SoC Architecture Explorer

Imai Laboratory, Osaka University, Japan
http://vlsilab.ics.es.osaka-u.ac.jp/

Abstract

SoC Architecture Explorer is an architecture exploration tool for SoC design. It explores all possible parameter sets under constraints and estimates design quality of every architecture candidates.

1. Introduction

In recent years, the requirement for multi-functionality and complexity of embedded systems increase design cost. Consequently, IP-based design methodology that aims to reduce the design time by reusing predesigned functional blocks, called IP (Intellectual Property), has been focused.

To find out optimal architecture that satisfies design constraints of hardware area and execution time, designers should evaluate the design quality of architectures consisting of different functional blocks and different bus architectures. Therefore, fast architecture evaluation method for various architectures and efficient architecture exploration method are required.

SoC Architecture Explorer automatically investigates the design space and reports a number of candidates, which have a trade-off relation between performance and hardware area. SoC Architecture Explorer provides tools for (1) modeling the target system at system-level, (2) specifying architecture constraints, (3) exploring architectures, and (4) checking design quality of a number of candidates.

2. Exploration Mechanism

Inputs to SoC Architecture Explorer are a system–level model of the target system, specification of each IP, range of parameters (bus execution frequency, bus bit width, and the number of buffers), and the constraints of hardware area and the execution time. Outputs are candidates of architecture-level models that are in trade-off relation between hardware area and execution time.

Figure 1 shows the architecture exploration flow of SoC Architecture Explorer. Designers should design system-level model of the target system and specify architecture constraints such as hardware area with partially defined process and channel mappings. Then, described system-level model is profiled. By this profiling, dependency and data flow between processes are recognized. Next, in the pre-scheduling step, a graph representing execution order among data processing and transfers (We call System-Level Execution Order Graph: SL-EOG) is constructed from system-level profile.

In the exploration phase, architecture-level model is constructed. Architecture-level model is comprised of functional blocks, such as processors and ASICs, performing data processing, buses conducting data transfers, and buffers storing data for transfer. Then, in post-scheduling step, Architecture Level Execution Dependency Graph (AL-EDG), which represents execution dependency among data processing and transfers, and process and channel mapping is constructed from architecture-level model and SL-EOG. Finally, execution time is estimated, using generated AL-EDG.

Estimation is performed for each candidate of architecture-level model. All candidates are expressed in parameter set search tree, and exploration is performed by tracing parameter set search tree. However, large system leads the number of architecture candidates to exponential increase. Therefore, exhaustive trace of parameter set search tree wastes enormous time. In our method, candidates are dramatically reduced by branch and bound, based on architecture constraints such as the constraints of hardware area and execution time, and partially defined process, channel mappings, and parameters.

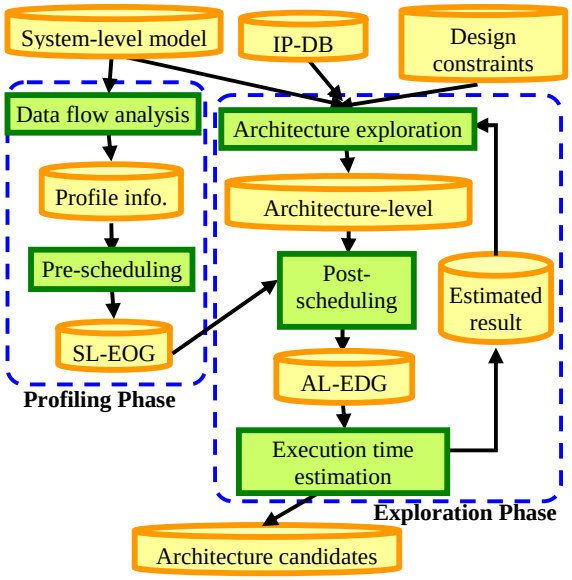

Fig.1: Architecture Exploration Flow.

3. Design Flow in the Tool

The design flow with our tool is as follows:

(1) Model the target system in system-level model.
(2) Set constraints on hardware area and application execution time.
(3) Set architecture parameter candidates of bus execution frequencies, bus bit widths, and the number of data blocks in buffer.

Table1: Experimental Results.

Method	# of explored arch.	# of explored nodes	# of estimation	Time for exploration
Exhaustive	$795x10^{10}$	$1,590x10^{10}$	$795x10^{10}$	2,521years
Proposed	7,769,925	101,384,625	18,654,516	52hours

(4) Set partially defined process and channel mappings.
(5) Run architecture exploration tool. The tool explores all possible parameter sets and outputs the number of architectures, which have a trade-off relation between performance and hardware area. Design space is reduced by branch and bound, based on architecture constraints such as the constraints of hardware area and execution time, and partially defined process, channel mappings, and parameters.
(6) Check design quality of architecture candidates, which the exploration tool outputs. Designers can check each visualized candidate and its data transfer to decide an optimal architecture.

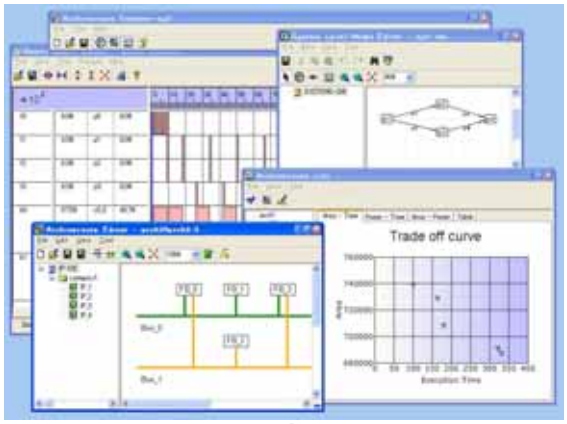

Fig.2: Tool Overview.

4. Experiment

To confirm the effectiveness of our tool, we explored optimal architecture of audio/video encoding system which consists of ten processes and nine channels by both exhaustive search and the proposed method. Following constraints have been imposed on target system: bus bit width is 16 bit or 32 bit, bus execution frequency is 1MHz or 2MHz, and the number of buffers is one or two. However, no constraints have been imposed on area and execution time.

Table 1 shows the experimental results for an audio/video encoding system. **Exhaustive** denotes the exhaustive search that evaluates all parameter set. **Proposed** denotes the proposed method using branch and bound. **# of explored arch.**, **# of explored nodes**, and **# of estimation.** show the number of explored architectures, explored nodes in parameter set search tree, and the number of estimations, respectively. **Time for exploration** shows the time needed for exploration. Because the exploration of an exhaustive search takes a very long time, it is calculated as the product of the average time needed to estimate one architecture (0.011 second), and the number of estimations. The results of the exhaustive search and the proposed method are the same because the proposed method only prunes the nodes that are not the optimal solutions. It is almost impossible to explore the design space by exhaustive search. The experimental results show that the proposed method can greatly reduce the time needed for exploration.

5. Conclusion

In this paper, SoC Architecture Explorer is introduced. SoC Architecture Explorer explores the architecture candidates by tracing the parameter set search tree and then pruning it. All possible architecture candidates are explored, and some architectures are identified as having a trade-off relation between the hardware area and the application execution time. The Experimental results show that SoC Architecture Explorer provides fast design space exploration for IP-based design.

Demonstration

In our booth, SoC Architecture Explorer will appear with a small system example. We will show how to make the system-level model with our editor and how the tool provides design quality of architecture candidates.

References

[1] K. Ueda, K. Sakanushi, Y. Takeuchi and M. Imai, "Architecture-level Performance Estimation for IP-based Embedded Systems", In Proceedings of Design Automation and Test in Europe (DATE 2004), pages 1002-1007, February 2004.
[2] K. Ueda, K. Sakanushi, Y. Takeuchi, and M. Imai, "Architecture-level performance estimation method based on system-level profiling", IEE Proceedings Computers & Digital Techniques, vol. 152, no. 1, pages 12-19, January 2005.

Acknowledgement

This work is partly supported by STARC (Semiconductor Technology Academic Research Center).

Contact Person

Professor Masaharu Imai,
Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, JAPAN
Tel: +81-6-6879-4520, Fax: +81-6-6879-4524