PARLGRAN: Parallelism granularity selection for scheduling task chains on dynamically reconfigurable architectures *

Sudarshan Banerjee, Elaheh Bozorgzadeh,

Nikil Dutt

Center for Embedded Computer Systems (CECS)

Donald Bren School of Information and Computer Sciences

University of California, Irvine

http://www.cecs.uci.edu/~aces

*Work partially supported by NSF grants CCR-0203813, CCR-0205712

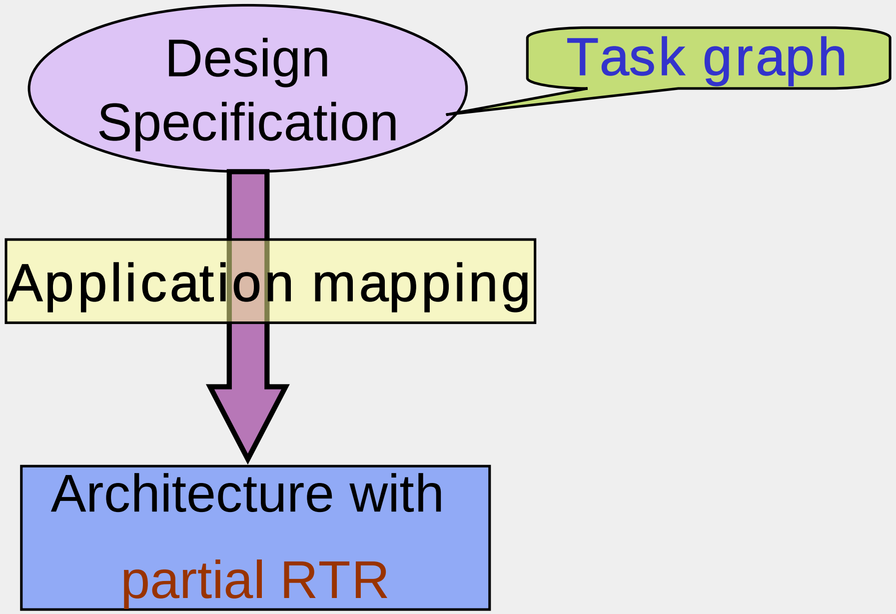
Outline

- Introduction
  - Dynamically reconfigurable architecture
- Problem overview
- Related work
- Detailed problem description
- Approach
- Experiments
- Conclusion

Introduction

- Partial dynamic reconfiguration (RTR)
  - Modify hardware during application execution
  - Commercial example: Xilinx Virtex architecture

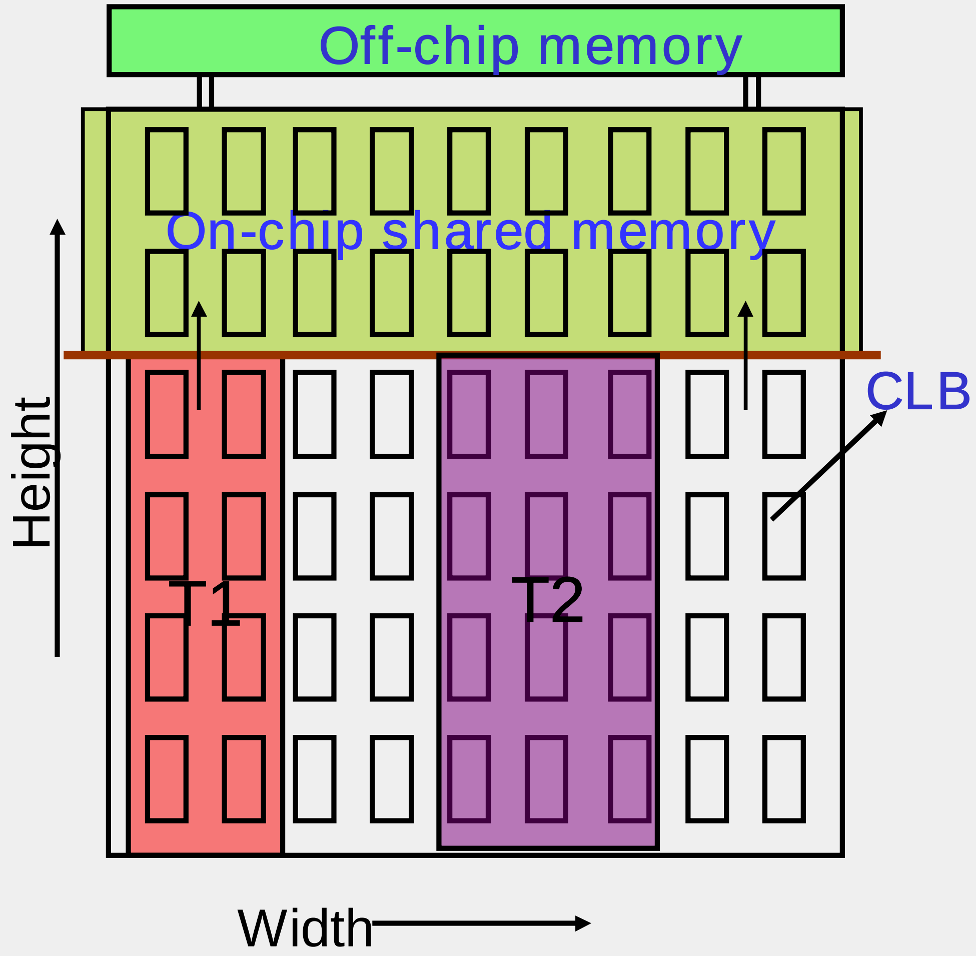
Performance



- Problem space

  Maximize performance under area constraint

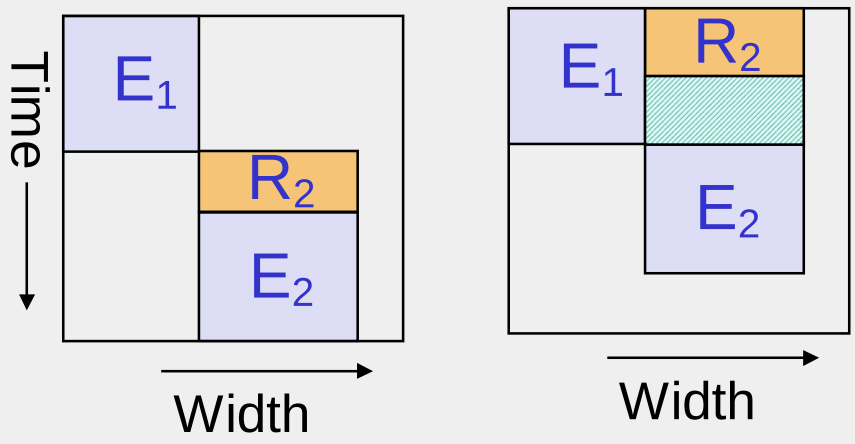
Dynamically reconfigurable architecture



- Single context
  - Significant reconfig delay
- Column-based partial RTR
  - Placement constraints
- Sequential reconfiguration

- Configuration prefetch

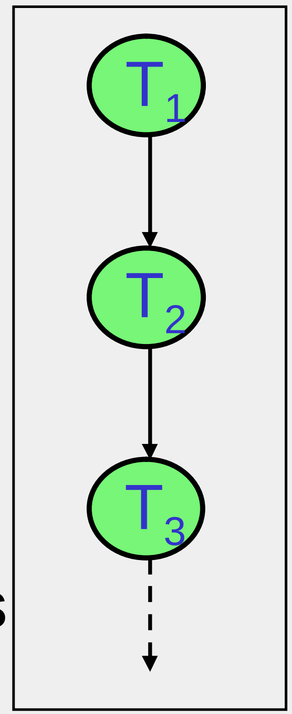
  Hide reconfig delay



Problem overview

- Task chains
  - Common in image processing applications
  - Task execution time predictable
    - Proportional to data volume
  - Key tasks such as DCT are data-parallel
    - Result of task execution on one data block independent of results on another block

Task chain

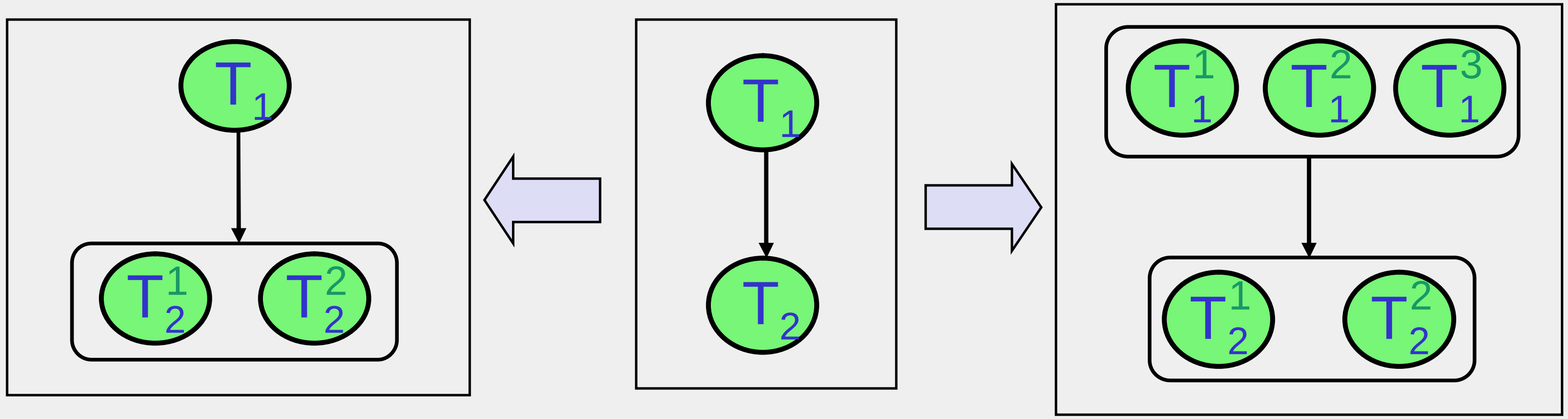


Instantiate multiple copies of data-parallel tasks

- Each copy uses identical HW resources, processes different volumes of data
- Much more scope with partial RTR by reusing space for completed tasks

Problem overview (contd)

Task chain



- Determine number of instances of each task
- Determine workload of each task instance

Granularity

Maximize application performance by selecting parallelism granularity for individual data-parallel tasks

Key challenges: Physical (placement), architectural constraints

Related work

- Work in compiler domain on program parallelization

  ➔ NO consideration of placement, other aspects of partial RTR

- Large body of work on mapping task chains to reconfigurable archi.

  Noguera et al (CODES+ISSS '04), Quinn et al (FCCM '03), …

  ➔ NO partial RTR considerations

  Or, NO placement considerations

- Work on joint scheduling and placement for dependency graphs

  Fekete et al (DATE '01), Yuh et al (ICCAD '04)

  ➔ Theoretical treatment (closer to rectangle-packing)

  NO considerations of prefetch, architectural constraints

  Banerjee et al (DAC '05)

  ➔ Detailed physical + architectural considerations

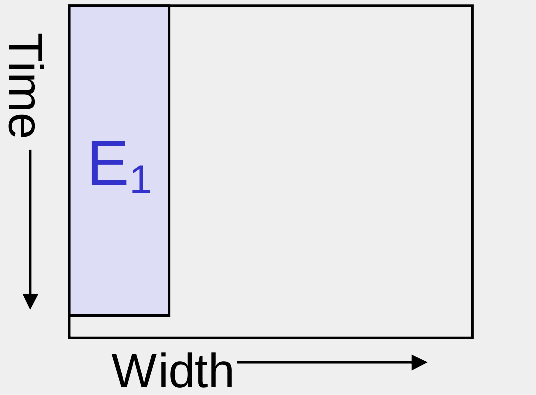
  NO granularity selection

Outline

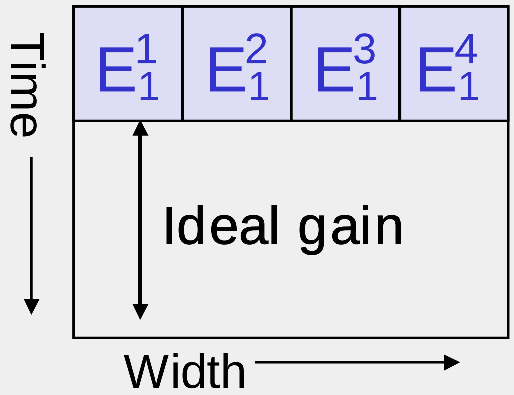
- Introduction
  - Dynamically reconfigurable architecture
- Problem overview
- Related work
- **Detailed problem formulation**
- Approach
- Experiments
- Conclusion

Key issues

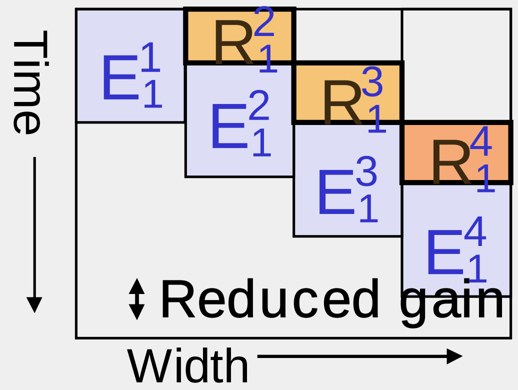
- Reconfiguration overhead
- Load balancing



Sequential execution

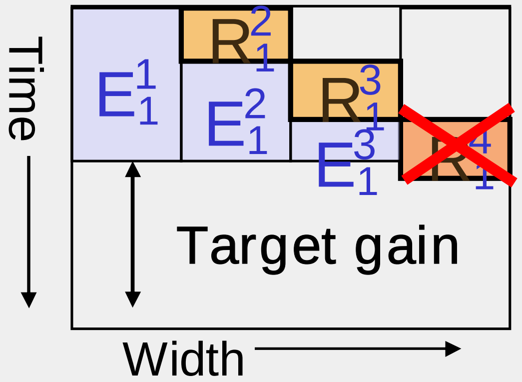


"Ideal" parallel execution



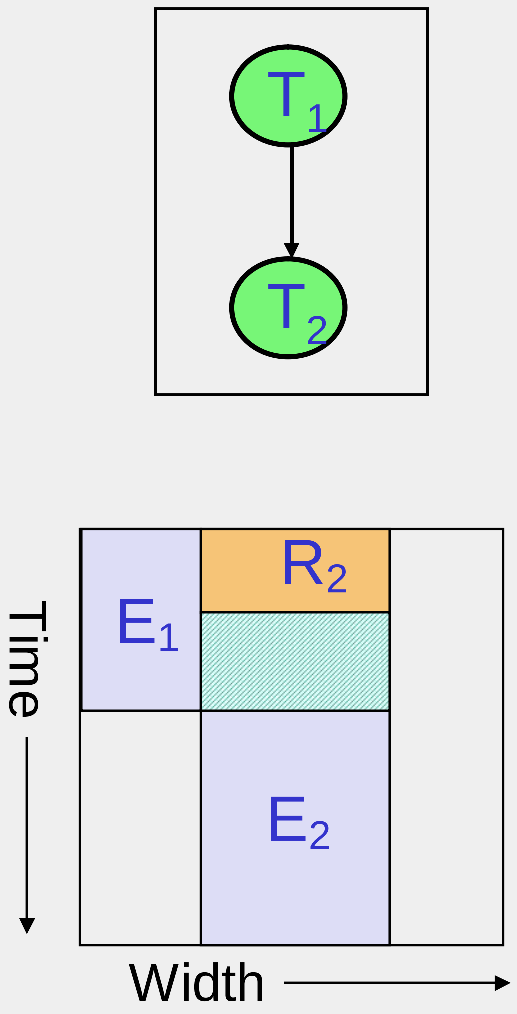
Execution with reconfig overhead

"Load-balanced" Execution

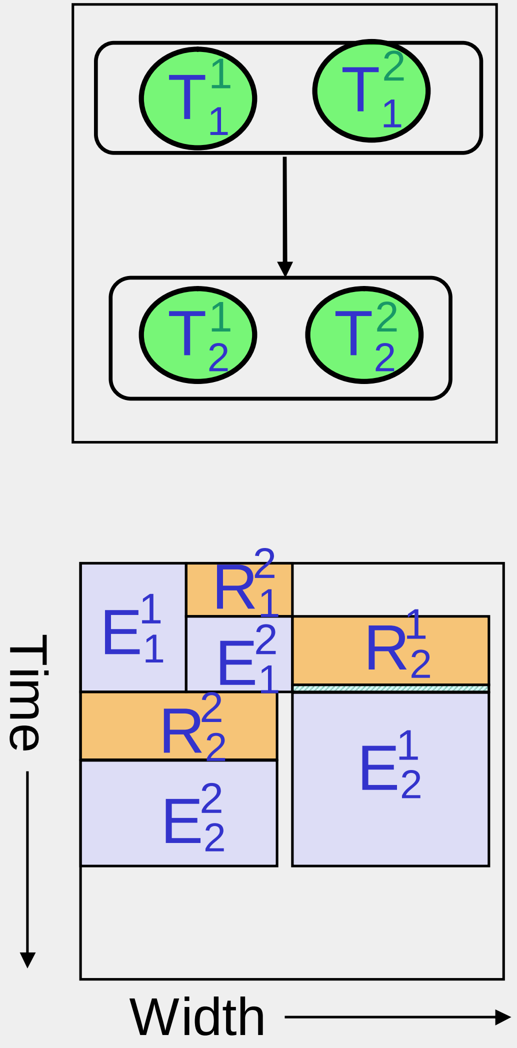


Key issues: precedence constraints

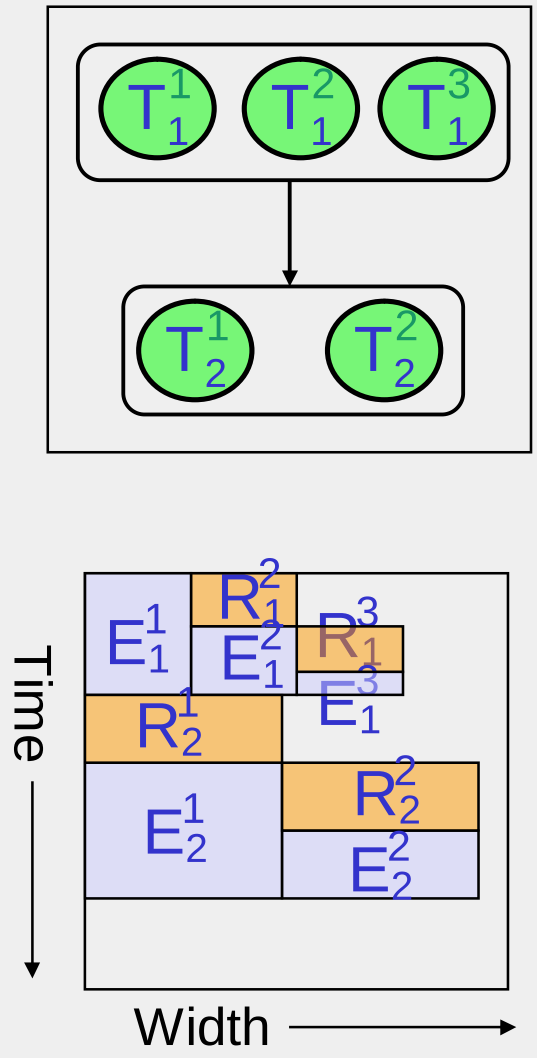
Task chain



Case A



Case B



Detailed problem formulation

Problem inputs:

- Task chain : some tasks are data-parallel
- Hard constraint on area (number of columns)

Objective: Maximize application performance

- Number of instances (copies) of each data-parallel task
- Workload (execution time) of each instance
- Placed schedule for transformed task graph
  - Start time of each task instance
  - Physical location of each task instance

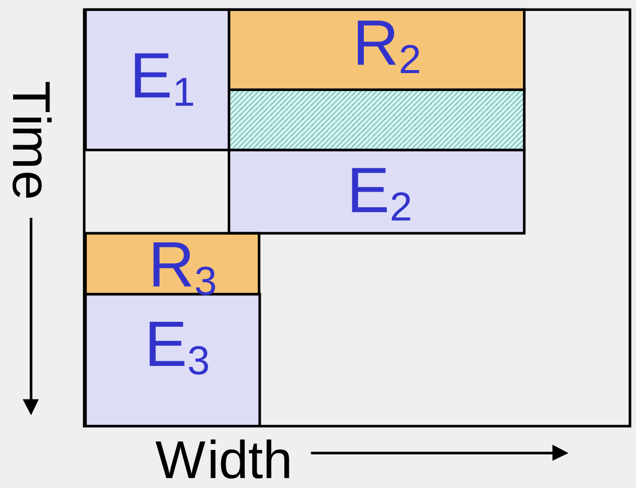
Outline

- Introduction
  - Dynamically reconfigurable architecture
- Problem overview
- Related work
- Detailed problem description
- **Approach**
- Experiments
- Conclusion

Approach

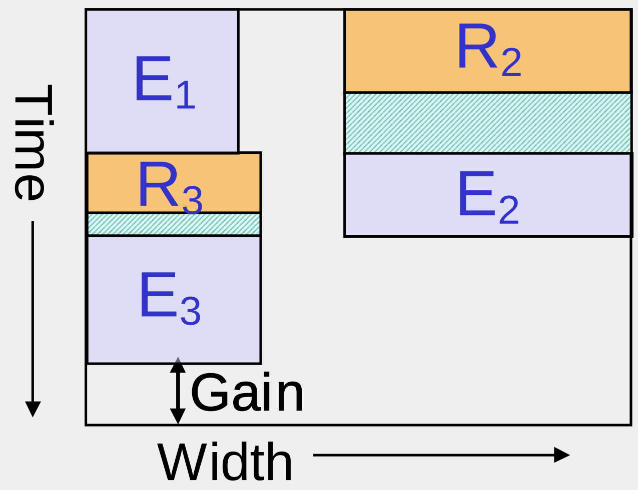
- Detailed analysis of chain-scheduling with partial RTR
  - Joint scheduling and placement of task chain is NP-complete
  - MFF heuristic for task chains (no granularity selection)
- MFF (Modified First-Fit) heuristic
  - Adaptation of FF (first-fit) placement based scheduling for dependency graphs (DAC '05)
  - Simple, local chain-specific optimizations for less fragmentation
- PARLGRAN (granularity selection) heuristic
  - Simple, local optimizations based on MFF principles
  - Select number of instances, Load-balancing

Simple fragmentation reduction



FF: More fragmentation

(First Fit)



MFF: Less fragmentation

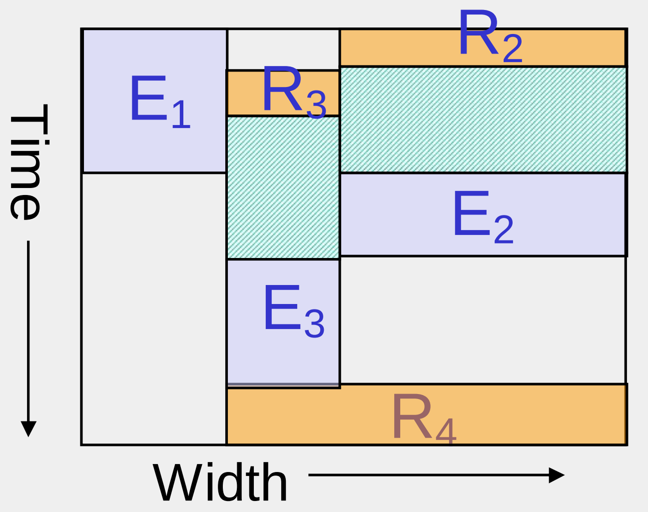
(Modified First Fit)

E_i Execute Task T_i

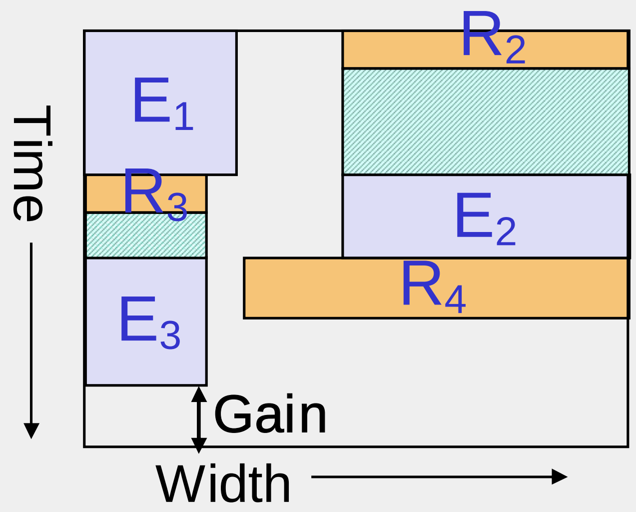
R_i Reconfig process for Task T_i

Task T_i ready, in wait state

Exploiting slack in reconfiguration



More fragmentation



Less fragmentation

- E_i: Execute Task T_i
- R_i: Reconfig process for Task T_i
- Task T_i ready, in wait state

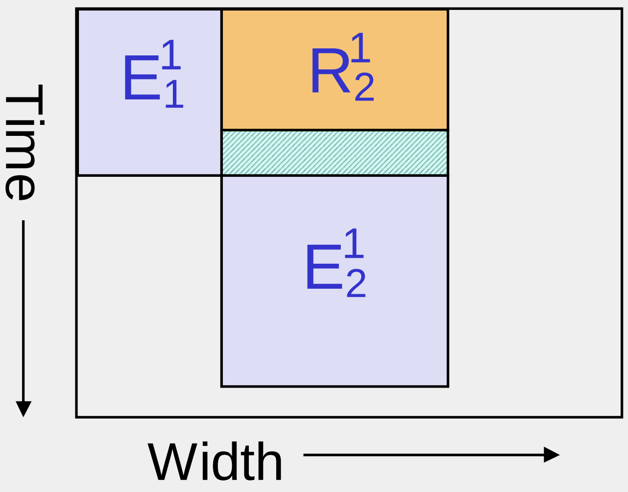
PARLGRAN

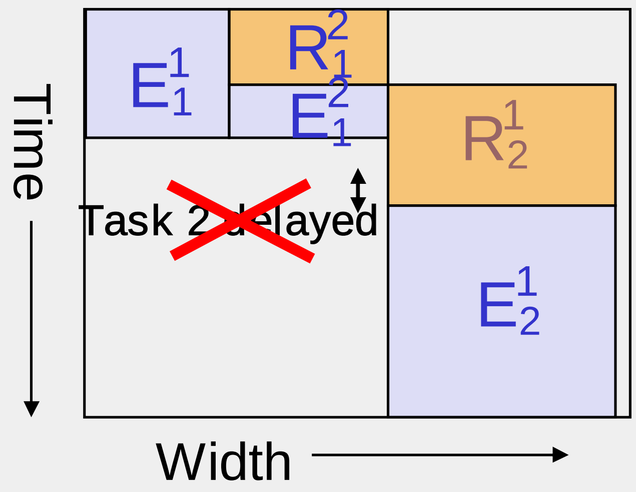
- Chain-scheduling (MFF) provides insight
  - Local optimization helps improve performance
- Heuristic execution time comparable to task execution
  - Application in semi-online scenario

    Semi-online: Key information available only at run-time
    Task execution time (data size), area constraint

- Simple, greedy approach
  - Attempt to improve solution quality locally
- Heuristic outline
  - Static pruning
  - Dynamic granularity selection

PARLGRAN: Static pruning

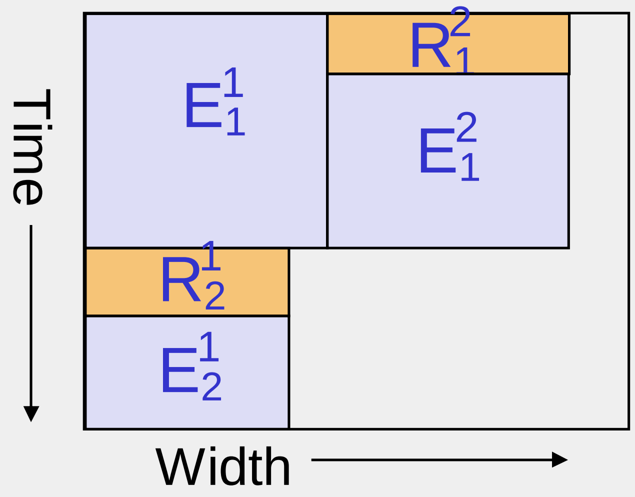




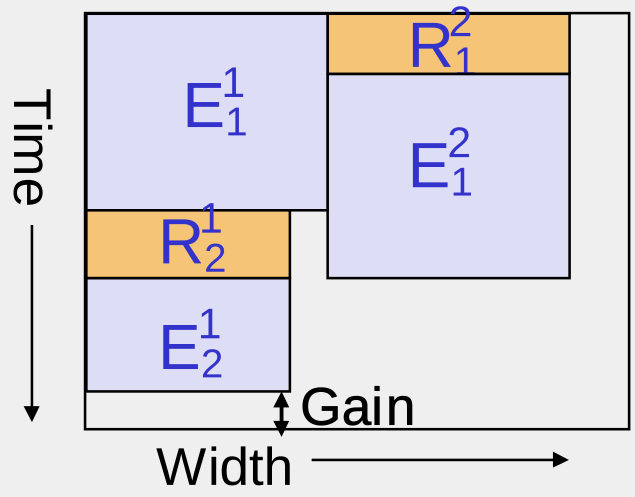
Static

- Pruning based only on timing considerations
- No placement considerations

PARLGRAN: Load balancing



Identical finish times for task copies



Different finish times for task copies

PARLGRAN

For each task T_i

- Determine earliest execution start time

  (consider placement, reconfiguration mechanism)

- While (no degradation in start time)

  1. Add new instance of parent task

     (assign physical location, start time)

  2. Adjust workload (load balancing) of existing instances of parent task

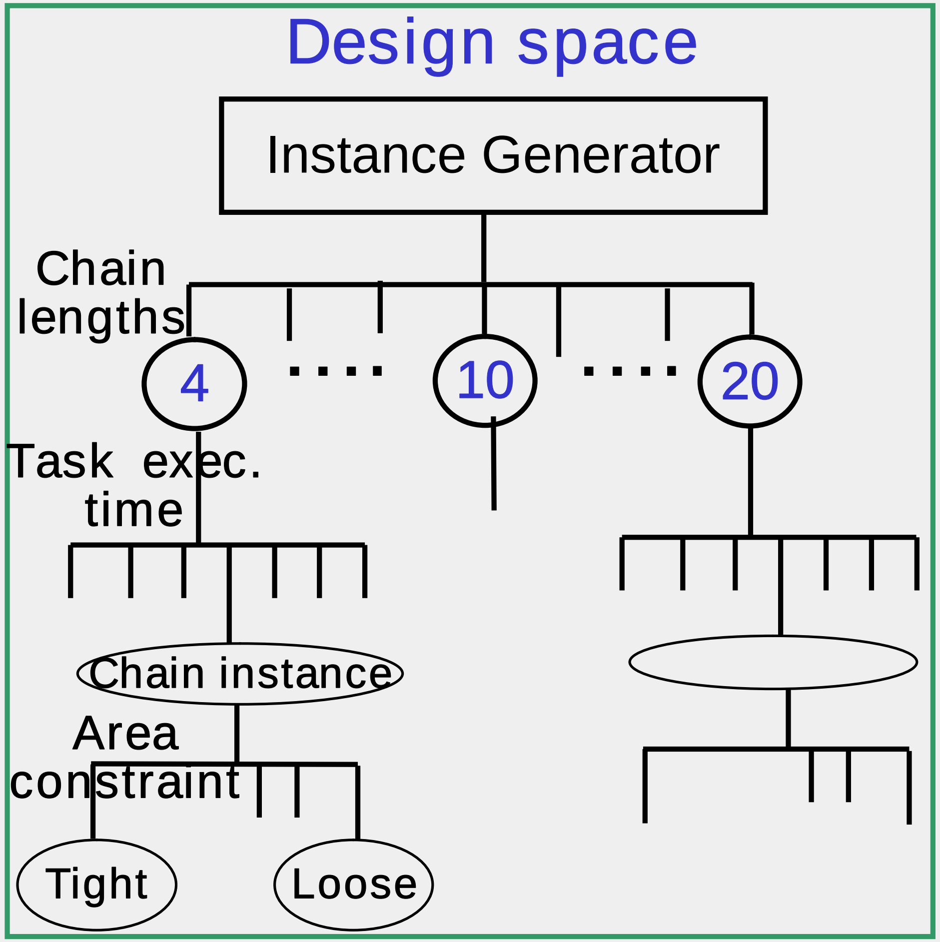
- Apply local optimizations (from MFF) to improve schedule

Outline

- Introduction
  - Dynamically reconfigurable architecture
- Problem overview
- Related work
- Detailed problem description
- Approach
- **Experiments**
- Conclusion

Experimental Setup

- Large set of synthetic benchmarks
  - Individual task data obtained from constrained (placement, routing) synthesis on XC2V2000
  - Varying chain length
  - Varying task execution time
  - Varying area constraints



- Application case study
  - JPEG encoding

Experiments

- Heuristic quality of MFF (chain-scheduling)
  - Compare with FF (first-fit based approach, DAC '05)
- Heuristic quality of PARLGRAN (granularity selection)
  - Compare with FF
  - Compare with MAXPARL

    MAXPARL: maximum parallelization in available area

    (fixed granularity DAG, scheduled with configuration prefetch)
- Application case study of JPEG encoding
  - Compare schedule length of PARLGRAN with MFF, MAXPARL
- Estimated run-time of PARLGRAN

Heuristic quality: MFF Vs FF

- MFF better in 21% tests (243/1140)
- MFF worse in 0.4% tests (5/1140)

- Worst case for MFF:

  Negligible increase in schedule length (0.44%)

- Good cases for MFF:

  10% tests, FF schedule length longer by 3 %

- MFF, FF quality similar on long chains, loose area constraint

- MFF frequently generates better schedules on short chains, tight area constraint

Experiments

- Heuristic quality of MFF (chain-scheduling)
  - Compare with FF (first-fit based approach, DAC '05)
- Heuristic quality of PARLGRAN (granularity selection)
  - Compare with FF
  - Compare with MAXPARL

    MAXPARL: maximum parallelization in available area

    (fixed granularity DAG, scheduled with configuration prefetch)
- Application case study of JPEG encoding
  - Compare schedule length of PARLGRAN with MFF, MAXPARL
- Estimated run-time of PARLGRAN

Heuristic quality: PARLGRAN Vs FF

Quality = $(T_{FF} – T_{parl})/T_{FF} * 100$

Chain length	Average gain
4-7	46.3%
8-11	51.7%
12-15	55.0%
16-20	58.3%
Average gain	> 50%

Even with high reconfiguration overhead, significant benefits from exploiting data-parallelism

T_{FF} Schedule length generated by FF (first-fit)

T_{parl} Schedule length generated by PARLGRAN

Heuristic quality: PARLGRAN Vs MAXPARL

Quality = $(T_{max} – T_{parl})/T_{max} * 100$

Chain length	Average	Best	Worst
4-7	9.8%	142.5%	-49.6%
8-11	15.8%	109.6%	-30.9%
12-15	18.5%	82.3%	-15.5%
16-20	33.8%	151%	-17.5%
Avg gain	> 15%		

PARLGRAN much better than "static parallelization" as chain length increases

T_{max} Schedule length generated by MAXPARL (maximum parallelization in available area)

Case study on JPEG encoding

- Tasks synthesized under placement, routing constraints on XC2V2000
- Aggregate task area = 11 columns

Test	Area constraint	T_{mff} (ms)	T_{max} (ms)	T_{parl} (ms)
256 X 256 JPG	5	12.71	12.73	12.36
	6	11.24	12.52	10.81
	7	11.24	11.38	10.05
	8	11.24	12.11	9.08
	9	10.10	12.79	9.08
512 X 512 JPG	5	42.86	40.68	40.30
	6	41.34	35.32	35.13
	7	41.34	34.18	34.37
	8	41.34	29.08	28.60
	9	40.20	28.38	27.71

Estimated run- time of PARLGRAN

- Preliminary estimate on PowerPC processor @400 MHz

  (available on Xilinx Virtex-II Pro platform)

- Estimated run-time: 3-4 ms
  - Large experiment: 12 tasks, 20 columns

- DCT execution time: ~11 ms
  - 512 X 512 colour image

- PARLGRAN suitable for semi-online scenarios

> Semi-online:
>
> Task execution time, area constraint available only at run-time

Conclusion

Contribution

- Approach to select data-parallelism granularity for task chains on dynamically reconfigurable architectures with partial RTR
- Determines number of instances of data-parallel tasks, AND, execution time (workload) of each instance
- Integrated in a joint scheduling, placement formulation
  - Physical location, reconfiguration start time, execution start time for each task instance
- Large set of synthetic experiments + JPEG encoding case study demonstrate heuristic quality

Future work

- Communications bandwidth, memory issues
- Power, energy considerations
- Extend heuristic for DAGs (directed acyclic graphs)

Thank You !

Questions/Comments?

E-mail: banerjee@uci.edu