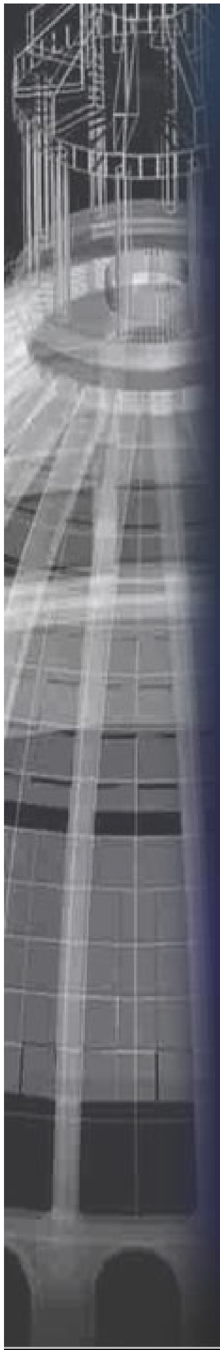
Fast Online Task Placement on FPGAs: Free Space Partitioning and 2D-Hashing

Herbert Walder

Christoph Steiger

Marco Platzner

Swiss Federal Institute of Technology Zurich, Switzerland



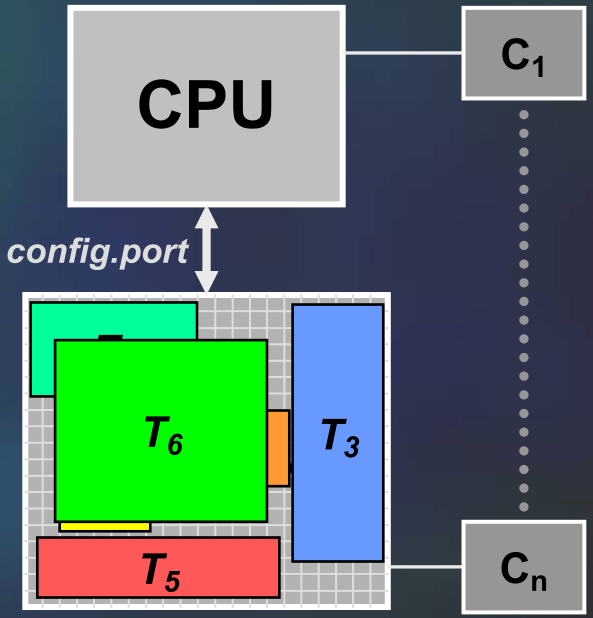
Outline

- Background
- Free Space Management / Partitioning
- Fast Task Placement based on 2D-Hashing
- Conclusion

Background

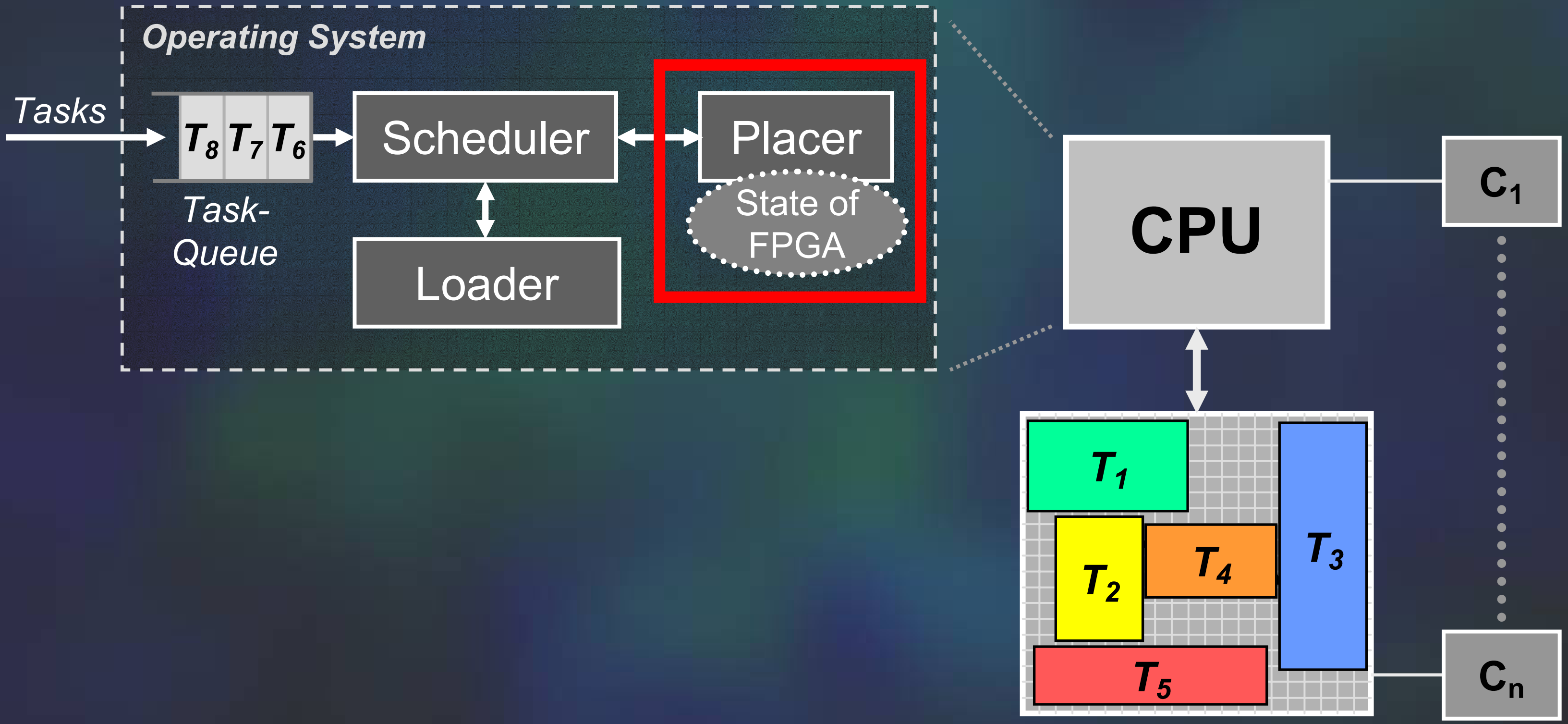
Reconfigurable Embedded System

- CPU, FPGA, external devices
- Reconfigurable hardware OS
- Task model
  - rectangular shape
  - relocatable
  - unknown arrival time & exec. time
  - independent
  - non-preemptive
- FPGA model
  - homogeneous
  - partially reconfigurable
- Online scenario



System Model

Reconfigurable Hardware OS



Operating System

Tasks

T_8 T_7 T_6

Task-Queue

Scheduler

Loader

Placer

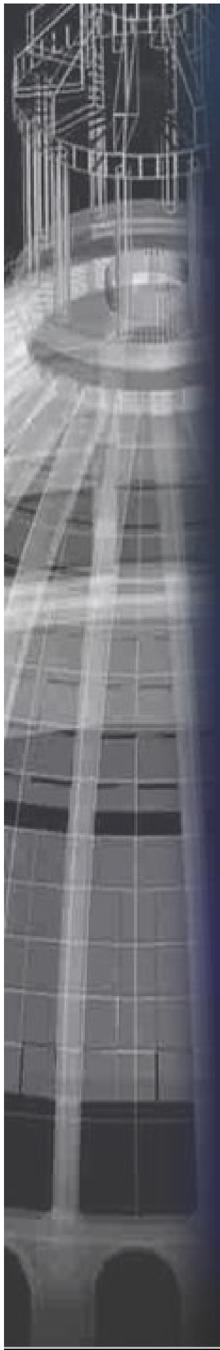
State of FPGA

CPU

C_1

C_n

T_1 T_2 T_3 T_4 T_5



Goals

- → **Manage the resources of the FPGA** (reconfigurable area) **in an efficent way**

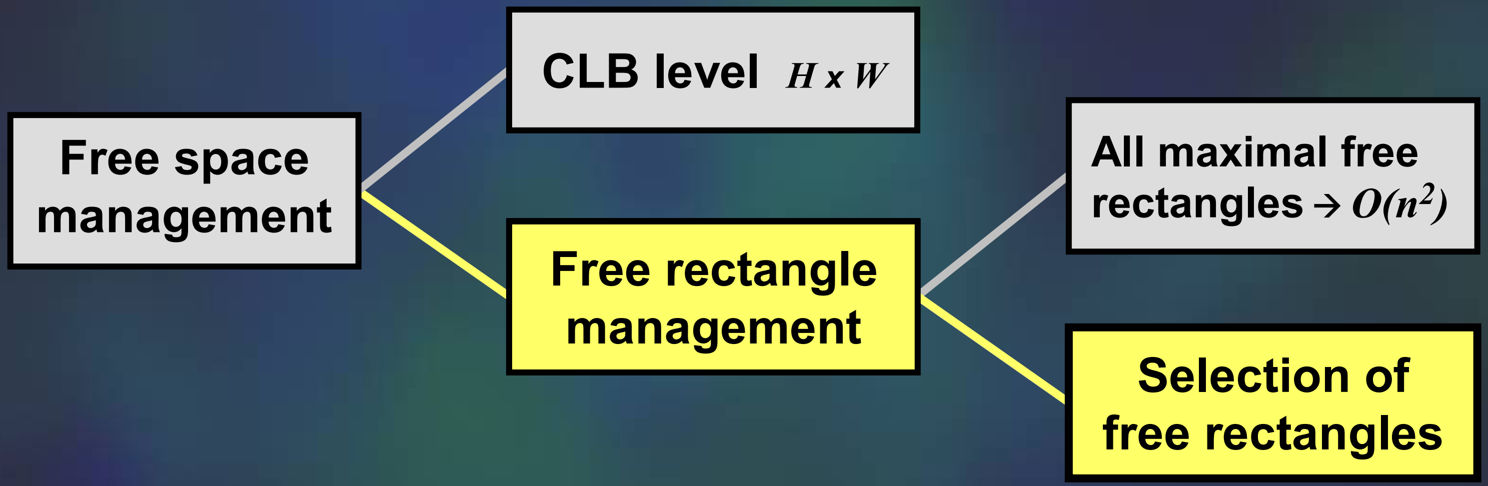
  (low time- and space complexity)

- → **Find as quick as possible a location to place a task**

- → **Start execution of arrived tasks a soon as possible**

  (minimize task waiting time)

Placer: Free Space Management



Free space management

CLB level $H \times W$

Free rectangle management

All maximal free rectangles → $O(n^2)$

Selection of free rectangles

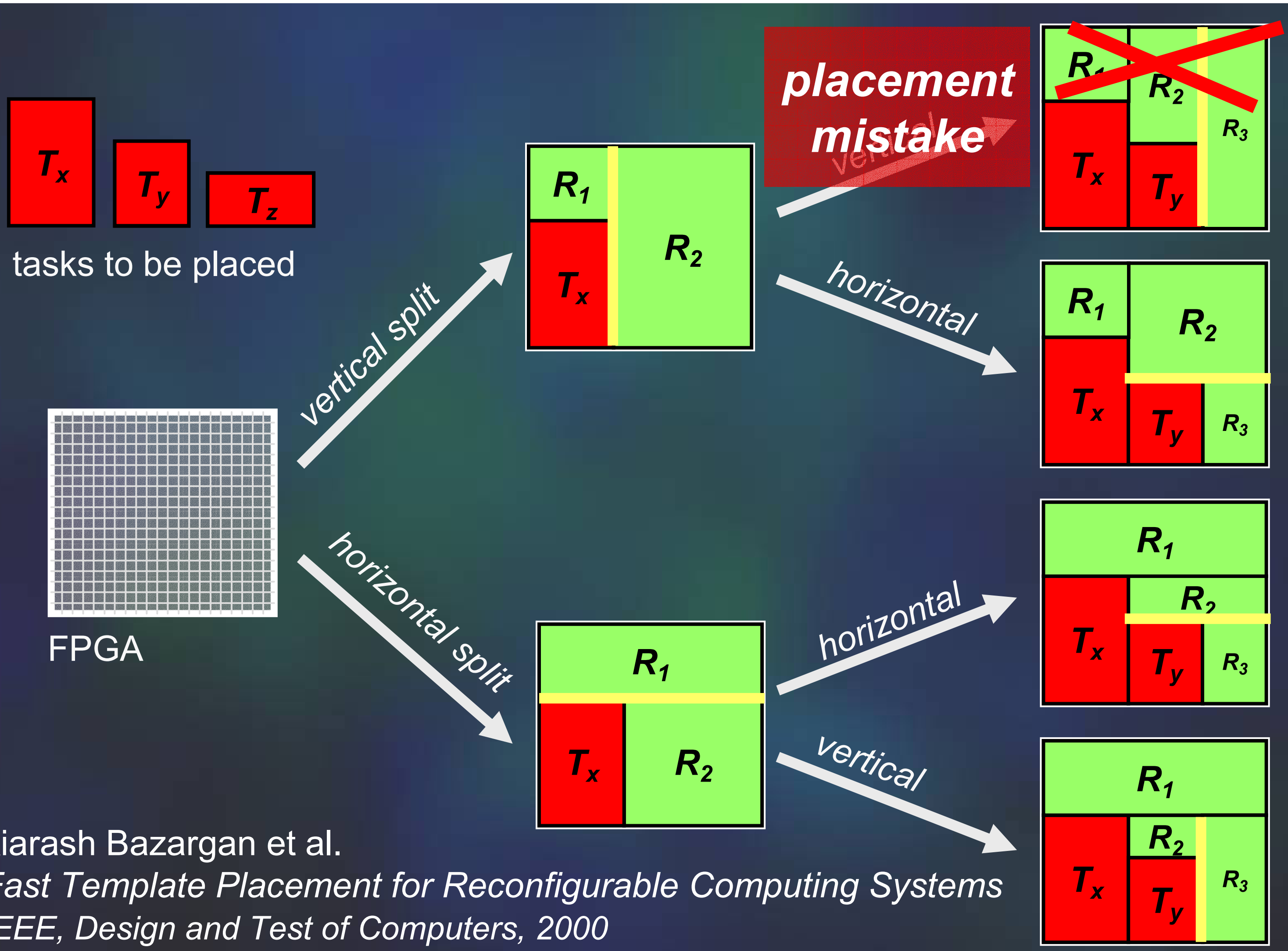
Trade-off → *Placement Quality vs. Efficiency*

Free Rectangle Management → Questions

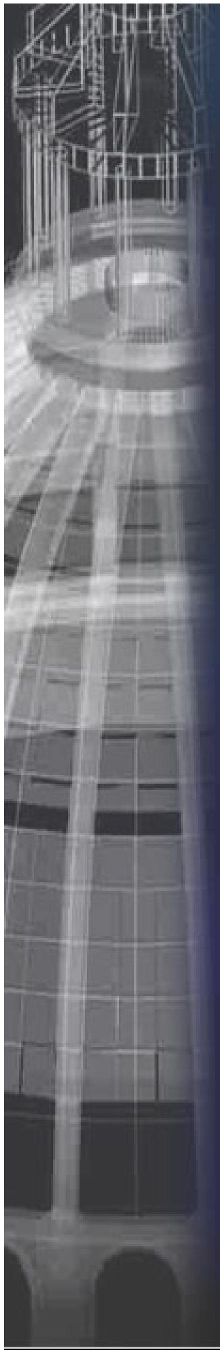
- Which free rectangles are managed? → **Partitioning**
- How are the free rectangles managed?
- Which fitting strategy is used?

} → **Free Rectangle Mgmt.**

Partitioning: Bazargan‘s Approach



→ Kiarash Bazargan et al.
Fast Template Placement for Reconfigurable Computing Systems
IEEE, Design and Test of Computers, 2000



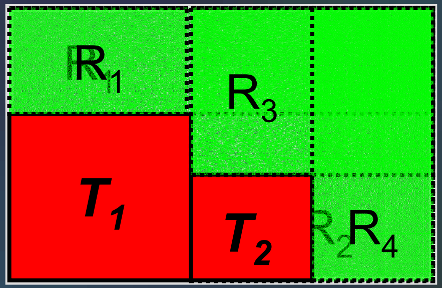
Partitioning: Bazargan‘s Approach

- keeps a set of non overlapping empty rectangles
- uses heuristics to decide whether to split rectangles horizontally or vertically
- split decision taken at task insertion time
- complexity: $O(n)$
  (n = number of tasks placed)

Enhanced Partitioners

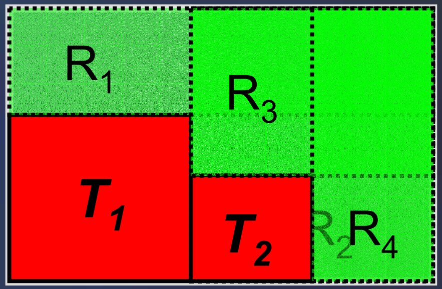
1. Enhanced Bazargan

- delay split decision
- keeps overlapping „child rectangles“
- resize child rectangle upon task insertion



2. „On-the-fly“ (OTF) Partitioning

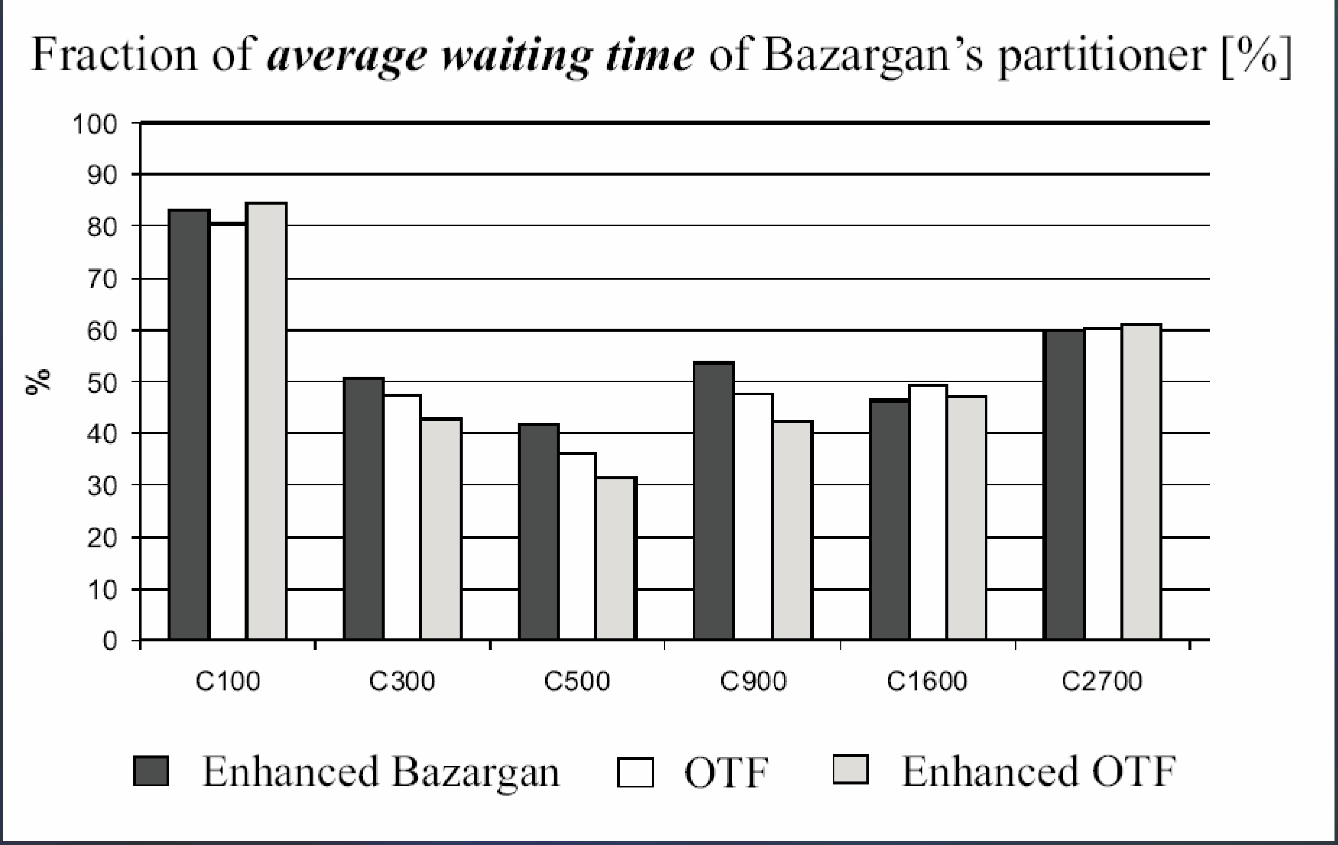
- delay split decision
- keeps overlapping „child rectangles“
- resize child rectangle only if overlapping



3. Enhanced OTF Partitioning

- similar to OTF partitionig: „selective resizing“
- detailed description → see paper

Simulation Results: Partitioning

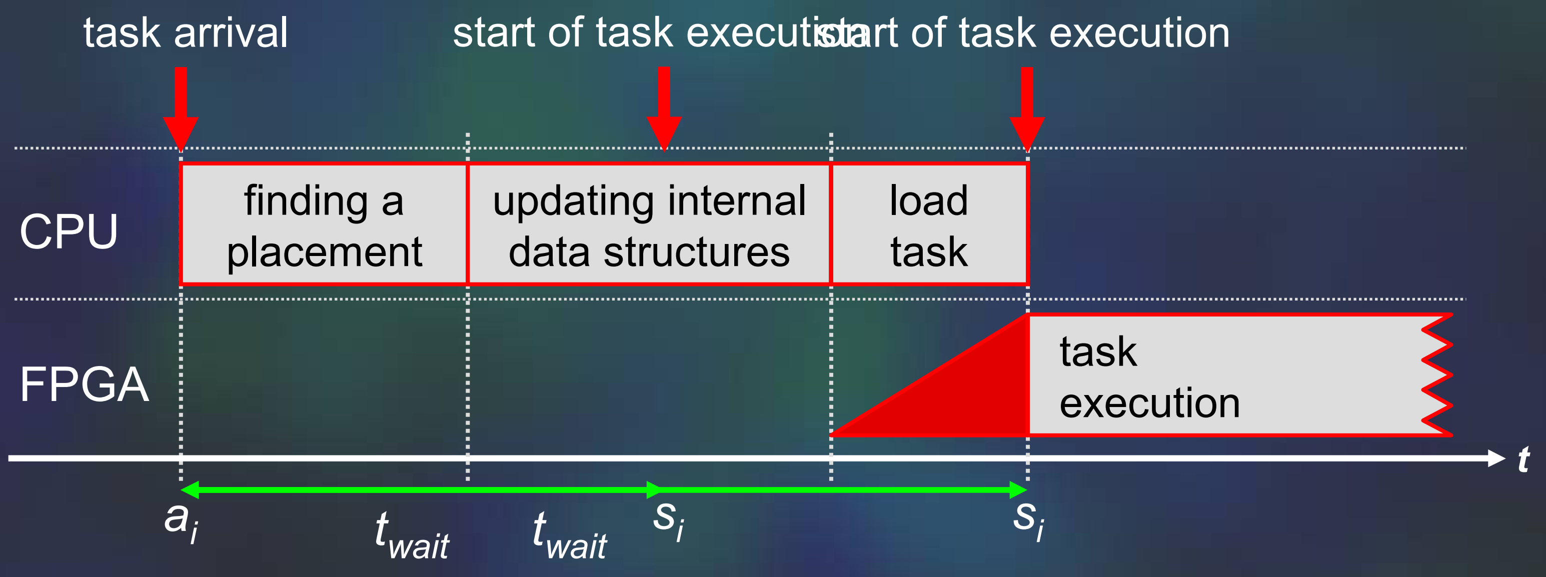


Fraction of ***average waiting time*** of Bazargan’s partitioner [%]

Fast Task Placement

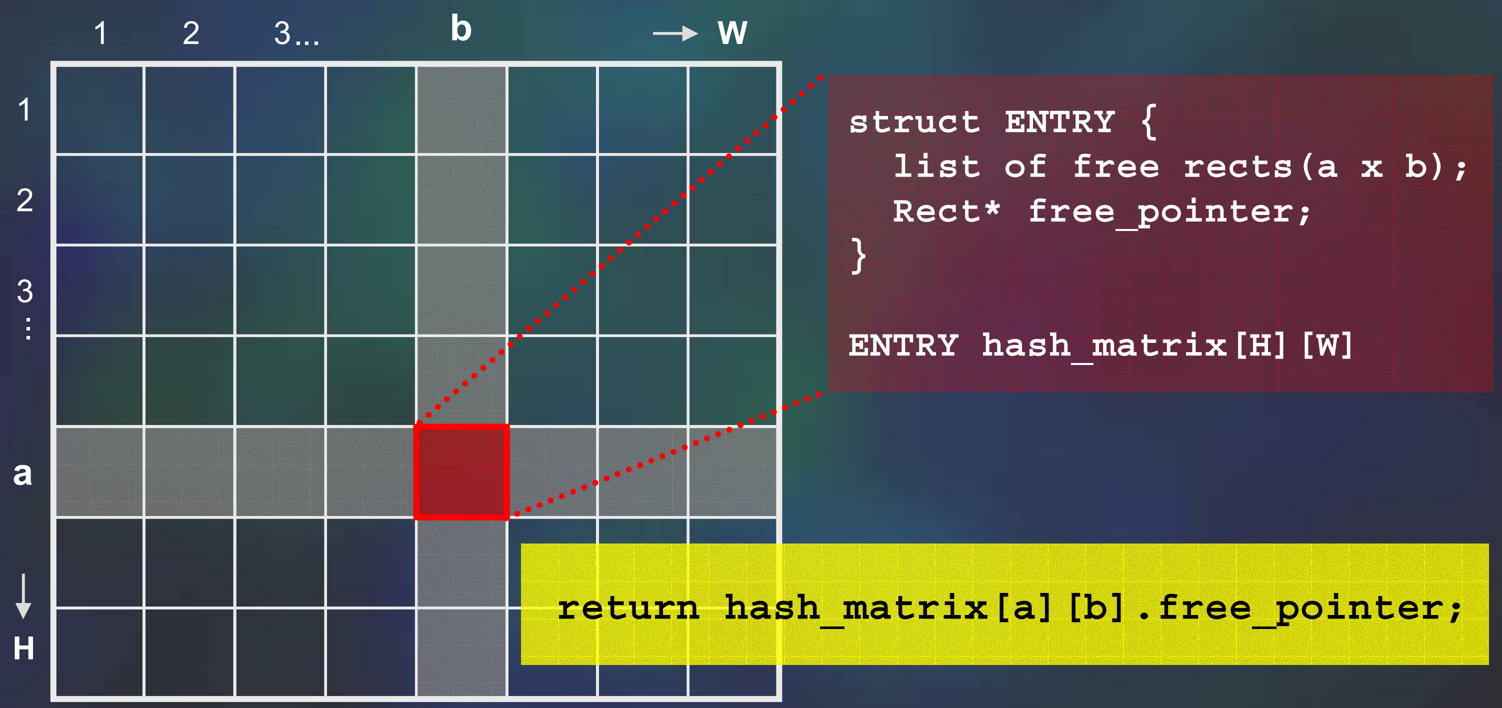
Goal:

→ start execution of arrived tasks as soon as possible



2D Hashing

- 2D hash table holding pointers to free rectangles
- row / column indices represent task dimensions



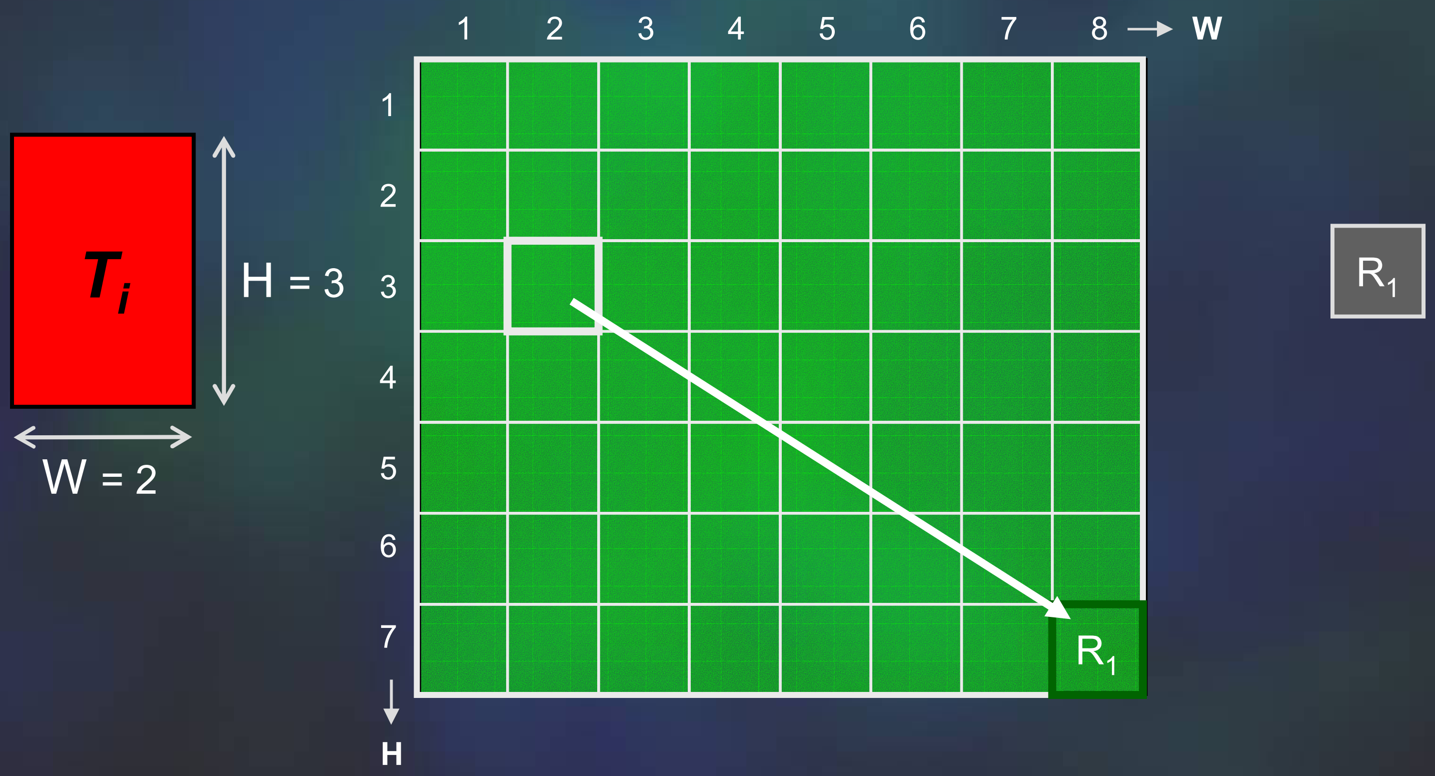
```
struct ENTRY {
  list of free rects(a x b);
  Rect* free_pointer;
}

ENTRY hash_matrix[H][W]
```

```
return hash_matrix[a][b].free_pointer;
```

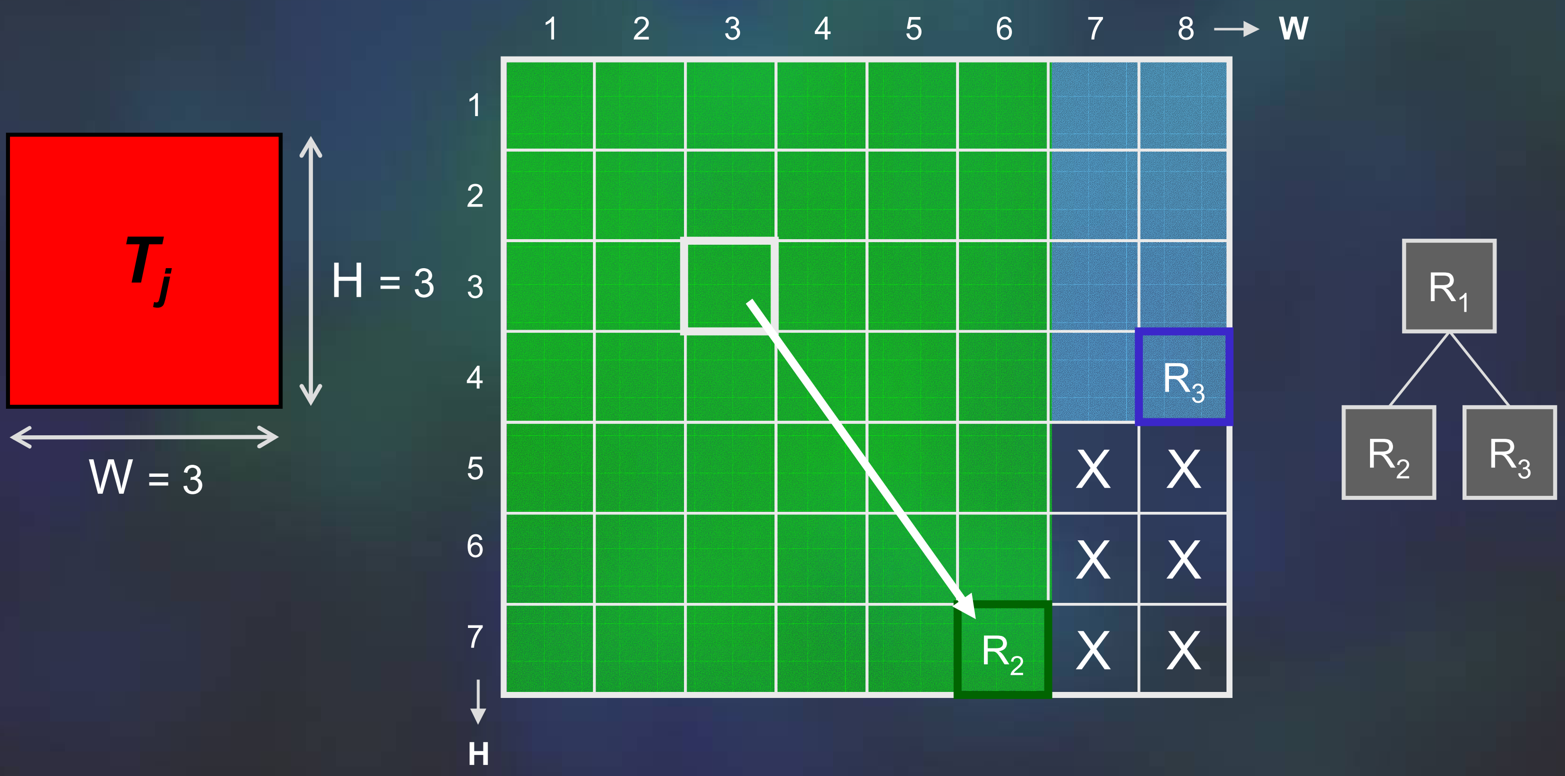
2D Hashing: Example

- no tasks placed on the FPGA
- task T_i with height = 3, width = 2 arrives

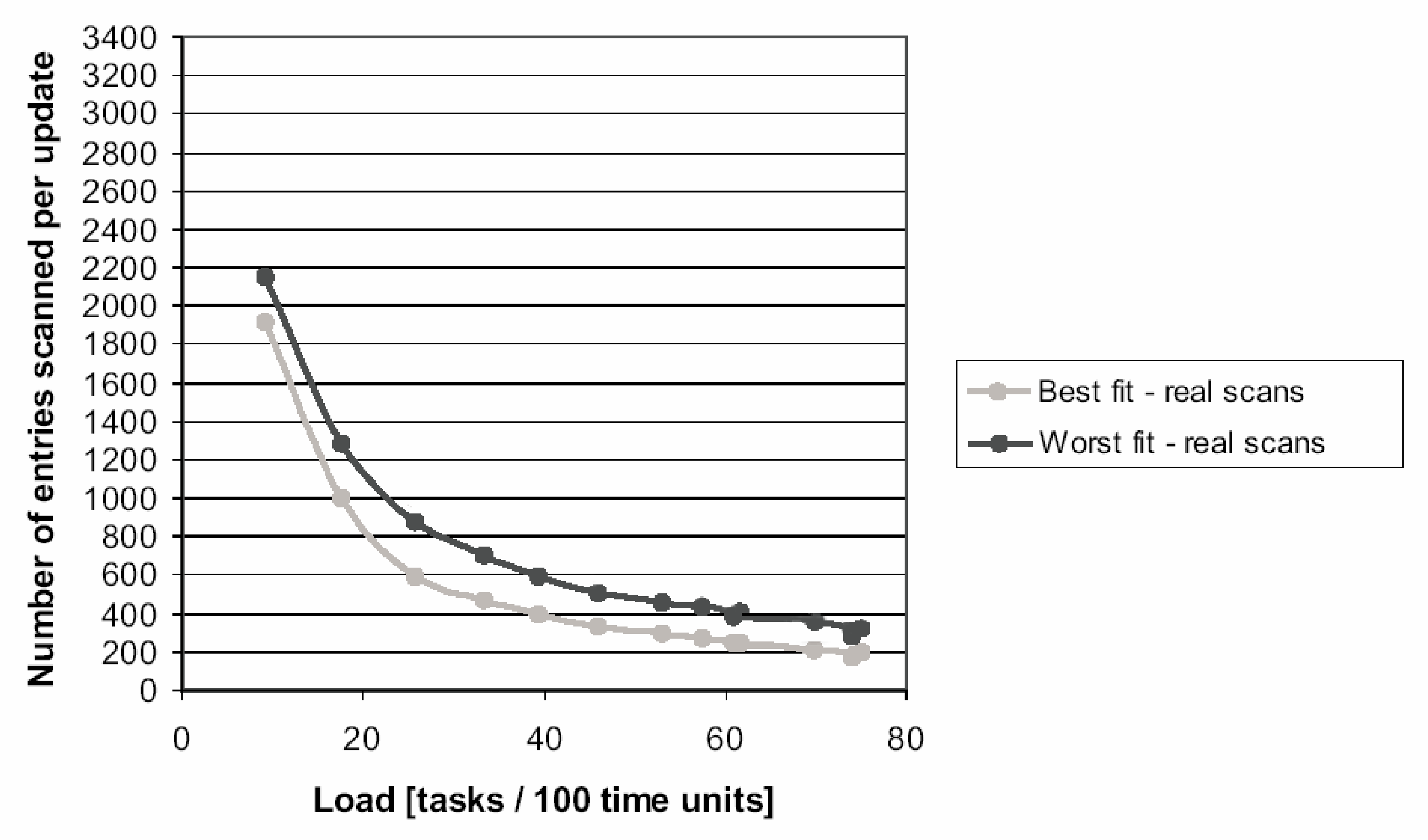


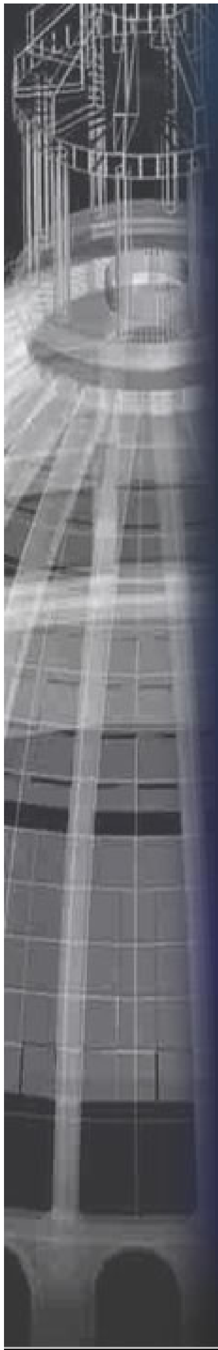
2D Hashing: Example

- task T_i placed, hash matrix updated
- task T_j with height = 3, width = 3 arrives



Simulation Results: Hash-Table





Conclusion

- presented three new partitioning algorithms based on Bazargan‘s approach

  → simulation results show improvement of placement quality (up to 70%)

- introduced method based on 2D hashing to find a feasible placement in $O(1)$ time