Operating Systems Support for Dynamically Reconfigurable Architectures

» *Run-time Partitioning for Just-in-time Compilation* «

Oliver Diessel, UNSW

Grant Wigley and David Kearney, UniSA

Outline

- Motivate OS for Multi-tasking RS
- Some design options
- Task scheduling
- Compilation goals
- Models
- Task partitioning

Motivation for Multi-tasking

- Growing size of reconfigurable logic resource
- Growing range and integration of applications
- Orientation towards real-time tasks

→ How to manage/support large reconfigurable logic resource in a multitasked environment?

→ space- or time-shared?
→ fixed or variable partitioning?

Fixed Partition Size

<u>Pros</u>

- “easy” to design for
- supports location independence & fault tolerance
- “easy” to manage

<u>Cons</u>

- task performance can suffer
- internal fragmentation

Variable Partition Size

<u>Pros</u>

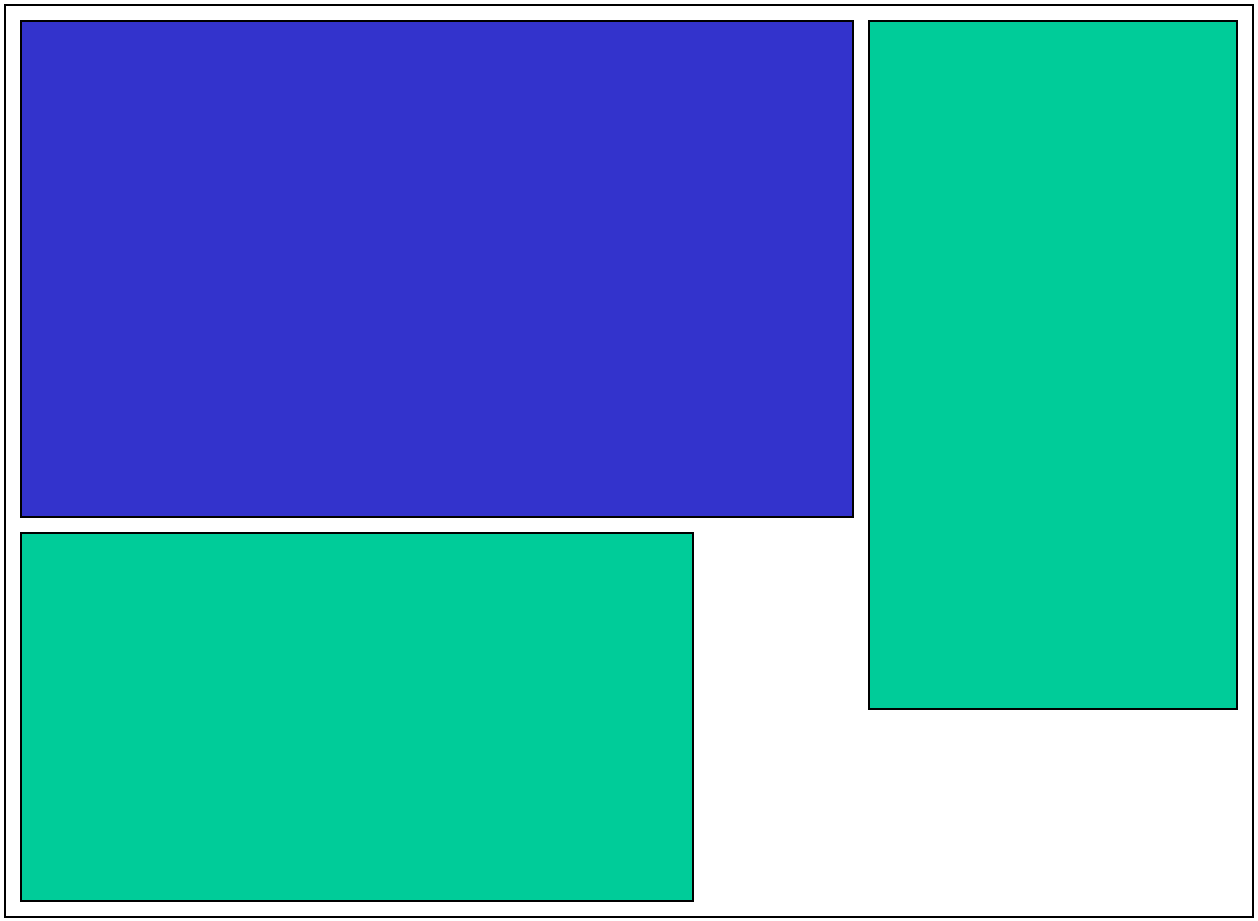
- allocation flexibility allows optimal performance to be achieved
- better utilization possible

<u>Cons</u>

- need to cope with dynamic resource availability — scheduling complexity
- external fragmentation

Coping with Variably Sized Tasks

- Wait for or preempt executing tasks
- Move executing tasks to free up space
- Adapt incoming task to space available at run time
  - requires fast partitioning, placement, and routing
  - requires on-line scheduling of
    - logic
    - routing resources
    - pins
    - memory



Long-Term Goals

1. Develop accurate and efficient indicators that can tell us whether to accept or reject real-time reconfigurable computing tasks into common architectural models.

2. Develop algorithms and techniques that allow us to complete the physical design of tasks without contributing significantly to overheads.

Models

Task

- Netlisted combinational task graph (DAG)
  - each vertex has associated area and latency
  - each edge has associated width
- Dataset representing given number of cycles
- Off-line processing

Machine

- Single context logic resource of given area and pin count
- Assume fixed clock period
- Given amount of fast local buffer memory
- Assume reconfiguration time proportional to sum of vertex areas, sum of edge weights, and number of pins in config

Partitioning Goal

- Determine partitioning of DAG and a configuration sequence (subject to correctness, area, latency, pin, and buffer size constraints) that identifies for each partition the number of cycles it should execute for in order to minimize the total execution time $\sum_{i=1,l} R(P_i) + N_i * T_r$

- Do so quickly

First steps

Previous work in area

- Partitioning
  - Survey: Alpert & Kahng, Integration, Aug 95
  - Single Context:GajjalaPurna & Bhatia, TOC, Jun 99; Kaul & Vemuri, RAW99
  - Multiple Context: Trimberger, FPGA98; Liu & Wong, FPGA99; Chang & Marek-Sadowska, TOC, Jun 99
- Placement
  - Sankar & Rose, FPGA99
- Routing
  - Swartz, Betz & Rose, FPGA98