

Resource-Efficient Scheduling for Partially-Reconfigurable FPGA- based Systems



POLITECNICO
MILANO 1863

Andrea Purgato: andrea.purgato@mail.polimi.it

Davide Tantillo: davide.tantillo@mail.polimi.it

Marco Rabozzi: marco.rabozzi@polimi.it

Donatella Sciuto: donatella.sciuto@polimi.it

Marco D. Santambrogio: marco.santambrogio@polimi.it

Objectives

Given:

- a taskgraph representing an application.
- a heterogeneous board, with a reconfigurable logic part (FPGA) and homogeneous processors.

Provide a **Mapping and Scheduling** of the tasks considering:

- resources availability of FPGA.
- dependences among the tasks.
- partial reconfiguration constraints.

Minimize the execution time of the schedule.

Outline

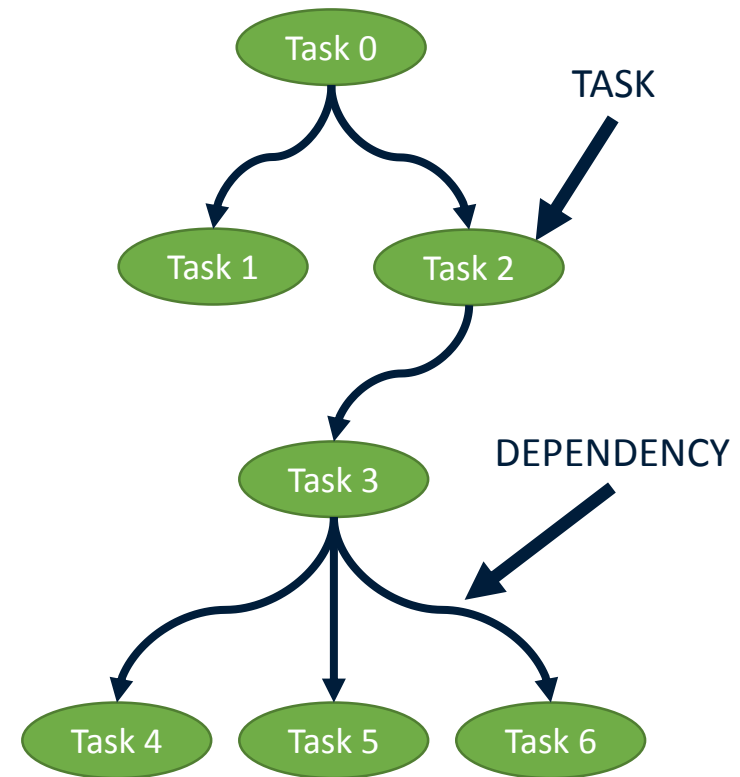
1. Problem Description
2. State-of-the-art
3. Implementation
4. Results Analysis
5. Conclusions and Future Work

Problem Description (1/3)

The description of the application is given as a **Taskgraph**.

It is a **Direct Acyclic Graph (DAG)** describing:

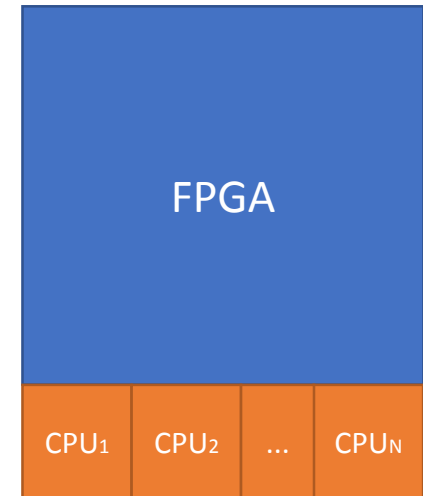
- the functionality of the program, through **Tasks**.
- and **dependency** among them.



Problem Description (2/3)

The target architecture is a **heterogeneous board** composed of:

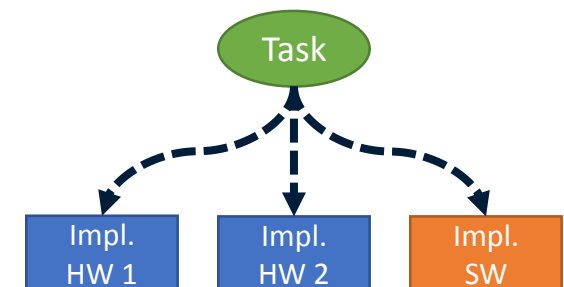
- A reconfigurable logic part (FPGA).
- Homogeneous processors.



A task may be executed on:

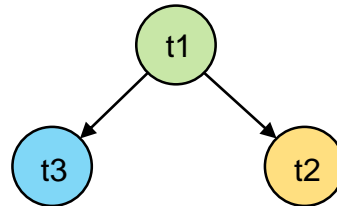
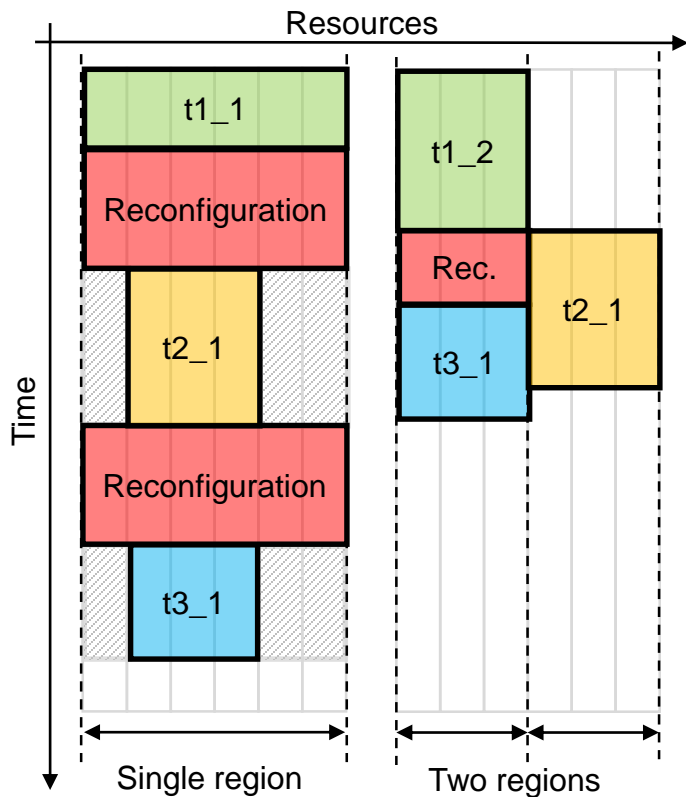
- the FPGA logic (**HW**).
- a processor on the board (**SW**).

Each task can have both HW and SW implementations.



Problem Description (3/3)

The **selection** of the implementation for each task can change the final solution.



| implem. | time | resources |
|---------|------|-----------|
| t1_1 | 2 | 6 |
| t1_2 | 5 | 3 |
| t2_1 | 4 | 3 |
| t3_1 | 3 | 3 |

HW Implementations with large resources requirements:

- Generally faster.
- Higher reconfiguration time.

HW Implementations with small resources requirements:

- Generally slower.
- Faster to reconfigure.

State-of-the-art

| Authors | Partial Reconfiguration aware | Explicit communication handling | Multiobjective optimization | Multi-resources Floorplan Validation | Tunable performance |
|---------------------|-------------------------------|---------------------------------|-----------------------------|--------------------------------------|---------------------|
| Cattaneo et al. [1] | Yes | Yes | No | No | No |
| Deiana et al. [2] | Yes | No | Yes | Yes | Yes |
| Redaelli et al. [3] | Yes | Yes | No | No | No |
| Fekete et al. [4] | No | Yes | No | No | No |
| Proposed Approach | Yes | No | No | Yes | Yes |

[1] R. Cattaneo, R. Bellini, G. Durelli, C. Pilato, M. Santambrogio, and D. Sciuto, "Para-sched: A reconfiguration-aware scheduler for reconfigurable architectures," in Parallel Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International, May 2014, pp. 243–250.

[2] E. Deiana, M. Rabozzi, R. Cattaneo, and M. Santambrogio, "A multiobjective reconfiguration-aware scheduler for fpga-based eterogeneous architectures," in ReConFigurable Computing and FPGAs (ReConFig), 2015 International Conference on. IEEE, 2015.

[3] F. Redaelli, M. D. Santambrogio, and S. O. Memik, "An ilp formulation for the task graph scheduling problem tailored to bi-dimensional reconfigurable architectures," Int. J. Reconfig. Comput., vol. 2009, pp. 7:1–7:12, Jan. 2009.

[4] S. Fekete, E. Kohler, and J. Teich, "Optimal fpga module placement with temporal precedence constraints," in Design, Automation and Test in Europe, 2001. Conference and Exhibition 2001. Proceedings, 2001, pp. 658–665.

Proposed Approach

We propose two different approaches:

Deterministic Approach

Randomized Approach

Both present:


- Efficient use of the available resources.
- Multi-resources floorplan validation.

- Single iteration.
- Low execution time.
- Deterministic solution.

- Multiple iterations.
- Fixed execution time.
- Improved solution.

Algorithm Overview

The algorithm is composed by **eight** different steps:

1. **Implementation selection**
 2. **Critical path extraction**
 3. **Regions definition**
 4. **Software tasks balancing**
 5. **Start and end time computation**
 6. **Software tasks mapping**
 7. **Reconfigurations scheduling**
 8. **Feasibility check**
- 

1. Implementation Selection

Uses a cost index to compare HW implementations of a task.

$$cost_i = \frac{\sum_{r \in R} weightRes_r * res_{i,r}}{\sum_{r' \in R} weightRes_{r'} * maxRes_{r'}} + \frac{time_i}{maxT}$$

$$weightRes_r = 1 - \frac{maxRes_r}{\sum_{r' \in R} maxRes_{r'}}$$

$$maxT = \sum_{t \in T} \min_{i \in I_t} time_i$$

- R : set of available resources.
- T : tasks in the taskgraph.
- I : set of implementations.
- I_t : set of implementations for task $t \in T$.
- $res_{i,r}$: resources of type $r \in R$ required by HW implementation $i \in I$.
- $maxRes_r$: number of resources of type $r \in R$ available on the FPGA.
- $time_i$: execution time of implementation $i \in I$.

- More importance is given to scarce resources.
- High cost is given to implementations having high execution time or high resource usage.

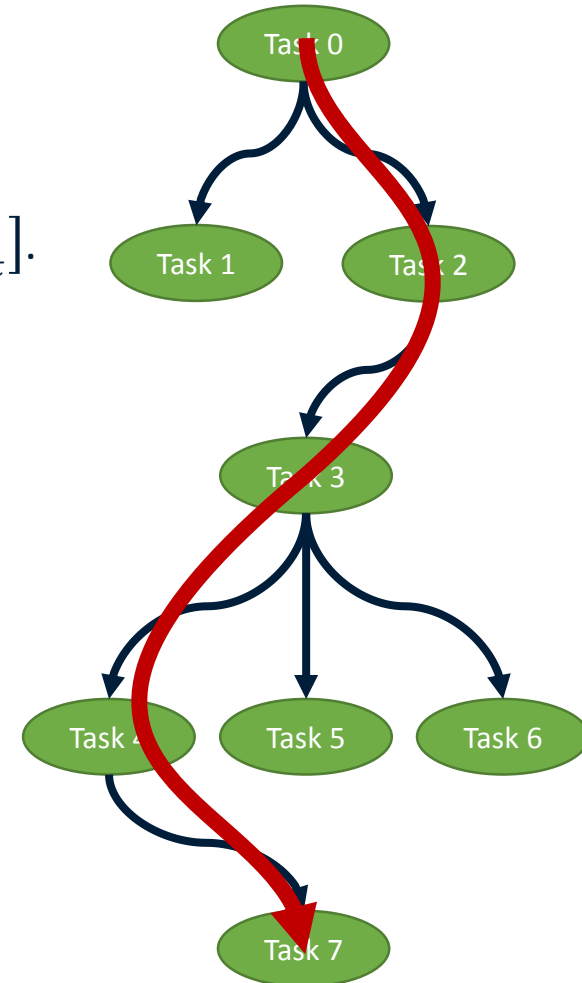
For each task, chose an implementation with minimum execution time among:

- the available SW implementations.
- the available HW implementations with lowest cost.

2. Critical Path Extraction

Method Used: Critical Path Method (CPM).

- Generates a time interval for each task: $w_t = [T_{MIN_t}, T_{MAX_t}]$.
- Each task should be executed in its interval to avoid delay.
- Tasks in the critical path are labeled as **Critical**.



3. Regions Definitions

Defines the reconfigurable areas for the heterogonous board used.

In this phase, an **efficiency index** is defined for each HW implementation $i \in I$.

$$eff_i = \frac{time_i}{\sum_{r \in R} res_{i,r} * weightRes_r}$$

Critical tasks are processed before non-critical tasks. All the tasks are ordered with respect to the **efficiency index**.

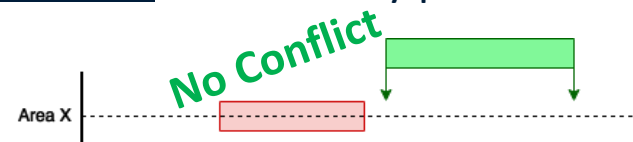
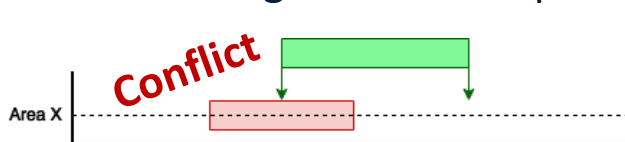
Critical Tasks:

1. Check if existing area can map the task.
2. Create a new area.
3. Switch to SW implementation.

Non-Critical Tasks:

1. Create a new area.
2. Check if existing area can map the task.
3. Switch to SW implementation.

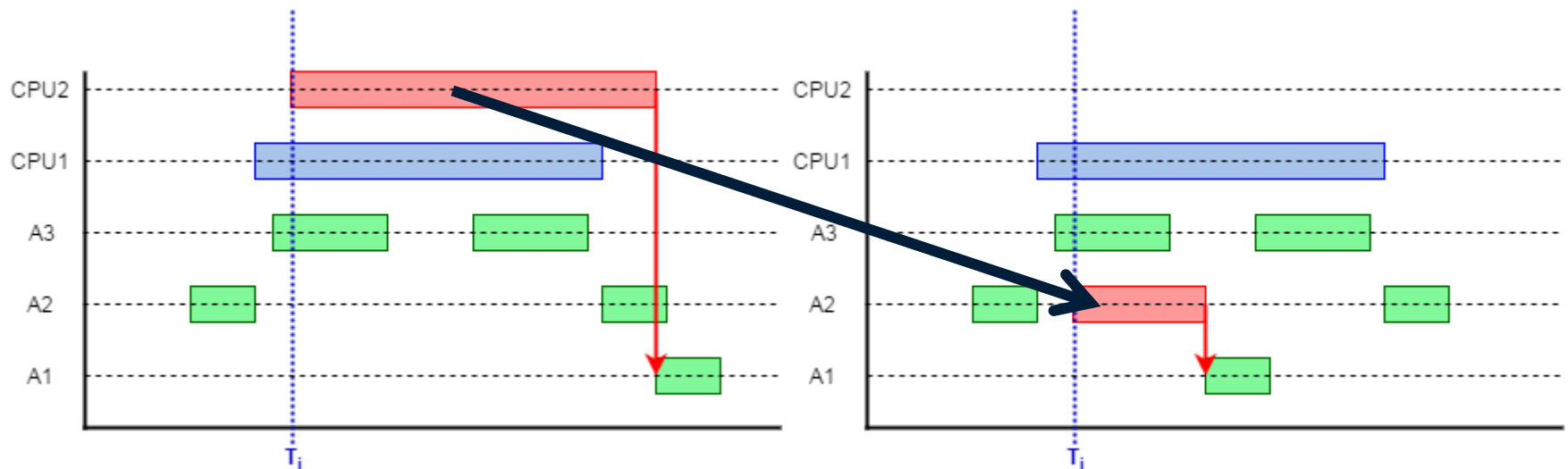
Check if existing area can map the task: NO CONFLICT with already placed tasks.



4. Software Tasks Balancing

In the previous phase some tasks may have switched to a SW implementation.

- Time intervals for these tasks are modified.
- In order to improve the schedule we check whether a task can be moved back to a HW implementation.



5. Start/End Calculation

- Computes the **Start time** and the **End time** for each task.

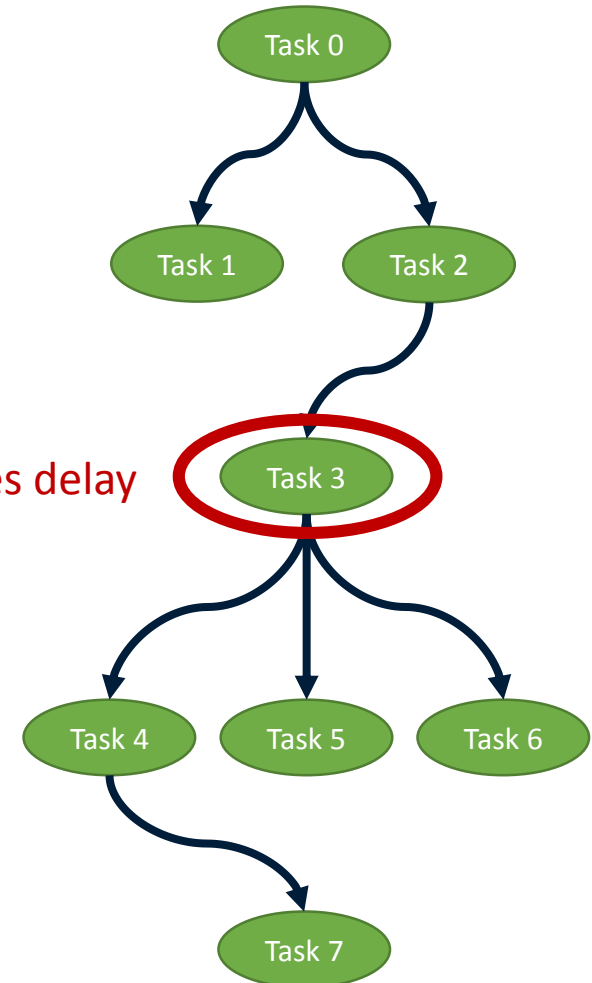
$$T_{START} = T_{MIN}$$

$$T_{END} = T_{START} + executionTime$$

- Checks if **delay** is generated.

$$T_{END} > T_{MAX}$$

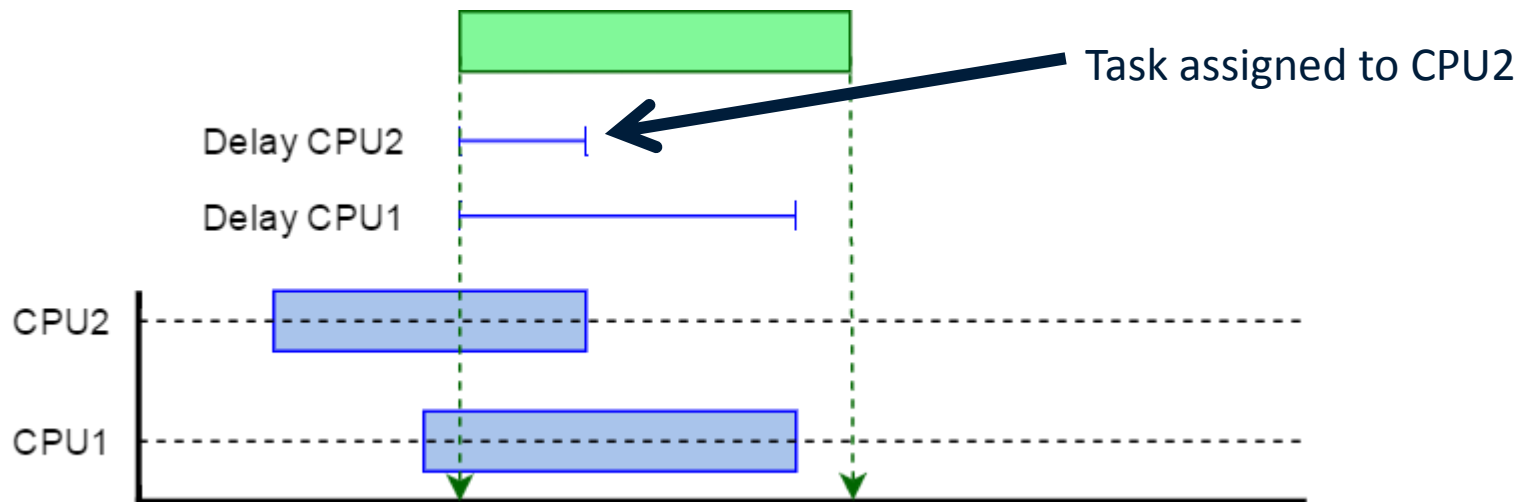
- If delay is generated it is propagated in the subgraph of the task that generates it.



Task 3 generates delay

6. Software Task Mapping

- All the SW tasks are mapped on the CPUs available on the board.
- The tasks are assigned to the CPU that generates the lowest delay.



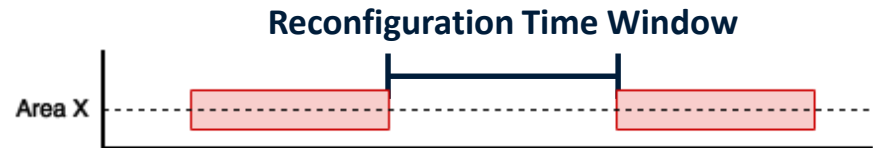
7. Reconfiguration Tasks

Step 1:

- Reconfiguration tasks are generated for the areas having multiple mapped tasks.
- Each reconfiguration has a time window in which should be executed to avoid delay.

$$T_{MIN,reconfiguration} = T_{END,predecessor}$$

$$T_{MAX,reconfiguration} = T_{START,successor}$$

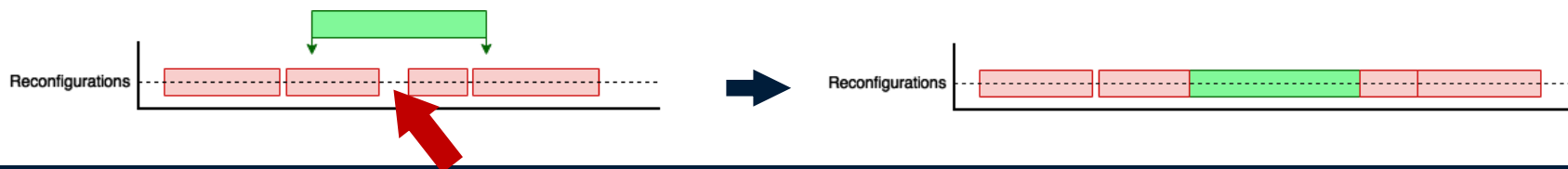


Step 2:

- Schedules all the reconfiguration tasks that have a successor critical task.
- Schedule generated ordering the reconfiguration tasks by T_{MIN} .

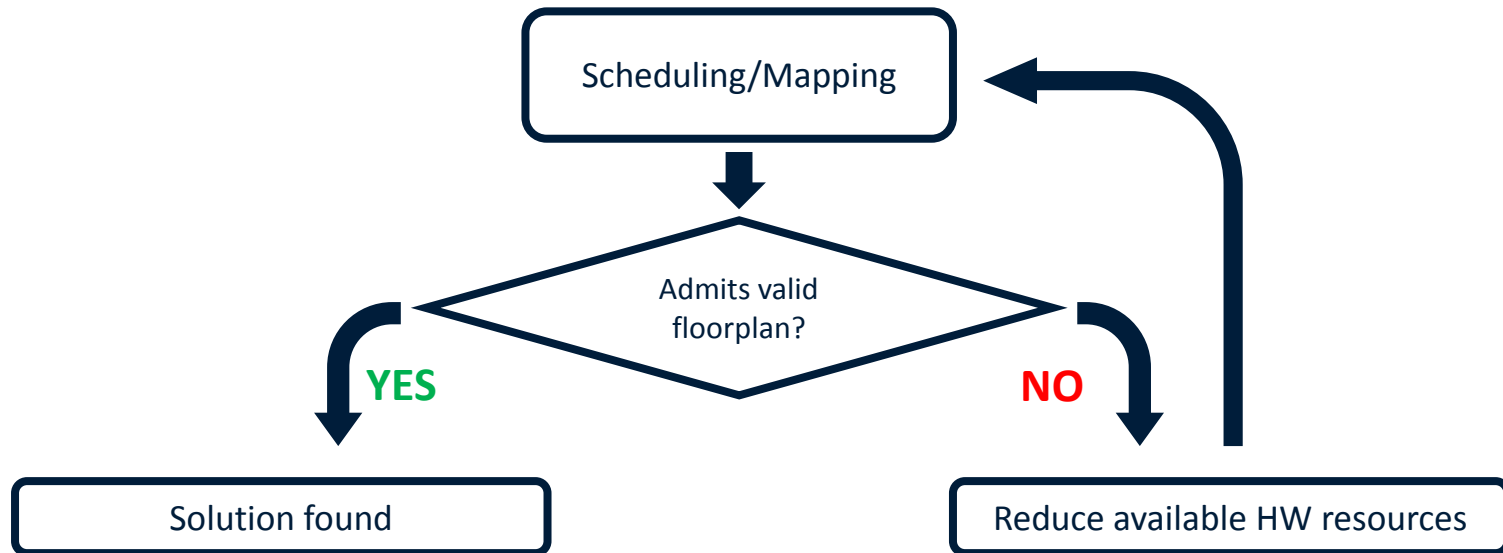
Step 3:

- Schedules all the remaining reconfigurations.
- Reconfiguration tasks ordered by T_{MIN} .
- Schedule generated inserting reconfiguration tasks in the first available slot.



8. Feasibility Check

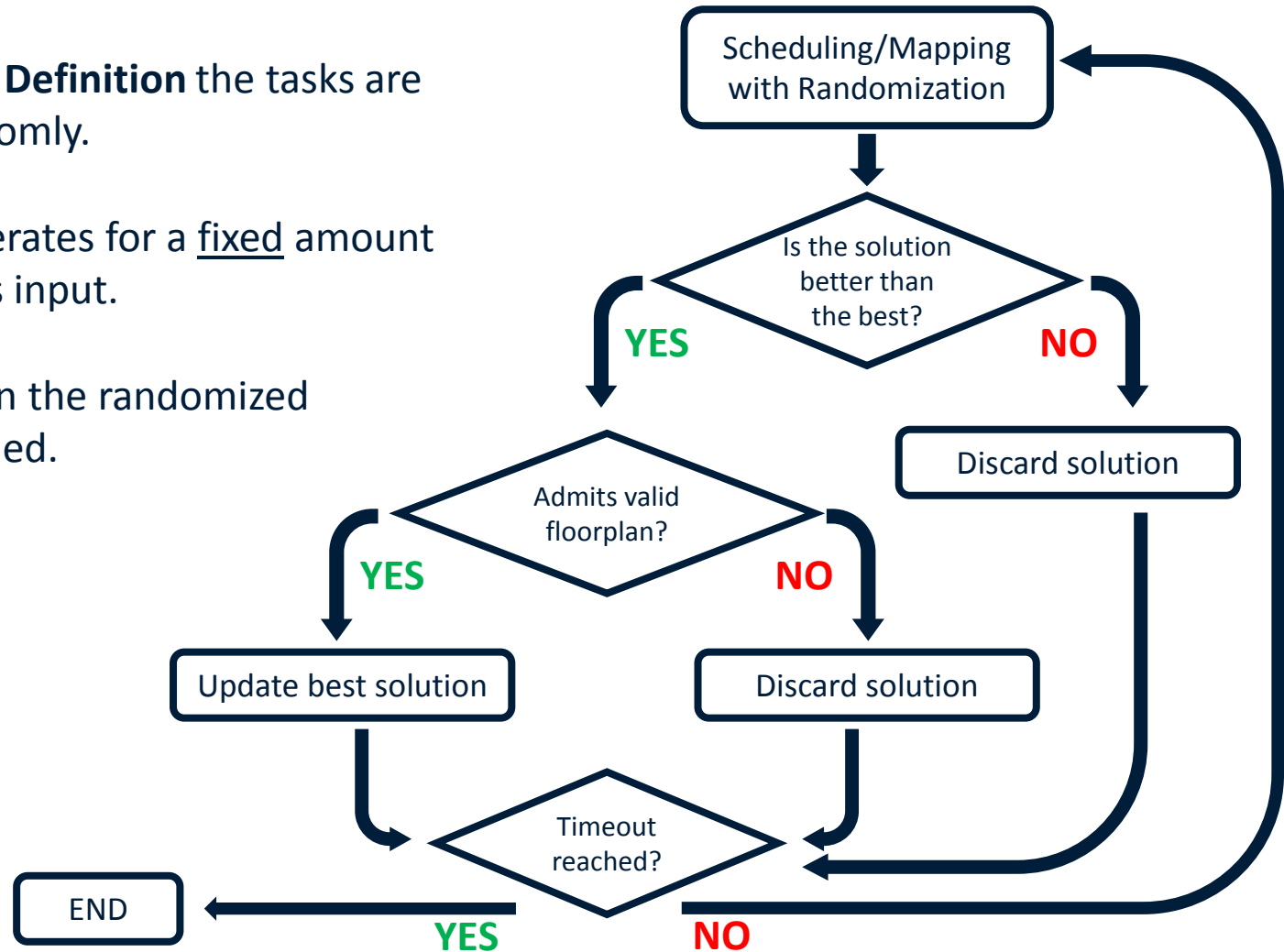
- Checks if the solution found fits in the FPGA.
- Performed using a MILP-Based floorplanning algorithm [1].



[1] M. Rabozzi, A. Miele, and M. D. Santambrogio, "Floorplanning for partially-reconfigurable FPGAs via feasible placements detection," in Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium, 2015, pp. 252–255.

Randomized Version

- In step **Regions Definition** the tasks are processed randomly.
- The program iterates for a fixed amount of time given as input.
- At each iteration the randomized scheduler is called.



Results Analysis (1/3)

Experimental settings:

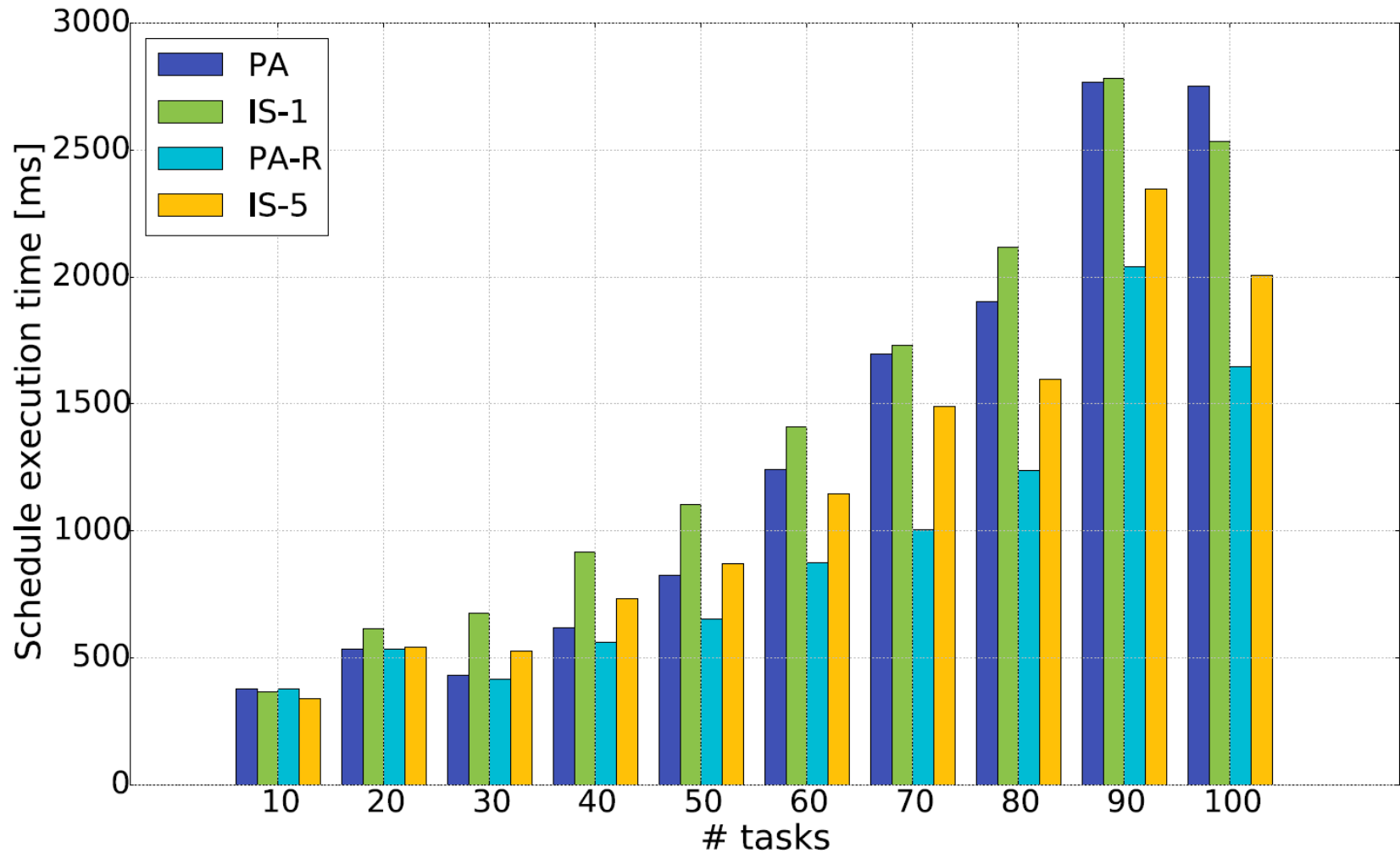
- Tests performed using 100 pseudo-random taskgraphs organized in 10 groups with 10 taskgraphs each.
- Within each group the taskgraph has the same number of tasks.
- Each task has one software implementation and 3 hardware implementations with heterogeneous resource requirements.

Comparison performed against IS-k algorithm [1] ($k = 1$ and $k = 5$) in terms of:

- Scheduling execution time.
- Algorithm execution time.

[1] E. Deiana, M. Rabozzi, R. Cattaneo, and M. Santambrogio, "A multiobjective reconfiguration-aware scheduler for fpga-based heterogeneous architectures," in ReConFigurable Computing and FPGAs (ReConFig), 2015 International Conference on. IEEE, 2015.

Results Analysis (2/3)



Results Analysis (3/3)

TABLE I
ALGORITHMS EXECUTION TIME

| # Tasks | PA [s] | | | IS-1 [s] | PA-R / IS-5 [s] |
|---------|------------|---------------|-------|----------|-----------------|
| | scheduling | floorplanning | total | | |
| 10 | 0.070 | 0.332 | 0.402 | 1.211 | 4.734 |
| 20 | 0.097 | 0.526 | 0.623 | 3.286 | 68.387 |
| 30 | 0.118 | 0.979 | 1.097 | 13.628 | 90.304 |
| 40 | 0.139 | 1.074 | 1.213 | 25.786 | 149.460 |
| 50 | 0.161 | 1.028 | 1.189 | 57.215 | 135.362 |
| 60 | 0.180 | 1.005 | 1.185 | 120.131 | 189.140 |
| 70 | 0.197 | 1.091 | 1.288 | 256.967 | 413.137 |
| 80 | 0.216 | 1.166 | 1.382 | 276.271 | 288.639 |
| 90 | 0.236 | 0.981 | 1.217 | 328.214 | 288.528 |
| 100 | 0.276 | 1.041 | 1.317 | 564.855 | 563.129 |

Conclusions and Future Works

Contributions:

- Provided a fast deterministic scheduling heuristic.
- Provided a tunable randomized scheduling algorithm.
- Floorplan validation of the results.

Feature works:

- Leverage module reuse.
- Explicit communications handling among tasks.
- Consider additional optimization metrics (e.g. power consumption).

Questions?



POLITECNICO
MILANO 1863

POLITECNICO MILANO 1863

NECST
laboratory

Thanks for your attention!