Dynamic Reconfiguration of an Application on Hybrid Reconfigurable Systems

1Mahendra Vucha 2Arvind Rajawat
1Research Scholar, MANIT, Bhopal and Asst. Prof, Dept. EC, GGITM, Bhopal, India
Email: mahendra.1548@gmail.com
2Associate Professor, Dept. Electronics and Communication, MANIT, Bhopal, India
Email: rajawata@manit.ac.in

Abstract- A formal methodology for automatic hardware-software partitioning and co-scheduling the tasks of an application between Microprocessor and Programmable Logic Devices (PLDs) has become emerging research area of hardware software co-design. The main objective of this research is to get full advantage of hardware utilization and speedup the application execution.

Hardware software partitioning and scheduling the tasks of an application on Hybrid Reconfigurable Systems (HRS) became a critical issue for efficient hardware resource utilization and speedup the application execution. The main objective of this paper is, partition the application into software, hardware and hybrid tasks by considering some predefined parameters of tasks of the application, reconfigurable logic unit array and schedule them onto HRS. In this research, the HRS is combination of Microprocessor (µP) and array of Reconfigurable Logic Devices like Field Programmable Gate Arrays (FPGAs) with variable area reconfigurable logic units (RLUs). In this research, Algorithms for Partition and schedule the tasks of an application onto HRS would be implemented and verified them by using SystemC environment to support hardware software co-design.

Index Terms - Dynamic Reconfiguration, Hybrid Reconfigurable Systems, Field Programmable Systems, Microprocessor, Reconfigurable Logic Unit.

I. INTRODUCTION

Reconfigurable Systems (RS) promised to be a valuable alternative for conventional computing systems such as Personal Computers, µP based systems, Application Specific Integrated Circuits (ASIC) and Application Specific Instruction set Processors (ASIP). RS customizes hardware at run-time through dynamic reconfiguration and gives efficiency compared to ASIC/ASIP. FPGA gives a design close to the performance benefits of an ASIC solution and provides reduction in NRE cost due to post-fabrication flexibility. Reconfigurable Computing (RC) is an emerging technology that combines the flexibility of FPGAs with the programmability found in General Purpose Processor (GPP)/ Digital Signal Processor (DSP) in a unified and provides easy programming environment. To get real advantage of RS, one has to provide a general design methodology by combination of µP and RS called HRS to explore architectures which support hardware reconfiguration at run-time. Early works on scheduling methods for RC systems focus on online task placement algorithm to assist designers in finding efficient area utilization [1]. In this research, the preferred HRS should be combination of µP and FPGAs having variable area RLU. Here, the scheduling algorithm considered the resource constraints of the PLDs of HRS such as the number of Flip-Flops (FFs), Look Up Tables (LUTs), Multiplexers, Configurable Logic Blocks (CLBs), configuration overheads, communication overheads, routing overheads, size constraints, throughputs and power constraints as inputs.

In this work, a model has been designed to configure the application onto HRS and it supports run-time reconfiguration by efficient partitioning, scheduling, mapping and migrating of the tasks of an application between µP and RS. Here, scheduling of an application will become a critical issue in order to achieve high efficiency. In this paper, a model was proposed with separate modules for partition, schedule, allocation and migration of tasks between µP and RS and also at times tasks may flip over modules to speed up task execution.

II. RELATED WORK

A scheduled model was proposed in [1] based on fixed area reconfigurable systems prototype to schedule software tasks, hardware tasks and hybrid tasks but mainly concentrated on placement of tasks. In [2], a scheduling algorithm had been presented to co-schedule the tasks onto µP and Field Programmable Gate Array (FPGA) environment by considering several static heuristic models from different communities like Embedded Computing (EC), Heterogeneous Computing (HC) and Reconfigurable Hardware (RHC). In [8], a co-scheduling algorithm was proposed by considering the resource constraints of Reconfigurable Hardware and the multiple implementations of task on target RC system before mapping task onto FPGA and it have been compared with EC, RC and RH scheduling algorithms. The work in [4] considered heuristic level-based list scheduling, Integrated Linear Programming (ILP), loop fission, static scheduling techniques and some of dynamic scheduling techniques to schedule the tasks onto RS but all of them failed to schedule the tasks within given time. The dynamic scheduling algorithm Maximum Laxity First (MLF) was proposed in [4] considers maximum laxity as parameter to prioritize the tasks but it may block critical tasks which are having low laxity forever. A scheduling algorithm in [3] addresses the problem of scheduling and mapping the tasks onto the RLU for a given application task graph by introducing a new concept that is parameterized modules and variable silicon area sizes and here, the tasks should be non-preemptive.
In this research, tasks may vary in terms of hardware area and the time to execute. There was an approximately similar research proposed to schedule hardware tasks onto variable area RS in [3] and the tasks should be non-preemptive but it leads to execution overheads at times. In [9] there was an online HW/SW partitioning and co-scheduling algorithm proposed to schedule the tasks by prioritize the hybrid tasks according to different important parameters of tasks and partitioning the sorted tasks according to their Earliest Finish Time (EFT) on software and hardware processing units. In this present research, a model was developed to eliminate issues represented above to get maximum utilization of resources and speedup application execution by co-scheduling the tasks onto HRS.

III. HRS PLATFORM

The HRS architecture been characterized as shown in fig 1.

All modules such as µP, system memory, communication and configuration management, periphery devices and variable area RLU FPGA arrays are connected by system bus. Communication and configuration control logic manages data communication and task configuration onto HRS. Reconfigurable Computation (RC) should perform on variable area FPGA array. Based on this platform, an application may be partitioned into three types of tasks called software task, hardware task and hybrid task. In this research, software task and hardware task should implement on µP and FPGA respectively and the hybrid task may implement either on µP or FPGA. The scheduler could take the task parameters like execution time, starting time, finishing time, delay time and RLU parameters like area, reconfiguration time, communicating overheads into consideration to schedule the application onto HRS. Finally, with the help of system bus the software tasks (binary codes), hardware tasks (bit-stream files) and hybrid tasks (both binary codes and bit-stream files) can be loaded into binary code library and bit-stream file library respectively and then configured them onto HRS from the libraries.

IV. HRS SCHEDULING MODEL

In this research, the software and hardware tasks are permanently allotted to µP and FPGA respectively but the hybrid task could be allotted to either µP or FPGA depending on system run-time status. In order to achieve high efficiency in hardware utilization and speed up the application execution, this model can be divided into three layers and each layer only communicates with its adjacent layer.

Layer 1, provide interface to collect application and partitioned them into different tasks depending on their run-time parameters. The partitioned tasks further can be sent to layer 2.

Layer 2, collects the tasks from layer 1 and prioritize them according to predefined parameters and prepare scheduling list of them according to priorities. The scheduled tasks further moved to layer 3.

Layer 3, collects the task from layer 2 and configured them onto HRS according to their priorities fixed by layer 2 at run-time.

In this work, the model was build with 6 queues and 7 modules to configure the application onto HRS. The proposed model is shown in fig 2 and the function of queues and modules of the model is stated bellow.

Q1 stores the arriving tasks in layer1, Q2, Q3, Q4 are stores the partitioned tasks like software, hybrid and hardware respectively in layer 2, and Q5, Q6 stores scheduled software and hardware tasks respectively in layer 3.

Application decode module decode the tasks of an application at runtime and forward them to Partitioning module, Partitioning module in layer 1 partition the tasks into software tasks, hybrid tasks and hardware tasks by considering their predefined parameters, Dynamic priority allocation module in layer 2 prioritize the tasks according to parameters of task and configuration.
The scheduled software and hardware tasks stored in Q5 and Q6 should transfer into Load and Configuration modules respectively. Load module can fetch the binary code corresponding to software task from binary code library and that can be loaded into system memory of Hybrid Reconfigurable System. Similarly Configuration module can fetch the bit-stream file of hybrid task from bit-stream file assembly. Thus, the tasks can be configured onto FPGA to get efficient utilization of reconfigurable hardware. Hybrid tasks have two implementation forms i.e. hardware & software implementations so that for hybrid tasks it have to take a decision to send them into either Q5 or Q6 based on availability of µP and FPGA. But hardware implementation gives better performance then software implementation so it always prefers Q6 then Q5.

D. Tasks migration:

Process of task migration involves the ability to stop a task execution on one processor and divert their execution onto another processor. As hybrid tasks can swapped between queues Q5 and Q6, the Migrating module should control tasks load balance between microprocessor and FPGA to enhance resource utilization. In this work, the task migrating module can considered three cases to swap hybrid tasks between queues Q5 and Q6.

i. Hybrid tasks can be migrated from queue Q6 to Q5, when there is no suitable FPGA are vacant to configure present hybrid tasks onto RLU array.

ii. Hybrid tasks can be migrated from queue Q6 to Q5, when there is a suitable FPGA vacant and it can be configured by hardware task which is having higher priority than the current hybrid task.

iii. Hybrid tasks can be migrated from queue Q5 to Q6 to increase FPGA resource utilization, when a part of RLU array is left idle.

E. Task allocation:

The scheduled software and hardware tasks stored in Q5 and Q6 should transfer into Load and Configuration modules respectively. Load module can fetch the binary code corresponding to software task from binary code library and that can be loaded into system memory of Hybrid Reconfigurable System. Similarly Configuration module can fetch the bit-stream file of hybrid task from bit-stream file assembly.
stream file library and that can be configured onto particular FPGA of HRS.

V. IMPLEMENTATION SCHEME

The model specified above, takes the application graph and their parameters $a_i$, $d_i$, $W_i$, $c_i$, $h_e$ as inputs. Each task of an application can be executed on $\mu$P to capture the software parameters $d_i$, $s_e$ and also be executed on FPGA to capture hardware parameters $a_i$, $d_i$, $W_i$, $c_i$, $h_e$. The captured parameters, binary code and bit-stream file of tasks of an application can be stored into binary code library and bit-stream file library of HRS respectively.

The application can be executed on three different environments (Intel Core 2 duo @2.0, @1.2G GHz $\mu$P, FPGA and HRS having one GPP, 4 FPGAs with variable gate count) as stated below

i. The application can be executed on $\mu$P by scheduling tasks of an application according to their cost function.

ii. The application can be implemented on Variable FPGA array by scheduling tasks of an application according to their cost function and parameters like area of tasks, configuration time and number of LUTs of an FPGA.

iii. Finally, the above specified application can be executed on HRS by scheduling the tasks of an application according to their cost function.

VI. CONCLUSION

In this work, a new model was proposed for dynamically reconfiguration of an application on HRS. This work paid more concentration on architecture of HRS and scheduling of tasks of an application at run-time. Unlike other scheduling methods, this model was proposed the methodology to increase efficiency of hardware utilization and speedup application execution by considering the advantage of hardware software co-design. As stated above the hardware implementation of tasks enhances the execution efficiency of an application compared with other implementations but it is impossible at runtime (Tasks may have gate count larger then FPGA count at runtime). Hence, implementation of an application on HRS only the solution to enhance hardware utilization at runtime and also it gives flexibility to implement the application either on $\mu$P or RS.

ACKNOWLEDGMENT

The authors wish to thank the Director Dr. R.P. Singh of MANIT, Bhopal, and the Management, Principal Dr. P. S. Venkataramu of GGITM, Bhopal for their constant encouragement to this work.

REFERENCES


Mahendra Vucha received his B. Tech in Electronics & Communication Engineering from JNTU, Hyderabad in 2007 and M. Tech degree in VLSI and Embedded System Design from MANIT, in 2009. He is currently working for his Ph. D degree at MANIT and also working as Asst. Prof in Gyan Ganga Institute of Tech & Mgmt, Dept. of Electronics and Communication Engineering, Bhopal (M.P), India. His areas of interest are Hardware Software Co-Design, Analog and Digital System Design and Embedded System Design.

Arvind Rajawat received his B. Tech in Electronics & Communication Engineering from Govt. Engineering College in 1989, M. Tech degree in Computer Science Engineering from GSITS, Indore, in 1991 and Ph. D degree from MANIT, Bhopal. He is currently working as Associate professor in Dept. Electronics and Communication, MANIT, Bhopal (M.P), India. His areas of interest are Hardware Software Co-Design, Embedded System Design and Digital System Design.