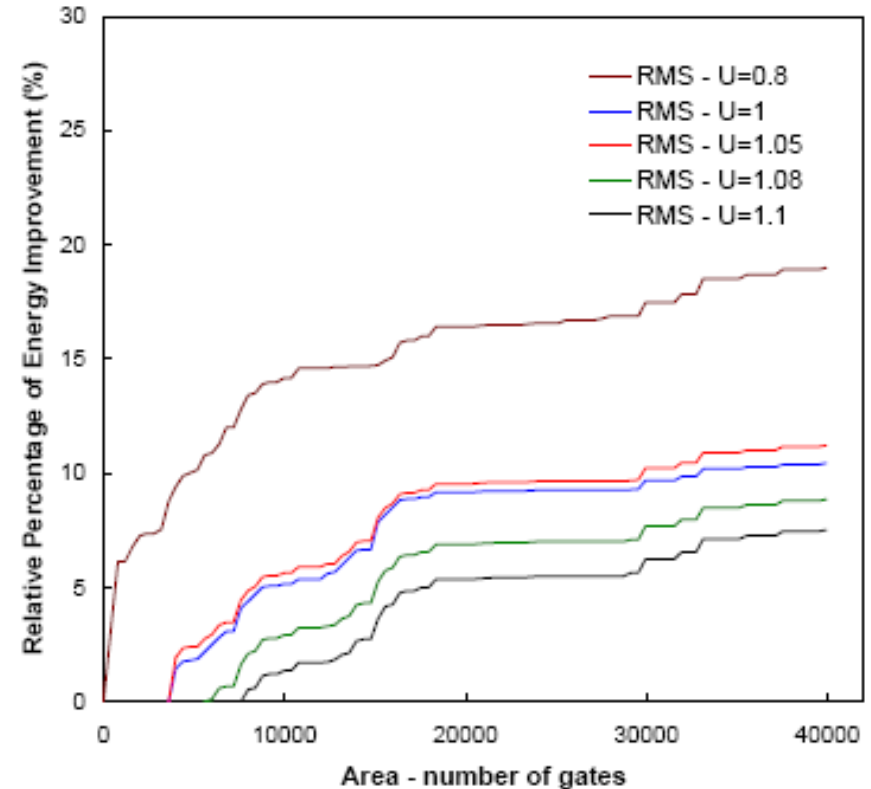
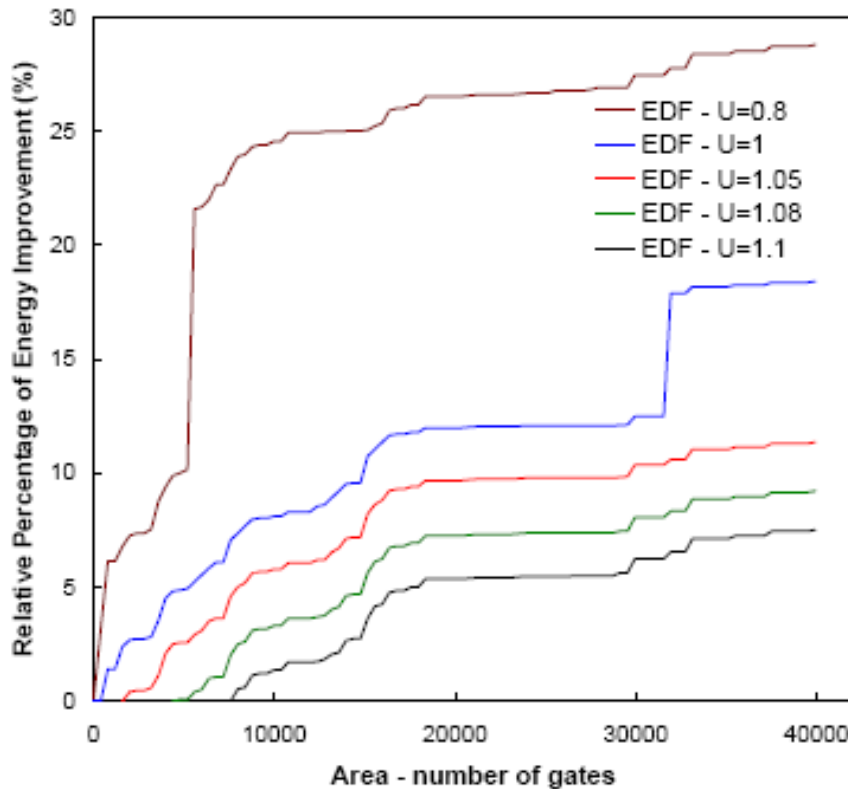
Overall, we get up to 19% reduction in processor utilization. On an average, we get about 14% (13%) reduction in utilization at roughly 75% (50%) of *Max Area* .

# Area versus Energy

---

- ❑ select the optimal customization for the task set under an area constraint.
- ❑ apply static voltage scaling to obtain the lowest operating voltage/frequency.
- ❑ compare the energy consumptions corresponding to these two configurations over the hyper-period of the task set.

# Area versus Energy for Task Set 3



We can obtain up to 30% energy reduction. On an average, the energy reduction is 10% for RMS and 14% for EDF at 75% of Max Area

# Conclusion

---

- ❑ Explore instruction-set customization for multi-tasking real-time embedded systems
- ❑ Propose efficient algorithm to select inter-task optimal customizations under EDF and RMS
- ❑ Experimental results show high reduction in processor utilization and overall energy consumption

# Questions

---

**THANK YOU!**