System-level Design Space Exploration and Resource Mapping Strategies for a Reconfigurable Hybrid System

Seong-Yong Ahn and Jeong-A Lee
Department of Computer Engineering,
Chosun University,
Tel. +82-62-230-7711, Fax.: +82-62-233-6896
e-mail : *dis@chosun.ac.kr, jalee@chosun.ac.kr*

Abstract:
In this paper we proposed the design space exploration environment of re-configurable hybrid systems and evaluate the performance by changing design parameters. With this, we analyzed the effect of various scheduling methods which determine how we allocate hardware/ software resources to application program. A simple static (fixed) mapping strategy produces almost the same performance compared with a sophisticated dynamic mapping strategy especially when a CPU is already busy with its pre-assigned own tasks.

1. Introduction

Reconfigurable architecture is a hardware system which can adapt its hardware structure to process given application programs more efficiently while conventional architecture implies a fixed hardware structure.[1] Although the progress of reconfigurable systems shows advances in performance and flexibility, the methodology and tools that design and analyze reconfigurable systems are still dependent on ad-hoc solutions. Especially the resource allocation strategy for H/W-S/W partitioning and how to configure a reconfigurable systems are difficult problems and emerging research topics[2],[4],[5],[6].

In this paper, we consider a hybrid processing architecture composed of a conventional CPU and a FPGA among many alternatives as a reconfigurable platform. It is a hard task to find an efficient hardware configuration which satisfies design constraints for an application and becomes harder when a set of applications will be executed since we need to deal with the huge design space to explore trade-offs caused by adjustments of architecture and mapping strategy. We adopted a design space exploration approach known as Y-chart approach for this challenging task as it allows to measure quantitatively the performance of different mapping strategy from algorithms in homeogenous hardware configurations[3].

The Y-chart Approach is a methodology to provide designers with quantitative data after analyzing the performance of architectures for a given set of applications. For performance analysis, each application is mapped onto the architecture and the performance of each application-architecture mapping combination is evaluated. The resulting performance numbers may inspire the architecture designer to improve the architecture. In this paper, we extended the Y-chart approach for a hybrid processing architecture by developing a DSE(design space exploration) tool.

2. DSE Tools

The DSE tool, as shown in Figure 1, a retargetable simulator for reconfigurable system consists of three parts which are an application simulator, a hardware simulator and a mapping controller. We assume that the input description for the application simulator is based on the Kahn-Process-Network which is wildly adopted for a DSP modeling so that the semantic gap between the application model and architecture model is minimized. This assumption is acceptable as there exists a CAD platform which transforms a system description into a Kahn-Process-Network description. We also assume that the target reconfigurable architecture in FPGA is a stream-based architecture so that the data-flow of the Kahn-Process-Network can be mapped to the target architecture transparently.

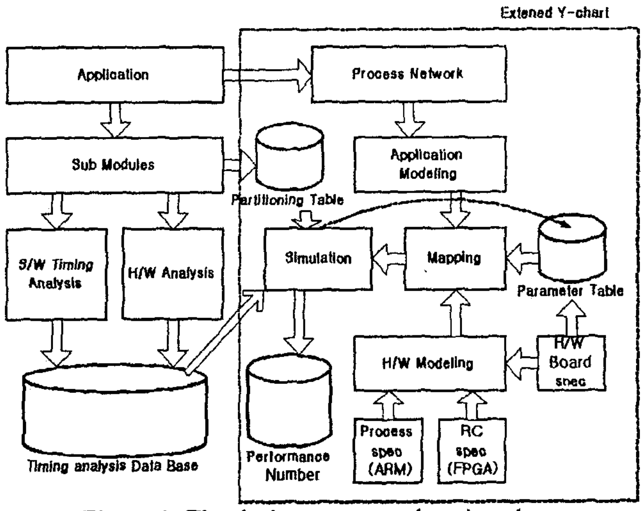

Figure 1. The design space exploration sheme

The application simulator fed with the input of the application model, simulates the flow of traces. We employ Kahn process networks for the application model. In Kahn process networks, buffers connecting to processes represent unbounded FIFO queues. These queues automatically buffer the output of a process and allow processes to consume the tokens from a buffer with a blocking read.

Therefore, Kahn process networks describe applications in a deterministic way[7].

The architecture simulator does not go through cycle-level simulation but collects timing estimates when a function is mapped to an architecture element. The application-architecture mapping controller holds information about mapping from an application program to an architecture element.

The simulator was developed using Trace-Driven simulation method which takes traces produced by the application simulator and allocates them to available hardware resources such as a CPU or a reconfigurable FPGA. Using this tool, a designer can vary design parameters and estimate performance numbers for each potential mapping as a software to be run in a CPU or a hardware configuration to be implemented, without building a prototype. The tool can provide useful design information including scheduling of tasks to a system designer who wants to know which partitioning cases, i.e., a hardware configuration satisfies the time and resource constraints in a timely and cost-effective way.

The application simulator produces traces to model the interactions among the sub-task of the application program. A sub-task in the application simulator is represented by a process. The process executes its own function, and passes the trace to the application-architecture mapping controller. The trace which was returned to the application simulator via application-architecture mapping controller, gives back to the processes that have requested the processing. Then this process passes a trace to the next process. When a trace produced is not consumed by other processes, it is put in the FIFO queue.

The application-architecture mapping controller assigns a trace to the corresponding architecture element. The assigned architecture does not accept any other processing requests until the processing completes. The architecture simulator collects the processing time of each architecture element. It can only be occupied after finishing the current processing. A collision occurs when several sub-tasks are mapped to the same architecture element. We will consider several shcemes to resolve these collison.

3. Mapping Strategies

There are various ways to map hardware resources when queued tasks are being chosen to be allocated. The mapping strategies can be fixed in advance or dynamic depending on the status of available hardware resources at the time. We employ several mapping strategies, such as fixed FCFS(first come first serve), Semi-dynamic FCFS and priority policy, to observe its effects. Here are overviews of these strategies.

Fixed FCFS is the simplest among them. The scheduler doesn't query the status of resoures. It means that each queued task has a fixed resource for mapping.

Semi-Dynamic FCFS is similar to fixed FCFS. Unlike fixed FCFS it query the status of resouces when collision occurs. Another word, if a resource selected is busy, The scheduler check availablity of other resources. If there is a available resource, The scheduler maps the task to it, otherwise write-backs to the waiting queue which is located between application simulator and mapping controller.

Priority Policy sets priority of queued tasks . when the scheduler chose a task withing the waiting queue for mapping, it considers priority of the tasks first. A task that is located closer to the final output(sink process) in the task graph is assigned a higher priority.

4. Experiment

For an experiment, we consider H.263 encoder as an application model and a hypothetical system which has a CPU and a FPGA as a hybrid system. The data-flow for H.263 encoder is shown in Figure 2.

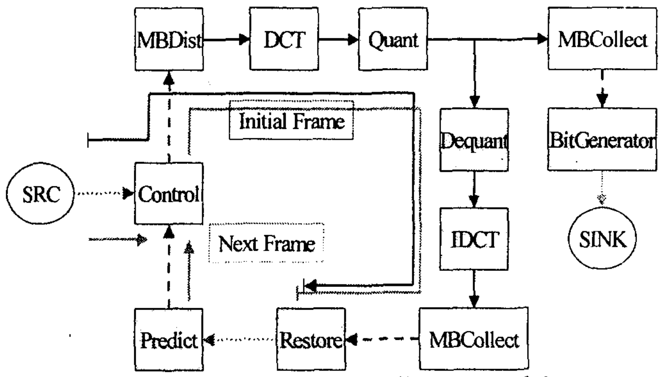

Figure 2. H.263 encoder application model

The performance metrics we used are parallelism and utilization . The formula for these metrics are shown below where T_end is the ending time of frame processing.

$$Parallelism = \frac{\text{Combined Executin Time of The Workloads}}{\text{T\_end}}$$

$$Utilization = \frac{\text{Time a Rseoure is Used}}{\text{T\_End}}$$

Based on these metrics, we evaluate different mapping strategies for H.263 encoder application.

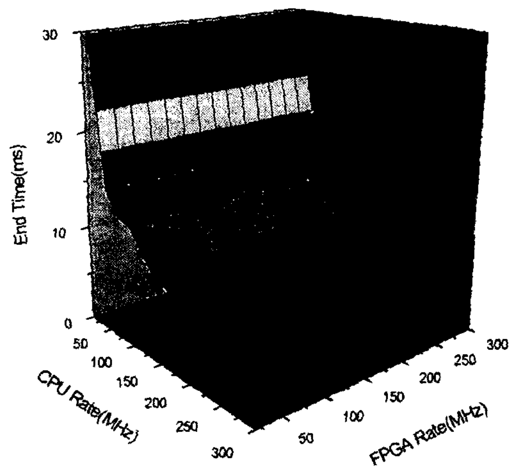

Figure 3. End time of the simulation by changing CPU rate and FPGA rate

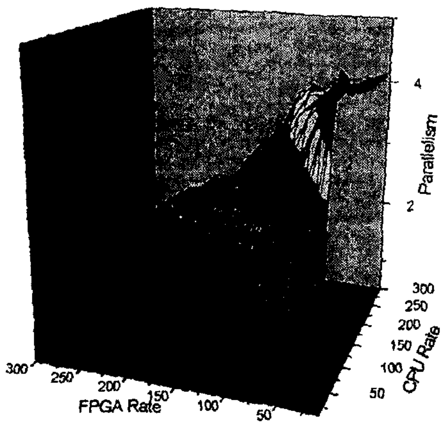

Figure 6. Parallelism by changing CPU rate and FPGA rate

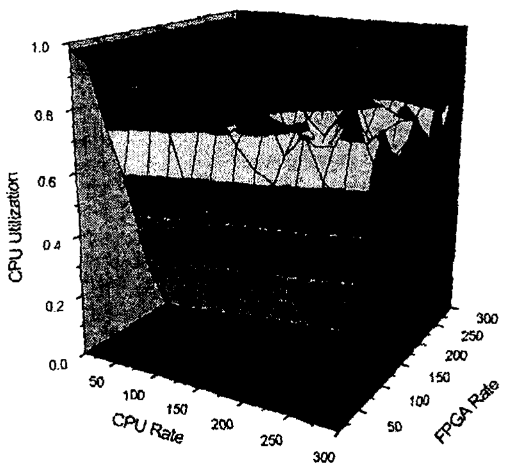

Figure 4. CPU utilization by changing CPU rate and FPGA rate

Figure 3-6 show end time, CPU utilization, FPGA utilization and parallelism repectively by changing CPU and FPGA rate. In Fingure 3, the knee point is found at the point where CPU rate is around 150MHz and FPGA rate is around 100 MHz. The end time is reduced remarkably as CPU rate becomes higher. FPGA rate has a similar impact but the slope of reduction for ending time in this case is smoother. The graph shows that a system designer does not need to use high-end hardware devices to achive certain constraints. In Figure 4, we can see that CPU utilizatin is less than 10% when CPU rate is above 250MHz and FPGA rate is less than 50MHz. The reason is in this case CPU is not the bottleneck for the processing any more. As shown in the Figure 4, CPU utilization is over 80% most cases. Especially when CPU rate is low, CPU becomes the bottleneck of the performance. As shown in Figure 5 and 6, when FPGA rate becomes lower and CPU rate becomes higher, FPGA utilization and parallelism becomes higher because the processing capacity of CPU becomes approximately the same as that of the FPGA.

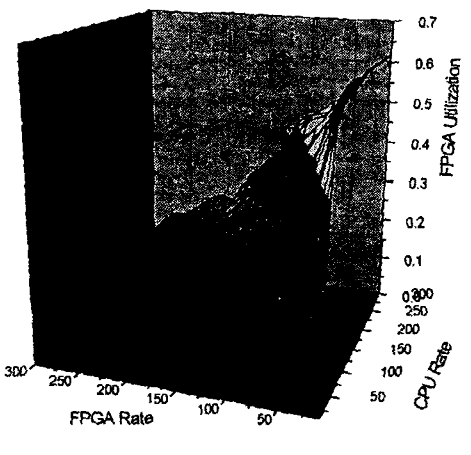

Figure 5. FPGA utilization by changing CPU rate and FPGA rate

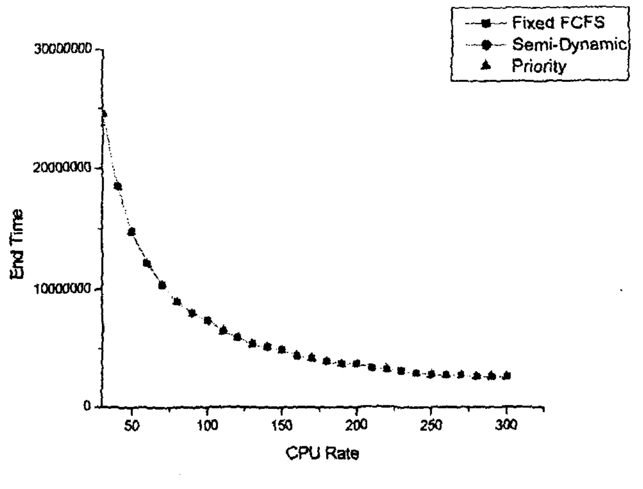

Figure 7. End time of the simulation for each scheduling method by changing CPU rate

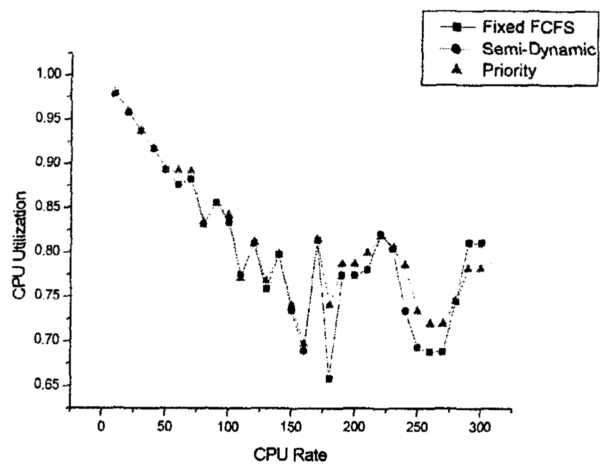

Figure 8. CPU utilization for each scheduling method by changing CPU rate

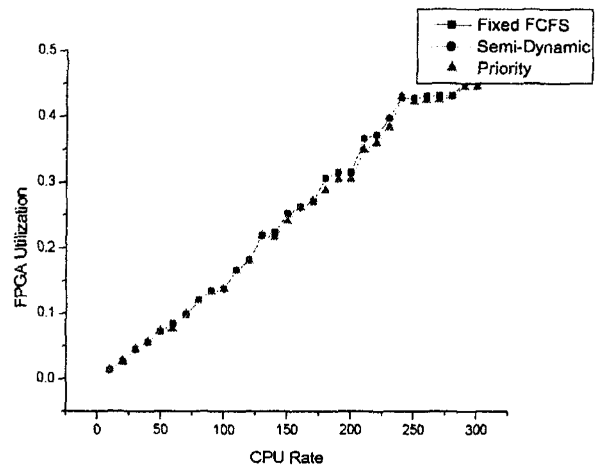

Figure 9. FPGA utilization for each scheduling method by changing CPU rate

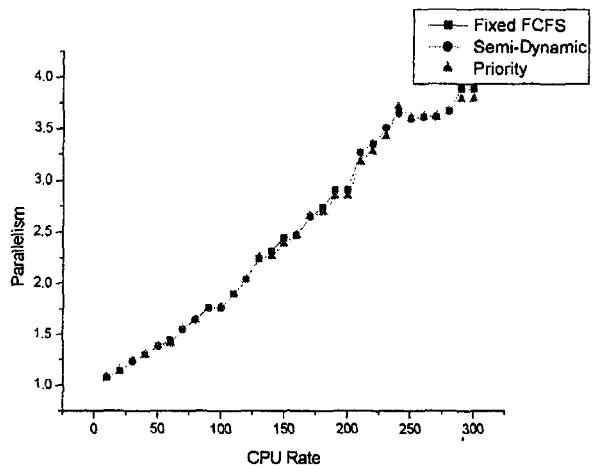

Figure 10. Parallelism for each scheduling method by changing CPU rate

Figure 7 – 10 show end time, CPU utilization, FPGA utilization and parallelism repectively for each scheduling method by changing CPU rate. The results show that the overall performance is almost the same for the scheduling schemes we considered.

5. Conclusion

In this paper we proposed the design space exploration environment of re-configurable hybrid systems and evaluate the performance by changing design parameters. With this DSE, we analyzed the performance of several scheduling methods. From our experiments, we found out that a simple static (fixed) mapping strategy produces almost the same performance compared with a sophisticated dynamic mapping strategy especially when a CPU is already busy with its pre-assigned own tasks. Considering the fact that the dynamic mapping strategy implies the overhead of maintaining the data structure to figure out the current resource usages, the simple static mapping strategy is much better choice to implement. In the paper, we presented the test data to support our understanding of mapping strategies which shows different experiments by changing the design parameters, especially the load of CPU with control intensive jobs and various processing capabilities of CPU and FPGAs.

References

[1] O. T. Albahama, P. Cheung, and T.J. Clarke, "On the Viability of FPGA-Based Integrated Coprocessors," In Proceedings of IEEE Symposium of FPGAs for Custom Computing Machines, pp. 206-215, Apr. 1996

[2] E. Sanchez. M. Sipper, J.-O. Haenni, J.-L. Beuchat, A. Stauffer, and A. Perez-Uribe, "Static and Dynamic Configurable Systems", IEEE Transactions on Computers VOL.48, No.6, June 1999.

[3] B. Kienhuis, E. Deprettere, K.A. Vissers, and P. Wolf. "An approach for quantitative analysis of application-specific dataflow architectures", In Proceedings of 11th Intl. Conference of Applications-specific Systems, Architectures and Processors (ASAP'97), pages 338-349, Zurich, Switzerland, 1997

[4] A.C.J. Kienhuis, Design Space Exploration of Stream-based Dataflow Architectures, PhD thesis, Delft University of Technology, Netherlands, 1998.

[5] S. Bakshi, D. D. Gajaski, "Hardware/ Software Partitioning and Pipelining", In Proceedings of the 34th annual conference on Design Automation Conference, 1997, pages 713-716

[6] A. Kalavade, P. A. Subrahmanyam, "Hardware/Software Partitioning for Multifunction Systems", In Proceedings of International Conference on Computer Aided Design, pages 516-521, 1997,

[7] G. Kahn, "The semantics of a simple language for parallel programming, "Info. Proc., pages 471-475, Stockholm, Aug. 1974