SysteMorph: Dynamic/Online/Adaptive System-Level Optimization for SoC

Yoshimatsu, Norifumi
Fukuoka Industry, Science & Technology Foundation

Yoshida, Makoto
Fukuoka Industry, Science & Technology Foundation

Soga, Takeshi
Fukuoka Industry, Science & Technology Foundation

Shuto, Makoto
Institute of Systems & Information Technologies/KYUSHU

他

出版情報：The 7th International Conference on High Performance Computing and Grid in Asia Pacific Region, pp.442-447, 2004-07. IEEE Computer Society
バージョン：
権利関係：

SysteMoprh: Dynamic/Online/Adaptive System-Level Optimization for SoC

Norifumi Yoshimatsu†, Makoto Yoshida†, Takeshi Soga†, Makoto Shuto††,
Yasuyuki Tanoue†††, Yosuke Fujii†††, Kazuhito Eshima†††, Takanori Hayashida†††,
Kazuaki Murakami†††
†Fukuoka Industry, Science & Technology Foundation
††Institute of Systems & Information Technologies/KYUSHU
†††Department of Informatics, Kyushu University
E-mail: †, ††ngarch@fleets.jp, †††arch@c.csce.kyushu-u.ac.jp

Abstract

This paper describes SysteMorph, a feedback directed dynamic, online and adaptive hardware/instruction set architecture (ISA)/software co-optimization technology. The technology enables to optimize performance, power, and consumption energy for a system dynamically. We describe SysteMorph's elemental technologies those are an online profiling, a dynamic adaptive optimization and a smart hardware. Functionality Morphing is one of the SysteMorph implementations for a system which consists of processor and reconfigurable logic. It reconfigures hardware dynamically then rewrites software to utilize the reconfigured hardware online. We demonstrated the SysteMorph on a reconfigurable processor by implementing Functionality Morphing. Online profiling, online synthesis, and binary rewriting are implemented and verified on the prototypal system. Then we propose a processor suitable implementing SysteMorph using a VLIW execution unit.

1. Introduction

Dynamic optimization is gaining attention as it provides capability to optimize systems. Progress of semiconductor technology enabled to design a large scale, complex functionality SoCs. As the complexity increased it is getting difficult to foresee circumstances which a system operates at its design time and it makes challenging to design a system optimizing performance, power consumption, and energy consumption for them. The dynamic optimization utilizes information that can be known at its run-time. With the dynamic optimization it becomes possible to optimize a system for application programs those behaviors can be changed by their inputs. However there are some issues to realize the dynamic optimization in a system. It must have small impact on system's operation, it is necessary to identify point(s) to apply optimization, and it is necessary to decide how to optimize the point(s), also the system has to have a function to reconfigure executing optimized code adaptively.

We propose SysteMorph as a concept of a system which can be dynamically optimized. SysteMoprh is a technology that optimizes systems dynamically and executes application programs adaptively based on application program's profile information for processor based systems. SysteMorph consists of three elemental technologies those are (1) an online profiling, (2) a dynamic adaptive optimization, and (3) a smart hardware. In this paper we describe Functionality Morphing as one of the SysteMorph's applications which uses reconfigurable hardware to run optimized application programs adaptively. In Sect.2 concept of SysteMorph is described. In Sect.3 the online profiling is described which is applied for the Functionality Morphing. The online profiling is discussed for a processor having a reconfigurable unit. In Sect.4 dynamic optimization technology is described. The dynamic optimization can be applied reconfigurable logic or processor based logic. We described optimization for both type of the systems. In Sect.5 system prototyping is described which uses a reconfigurable processor to implement SysteMorph. In Sect 6 system architecture using a VLIW execution unit is proposed to implement SysteMorph. Sect 7 is

conclusion and future work.

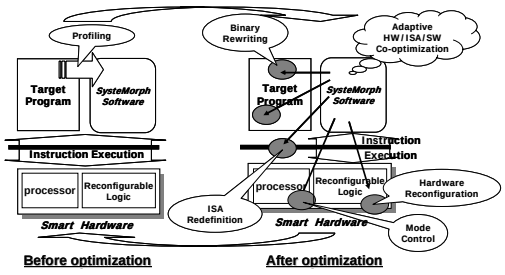
2. What is SysteMorph

2.1. Concept of SysteMorph

SysteMorph is a feedback directed dynamic software/ISA(Instruction Set Architecture)/hardware co-optimization technology mainly adapted in processor based systems. Figure 1 shows concept of SysteMorph. With the SysteMorph, a system is optimized in terms of performance, power consumption, and energy consumption using target application program's profile information while the target application runs. SysteMorph technology consists of following three elemental technologies.

1. Online profiling.
2. Adaptive dynamic optimization.
3. Smart hardware.

The online profiling collects profile information and it derives information which can be used for optimization while a target application runs, then the information is sent to the optimizer as feedback. The adaptive dynamic optimization is a technology to optimize whole system dynamically by using binary rewriting and hardware reconfiguration based on profile information. The smart hardware technology is a hardware configuration technology to perform hardware reconfiguration or mode change to apply the adaptive dynamic optimization.



"**Figure 1.** Concept of SysteMorph"

2.2. Application of SysteMorph

SysteMorph does not have specific implementation for its elemental technologies those are online profiling, dynamic adaptive optimization, and smart hardware. Each technologies can have any configuration to be implemented. In a small embedded system, it may not be practical to build in all hardware and software technology to implement SysteMorph. Therefore the small embedded system can only have a hardware for the online profiling and reconfigurable hardware. Then the optimization may be done by outside of the system.

2.3. Functionality Morphing

Functionality Morphing is one of the SysteMorph implementations for a system that consists of processor and reconfigurable logic. It reconfigures hardware dynamically then rewrites software to utilize the reconfigured hardware online.

To realize the Functionality Morphing the system needs to collect profile information running on it and the system needs to identify point of the application program to be optimized, then run-time hardware reconfiguration is applied. There are some architectural decisions to choose for a system implementing the Functionality Morphing. Those are,

1. Location of the reconfigurable hardware.
2. Architecture of the reconfigurable logic.
3. Timing to apply reconfiguration to the reconfigurable logic.
4. Architecture of the profiler.
5. Granularity of profile information generated by the profiler.

As for the location of the reconfigurable logic, it can be inside of a processor to access its general purpose register or it can be outside of the processor like as co-processors. As the architecture of the reconfigurable logic, there is wide range of choices in granularity in logic block, interconnections, I/O, and memory. And also the conventional processors can be used as the reconfigurable hardware as it reconfigures every instruction cycle to execute them. As for the timing to apply reconfiguration it can be done either during running application program or after running it.

3. Online path profiling

The online profiling is a technology to collect information of application program's behaviors at run-time, while off-line profiling collects whole information regarding to application program's

execution after running the application program. In this paper we mean profiling as the online profiling and we mean profile as information collected by online profiling. The issue of the online profiling is how to get precise profile while the profiling does not affect execution of an application program. The advantage of the online profiling is that it can get a profile for a running application program which behavior depends on its inputs. However in case of applying optimization dynamically it is required to predict application behavior using the profile and it is required to derive hints based on the prediction to apply for the optimization. In general, with more profile, accuracy of the prediction can be improved. In the online profiling it is expected that as the elapsed time to profile is increased more profile is collected then it improves accuracy of the prediction for the execution of the application program after it. However as the elapsed time to profile is increased the rest of application program's execution time to apply optimization is decreased, then the benefit of the optimization is lowered. Therefore it is necessary to run the profiling quick and precise to predict an application behavior.

3.1. Requirement for the online profiler

The online profiler must fulfill these requirements.

1. Small overhead.
2. Low cost to implement.
3. Precise prediction of the application program's behavior.

The profiler works during processor executing application program and the profiler only monitors application program's behavior and it does not affect application's execution itself. Therefore the performance overhead running the profiler has to be small. Also to implement the profiler in a system, cost of hardware, cost of software, and power consumption has to be small as possible as it can. In case hints to optimize are extracted from the profile information to apply for reconfiguring hardware and software, the profiler has to predict the application program's behavior after the point when the online profiler collects information. And if the prediction is not correct, the system after reconfiguration is not optimized for the application program. Therefore it is important to improve precision of prediction to have an optimized system.

3.2. Hot instruction sequence profiler

Hot Instruction Sequence or hot path is a part of instruction sequence in an application program which is frequently executed in its execution. Hot path profiler is an online profiler to profile the hot path. The hot path profilers for the Functionality Morphing are evaluated in [2]. Instruction sequences in the hot path consume most of cycle time in application program's execution. Therefore it is considered to be a key to improve performance by accelerating execution of the path. In Functionality Morphing we use the hot path as profile information. We apply one of the HIS method for a prototypal system described in Sec. 5.

4. Dynamic optimizing compiling

Here we describe about the dynamic optimization technique. As for the smart hardware we consider two kind of hardware to implement the Functionality Morphing. One is a reconfigurable functional unit and another is VLIW execution unit. We describe optimizing technology for both type of hardware.

4.1. Optimizing for reconfigurable functional units

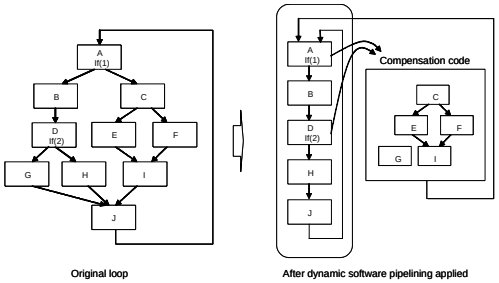
Reconfigurable functional unit is used as a smart hardware to implement Functionality Morphing which executes hot path. The reconfigurable functional unit can achieve high performance especially in computational intensive application. The hot path is synthesized optimizing the instruction sequence to have the performance advantage of the hardware and it is mapped into the reconfigurable functional unit.

4.2. Optimizing for VLIW execution unit

The VLIW execution unit is also used as a smart hardware to implement SysteMorph. The VLIW execution unit has number of independent functional units and it provides higher performance as compared with a processors having smaller number of execution units. Because of ISA based execution, the VLIW execution unit is suitable for executing ISA based instruction sequences. Also it can utilize sophisticated compiling technology.

4.3. Dynamic software pipelining

Software pipelining has been shown to be an effective technique for scheduling loop intensive program on VLIW processors [4]. The principle behind software pipelining is to overlap or pipeling different iterations of the loop body in order to exploit parallelism. However loops including conditional branches are difficult to handle for the software pipelining because there are multiple paths of execution to schedule. To address this problem Dynamic Software Pipelining (D-SWP) is proposed [6] which is a technique to apply for loops with conditional branches. Figure 3 shows outline of D-SWP technique. D-SWP applies to one hot path which is decided from online profile information. To execute other paths a compensate code is generated and it is treated by software pipelining.



"**Figure 2.** Outline of Dynamic Software Pipelining"

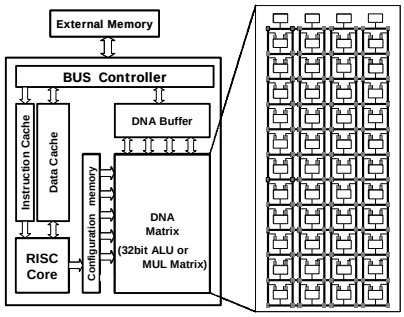
The dynamic software pipelining is used as a dynamic optimization technology for a VLIW based processor implementing SysteMorph.

5. Prototyping Functionality Morphing

We implemented SysteMorph's elemental technologies on a reconfigurable processor to verify the concept of SysteMorph.

5.1. Prototyping using DAP/DNA-HP

We use DAP/DNA-HP [3] from IPFlex to implement SysteMorph as a prototypical system. Figure 3 shows a block diagram of the DAP/DNA-HP. The DAP/DNA-HP consists of 32bit RISC processor called DAP and reconfigurable logic called DNA. The reconfigurable logic basically consists of 32-bit logical elements such as ALU element, buffer memory, interconnects buses, and configuration memory which stores configuration data for the reconfigurable logic. The reconfigurable logic is configured dynamically by writing data into the configuration memory from DAP. With the architectural features of DAP/DNA-HP the processor is capable to be implemented SysteMorph as a prototypical system.



"**Figure 3.** Block diagram of DAP/DNA-HP"

5.2. Online profiling on DAP/DNA-HP

The online profiler is implemented on DAP which is a standard RISC processor. Notably DAP has trace branch mode which is used to profile hot path. In the trace branch mode DAP generates trace exception when a branch instruction is executed. The profiler is implemented in exception handler. To profile hot path, NET (Next Execution Trail) method [1] is proposed. Here we propose BH (Branch History) method for the hot path profiling. In the prototypical system hot path is predicted by using the BH method.

We define Basic block which is an instruction sequence only includes a branch instruction at the bottom. And Basic block Start Address (BSA) which is a start address of the Basic block. In the trace exception handler the profiler finds branch instruction address (BIA) and branch target address (BTA) as the profiling information. The profiler creates a branch history table and a backward branch history table both located in main memory. After running application program hot path profiler examines the table then identify hot path. The branch history table stores BSA, BIA, BTA and count number and the backward branch history table stores BTA and count number. The table

entries are indexed by each Basic block. A flow to predict hot path in the BH method is as follows.

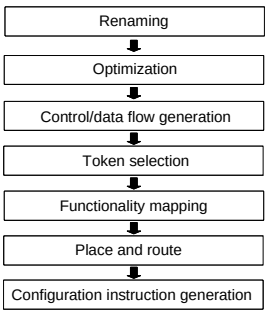
1. As the processor executes branch instructions, BIA and BTA are stored in the branch history table.
2. If a loop (BIA>BTA) is found the backward branch is stored in the backward branch history table.
3. In the backward branch history table if the number of branch is bigger than a threshold number the BTA is chosen as hot path head address.

Then hot path is detected using the hot path head address.

4. In the branch history table it is checked that if any entry matching the hot path head address and BSA.
5. If it finds matches in (4), the hot path head address is selected as hot path address. If there are multiple entries matched then entry which has bigger count number is selected.
6. The branch history table is searched to find entries matching hot path address and BSA.
7. If the hot path address matches hot path head address it is selected as hot path. If it does not match (6) and (7) are repeated until a backward branch is found.

5.3. Online synthesis on DAP/DNA-HP

Online synthesis is performed to generate a configuration data for DNA (a reconfigurable logic) to execute the hot path on it. Figure [4] shows a flow of the online synthesis.



"**Figure 4.** Online synthesis flow"

The instruction sequence of the hot path is synthesized and is compiled into configuration data which is loaded into configuration memory in the DNA.

1. Renaming: Hot path's instruction sequence is converted to SSA (static single assignment) form.
2. Optimization: Removing unused or redundant instructions.
3. Generate control/data flow graph: Dependency between the instructions are checked.
4. Token selection: token is selected. The token represents validation of data.
5. Functionality mapping: The instructions are converted into DNA functions.
6. Place and route: The mapped functions are placed and routed.
7. Generating configuration instruction: The mapping information, placement information, and routing information are converted into a format of instruction to configure DNA.
8. Generating control instruction: Instructions to pop and push DAP's general register and instructions to start and stop DNA are generated.

The application program code in the memory is update to run the DNA. The code to switch the execution to DNA is called at the beginning of hot path. When DNA finished executing the hot path, instruction execution returns to the point after the hot path.

5.4. Results

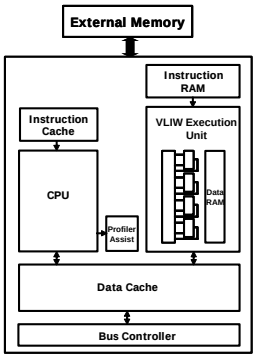
The elemental technologies for the Functionality Morphing are successfully implemented on the DAP/DNA-HP. Profiling information is taken during application program execution. After running the application program hot path is detected and the detected paths are synthesized then mapped into reconfigurable logic while the application program's binary is updated to use reconfigurable execution. Then application program runs on reconfigurable logic for the hot path parts.

6. Prototyping using VLIW execution units

The VLIW execution unit is used to accelerate hot path execution. Miyajima et al proposed HyperScalar processor architecture [5]. In the HyperScalar

processor architecture, a processor encompasses advantage of RISC type processor, VLIW, and vector processor. We consider the architecture as the base of the smart hardware for Functionality Morphing. Figure [5] shows block diagram of the proposed architecture of a VLIW based processor. The processor combines a RISC type processor and a VLIW execution unit. The VLIW execution unit has performance advantage with the parallel execution with a number of independent datapath. The VLIW approach is suitable for running hot path because of its ISA based execution. And it can utilize sophisticated compiling technology as the D-SWP technique is promising to accelerate execution of the instruction sequence. Also in ISA based time multiplexed execution achieves high utilization of die area. Besides, optimization such as the profiling and the dynamic software pipelining can be done on the VLIW execution unit to hide overhead.

In the VLIW based system SysteMorph is realized with online profiler, dynamic optimizing compiling of hot path for VLIW execution unit applying D-SWP technology, and rewriting binary of application program. A hardware assist for online profiling is considered. It stores BIA, BTA into profiler memory besides CPU executes application program. Then profiler reads the profiler memory to detect hot path.



"**Figure 5.** Block diagram of VLIW based system"

7. Conclusion and future work

SysteMorph is proposed and the elemental technologies are described. The elemental technologies are implemented and the technology is demonstrated by prototyping using a reconfigurable processor. The technology is promising. We research on system architectures such as a system using VLIW and a system using reconfigurable functional unit running SysteMorph technology. More study in application program side has to be done to find suitable architectural choice for the technology.

8. Acknowledgement

This research is supported by a lot of people. We especially thank Professor Murakami who gave us advices. This research was partly supported by a grant of the Cooperative Link of Unique Science and technology for Economy Revitalization (CLUSTER) of Ministry of Education, Culture, Sports, Science and Technology (MEXT).

9. References

[1] E. Duesterwald and V. Bala, "Software Profiling for Hot Path Prediction: Less is More", Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2001),pp.202-211, Nov., 2000.

[2] Takanori Hayashida, Kazuaki Murakami, "Evaluating Online Hot Instruction Sequence Profilers for Dynamically Reconfigurable Functional Units", pp.901-908, IEICE TRANS. INF. & SYST., VOL. E86-D, NO5, MAY 2003.

[3] IPFlex DAP users manual, http://www.ipflex.com/

[4] Monica Lam, "Software Pipelining: An Effective Scheduling Technique for VLIW Machines", In Proceedings of the ACM SIGPLAN '88 Conference on Programming Language Design and Implementation, Jun. 1998.

[5] Hiroshi Miyajima, Tetsuo Hironaka, Yasuhiko Satoh and Kazuaki Murakami, "Hyperscalar Processor Architecture –Principle of Operations and Performance Evaluation-", In Proc. Of IPSJ JURNAL, Vol.36, No8, pp.1964-1975.

[6] Yasuyuki Tanoue and Kazuaki Murakami, "Proposal and Evaluation of Dynamic Software Pipelining (in Japanese)", *IPSJ Technical Report, ARC*, Vol.2002, No.81, pp.85-90, Aug. 2002.