Software/Hardware Co-Scheduling for Reconfigurable Computing Systems
Proshanta Saha, Tarek El-Ghazawi
NSF Center for High-Performance Reconfigurable Computing (CHREC)
ECE Department, The George Washington University
{sahap, tarek}@gwu.edu

Abstract
A formal methodology for automatic hardware-software partitioning and co-scheduling between the µP and the FPGA has not yet been established. Current work in automatic task partitioning and scheduling for the reconfigurable systems strictly addresses the FPGA hardware, and does not take advantage of the synergy between the microprocessor and the FPGA. In this work, we consider the problem of co-scheduling task graphs on reconfigurable systems. The target systems have an execution model which allows any subtask that can run on the FPGA to also run on the microprocessor, and allows reconfigurability of the FPGA (subject to area, performance, resource, and timing constraints). In this paper, we introduce a new heuristic algorithm for such hardware/software co-scheduling, ReCoS. It will be shown that the proposed algorithm provides up to an order of magnitude improvement in scheduling and execution times when compared with hardware/software co-schedulers found in the embedded systems area, after adapting them for reconfigurable computing.

1. Introduction
Scheduling is critical for achieving efficient utilization and achieving high performance for reconfigurable computing (RC) systems. The primary targets of this paper are RC systems that have both microprocessors and Field Programmable Gate Arrays (FPGAs). Early works on scheduling for RC systems focus on online task placement algorithms to assist designers in finding efficient area utilization [1][2]. However these scheduling methods address only the reconfigurable hardware and do not offer solutions for co-scheduling between the µP and the FPGA. For the purpose of clarity this community will be referred to as the reconfigurable hardware (RH) community.

Co-scheduling has been explored by other communities in various capacities. The heterogeneous computing (HC) community has introduced several algorithms to handle the scheduling of heterogeneous processors in a distributed computing environment [3]. The embedded computing (EC) community tackles the issue of scheduling in a heterogeneous environment consisting of fixed logic devices such as ASIC and/or special purpose processors with the assumption that each task of a program can only execute on a given type of processor [4][5][6].

2. Scheduling Requirements
RC systems require scheduling algorithms for PEs that are not fixed and can be catered towards the current processing needs. The scheduling algorithms have to consider the resource constraints on the reconfigurable hardware such as the number of FFs, LUTs, MULTs, CLBs, the reconfiguration overhead, communication overhead, routing overhead, size constraints, throughput, and power constraints before selecting a task to map on to the FPGA. In addition, the scheduling algorithms have to consider multiple implementations of a task depending on the objective function. Unlike reconfigurable hardware only scheduling algorithms, RC systems need to schedule tasks between the µP and the FPGA to take full advantage of its synergy.

3. Proposed Scheduling Algorithm
The proposed algorithm takes several static heuristic models from the HC and RH communities and adapts the models to handle RC system constraints (communication overhead, number of FFs, LUTs, MULTs, CLBs, the reconfiguration overhead, routing overhead, size constraints, throughput, and etc) and multiple configurations for a given task.

Figure 1: ReCoS Algorithm
The primary techniques upon which our algorithm is based are MCT, MET, Min-Min and OLB from HC and the EDF-NF and MSDL from RH. Several modifications and optimizations are necessary to utilize existing algorithms for RC co-scheduling. The reconfigurable computing co-scheduler (ReCoS) algorithm combines the strengths of the HC and RH scheduling algorithms namely in speed and scheduling capabilities and pays...
special attention to load balancing issues and lack of co-design constraint satisfaction.

4. Results

The ReCoS algorithm is compared against three EC algorithms, Hou [4], Yen [5], and Oh [6] algorithms on an SRC6 platform. All subtasks targeting the µP are written in C (ISO-C99), while all subtasks targeting the FPGA are written in Verilog (Verilog 2001) along with the necessary wrappers to interface with the SRC6 system written in Carte’s MAP-C language [7]. The scheduling algorithms assume a 2 PE model (µP and FPGA). The test bed includes four applications implemented for both the µP and FPGA, JPEG IDCT, MPEG IDCT, DES encryption, and IDEA encryption.

Figure 2 shows the benefit of using the ReCoS algorithm. Despite the enhanced Yen, Hou, and Oh algorithms [8] effort to reduce the number of PEs, in this case resulting in the best case scenario of a single PE and thus similar execution times, they don’t exploit the synergy of the µP and FPGA nor the inherent parallelism FPGAs have to offer. The ReCoS algorithm is able to exploit the space remaining on a FPGA to add more implementations of the subtasks onto the chip without violating area constraint $\sum A_{rd} \leq 75$.

![Figure 2: ReCoS algorithm compared against enhanced reconfigurable aware Yen, Hou, and Oh algorithms on 1M blocks of data.](image)

5. Conclusions

To the best of our knowledge there are no automatic hardware/software co-scheduling algorithms for reconfigurable computing systems that take advantage of the synergy between the µP and the FPGA available today. Related work either focuses solely on the reconfigurable hardware or exists in related fields such as HC and EC. Manual partitioning efforts are currently the only means of co-scheduling on an RC system, and are often tedious for large applications. In this work we investigated automatic scheduling algorithms that can aid developers in identifying an optimal co-schedule that takes advantage of the synergy between the µP and the FPGA quickly and efficiently. Our proposed algorithm, ReCoS, addresses automatic co-scheduling for reconfigurable computing systems by leveraging the work done by the HC, RH, and EC communities. The ReCoS algorithm as compared to the Hou, Yen, and Oh algorithms, shows a significant improvement in both optimal schedule search time and reduction of execution time. In the time to generate the schedules, ReCoS is analytically and experimentally shown to be one order of magnitude faster than Yen, Hou, and Oh. In execution time, ReCos has produced schedules that exploited reconfigurability to also provide in most cases an order of magnitude improvement in speed. ReCoS ability to exploit RC features and the synergy between the µP and FPGA provides a clear distinction that RC systems require more than just a simple adaptation of existing schedulers.

6. Acknowledgements

This work was supported in part by the I/UCRC Program of the National Science Foundation under the NSF Center for High-Performance Reconfigurable Computing (CHREC).

7. REFERENCES