Effectiveness of the ASIP Design System PEAS-III in Design of Pipelined Processors

Akira Kitajima[†] Makiko Itoh[†] Jun Sato[‡] Akichika Shiomi[*] Yoshinori Takeuchi[†] Masaharu Imai[†]

[†] Dept. of Informatics and Mathematical Science
Osaka University
1-3 Machikaneyama, Toyonaka,
Osaka, 560-8531, Japan
Tel: +81-6-6850-6626
Fax: +81-6-6850-6627
peasv@vlsilab.ics.es.osaka-u.ac.jp

[‡] Dept. of Elect. Eng.
Tsuruoka National College of Tech.
104 Sawada, Inoka, Tsuruoka
Yamagata, 997-8511, Japan
Tel: +81-235-25-9086
Fax: +81-235-24-1840
jun@tsuruoka-nct.ac.jp

[*] Dept. of Computer Science
Shizuoka University
3-5-1 Johoku, Hamamatsu
Shizuoka, 432-8011, Japan
Tel: +81-53-478-1485
Fax: +81-53-478-1485
shiomi@cs.inf.shizuoka.ac.jp

Abstract— In this paper, the effectiveness of the ASIP (Application Specific Instruction set Processor) design system PEAS-III is evaluated through experiments. Examples in experiments are a MIPS R3000 compatible processor, DLX, a simple RISC controller, and PEAS-I core. While they are simple in-order pipelined processors, they have enough facilities for real embedded system design. Through experiments, easiness of design and modification for improvement and design quality in terms of performance and hardware cost are discussed. It has been confirmed that the design method used in PEAS-III is effective to design space exploration for simple pipelined processors.

I. INTRODUCTION

In SoC (System-on-a-Chip) development, design of ASIPs (Application Specific Instruction set Processors) is challenging since various design constraints must be satisfied for the requirement of applications. In addition, shorter time to market is preferable especially in the embedded system area. Thus, to obtain a processor suitable for the requirement of an application in a short term, a framework for effective design space exploration is desired.

In the recent research, several ASIP design systems have been proposed. However, most of them have limitations for flexibility of customization. For example, while Tensilica, Inc. has the design service of customizing ASIP named Xtensa [1], flexibility of the configuration of Xtensa is restricted to their model, e.g., upper limit of the number of customized instructions, or the number of pipeline stages, etc.

The PEAS-III system is a workbench for ASIP [2, 3]. It can generate an HDL description of a processor from a micro operation description which represents behavior of each instruction by each pipeline stage. With the PEAS-III system, various customizing processors can be designed in a short time. Furthermore, modification for improvement such as adding/removing several instructions or applying different functional units for a operation is easy.

In this paper, the effectiveness of PEAS-III is evaluated through experiments. In experiments, several processors have been designed with PEAS-III. Examples in experiments are a MIPS R3000 compatible processor, DLX, a simple RISC controller, and PEAS-I core. While they are straightforward pipelined processors, i.e., processors with single pipelining, in-order issue and possibly out-of-order completion, they have enough facilities for real embedded system design. In experiments, easiness of design and modification, and design quality in terms of performance and hardware cost are evaluated.

According to the experimental results, the effectiveness of the method used in PEAS-III has been confirmed. The proposed method can be effective for shortening the design period, and for design space exploration for straightforward pipelined processor design.

The rest of the paper is organized as follows. Section II describes the design flow in the PEAS-III system. Section III is devoted to experiments for several processor designs. In Section IV, the effectiveness of the method used in PEAS-III is discussed with the experimental result. Finally, conclusions and future work are presented in Section V.

II. DESIGN FLOW IN PEAS-III

In this section, an overview of the design flow in PEAS-III [3] is presented. The main function of PEAS-III is depicted in Fig. 1. Each part in this figure is described along the design flow in the followings.

First of all, a designer describes an instruction set architecture. The instruction set architecture includes the width of an instruction code, instruction code formats for each instruction type, etc. In PEAS-III, these works can be done with GUI.

In this phase, basic architecture of the processor is also determined by the designer. The parameters of architecture include the number of pipeline stages, the number of delayed branch slots. Currently, the target architecture of PEAS-III is limited to straightforward pipelined processors which includes mechanisms such as pipeline interlock for multi-cycle operation or interrupt handling. Harvard architecture is assumed as memory interface.

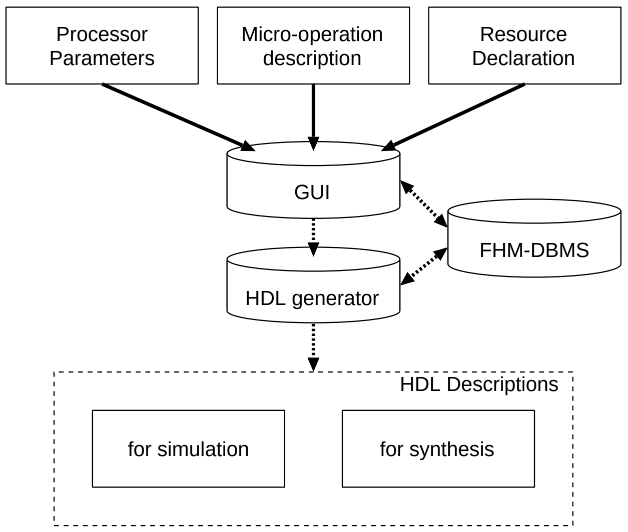

Fig. 1. Main function of PEAS-III.

Then, the designer determines what kinds of modules are used in the implementation. In this paper, these modules are called *resources*. For the selection of resources, a database called FHM-DB [4] are used. In FHM-DB, parameterized modules called FHMs are registered. Parameters for an FHM include bit width of inputs/outputs, algorithm of the operation such as carry-lookahead for addition, etc. Therefore, the designer can use various kinds of resources for design space exploration by specifying these parameters.

At the next phase, the designer describes a micro-operation description. The micro-operation description represents behavior for each instruction by each pipeline stage. This description also includes behavior for interrupts. Note that the behavior is defined independently against each instruction, i.e., inter-instruction operations such as pipeline interlock are described implicitly. Thus, the designer can concentrate on describing just the behavior of each instruction. Furthermore, addition or removal of instructions can be easily handled.

In this phase, the behavior of each instruction is represented by the data flow among resources. What kind of operation is performed is also specified for multi-function modules like ALU.

After these works have been done, the designer can obtain HDL descriptions with the HDL generator of PEAS-III. Two types of HDL descriptions are available with PEAS-III. One is for simulation purpose and the other one is for logic synthesis purpose. The HDL description for simulation can be used for validation and performance evaluation. In the description, contents of resources are described at the behavior level. Simulation time for the processor description is about ten to hundred times shorter than the case that contents of resources are described at the gate level. On the other hand, the HDL description for logic synthesis can be used for logic synthesis. Contents of the resources are described at the RT-level or the gate level. This description can be input to a logic synthesis tool.

In the generation of these HDL descriptions, the data path and the control logic including pipeline interlock or interrupt manipulation are automatically constructed from input descriptions. Thus, the designer can easily obtain the various kind of processors by modifying the input descriptions.

III. Experiments

Several processors have been designed with PEAS-III for various kinds of evaluation. Three experiments have been carried out. The first one is design of a MIPS R3000 compatible processor and DLX. The second one is design of a RISC controller. The third one is design of a PEAS-I core.

A. MIPS R3000 and DLX compatible processors

First, a MIPS R3000 [5] compatible processor was designed. Then it was modified into DLX [6] for evaluation of the easiness of design in PEAS-III.

At the first step, a subset of MIPS R3000 instruction set was implemented. The number of implemented instructions is 52 out of 74 instructions of all instructions on MIPS R3000. Coprocessor instruction and interrupt instruction were not implemented in this experiment. Required time for design is about 8 hours.

The results of logic synthesis are summarized in Table I. The column “#” denotes the number of components in the processor. The “Cell Area” indicates the component area without wiring area, and the “Total Area” indicates the component area including wiring area. The “Frequency” means the maximal clock frequency of the corresponding component. “User specified resources” are the resources that explicitly declared by the designer. “Registers,” “selectors,” and “controller” are the automatically introduced resources by the generator. “Sum of the above” means just the sum value of all components above in the column. “Processor” means the logic synthesis result as a processor.

From these experimental results, it is confirmed that automatically generated part does not so much affect for area and performance of the processor. Area of the generated part is about 25% of the whole processor. Frequency of the introduced resources including the controller is relatively high, hence they do not include the critical path alone.

At the second step in this experiment, a subset of DLX is implemented based on the R3000 implementation above. The number of implemented instructions is 51 out of all instructions 91. Similarly to the case of the R3000 compatible processor, coprocessor instruction and interrupt instruction were not implemented in this experiment. The reuse ratio for DLX design from the description of the R3000 compatible processor is 59% since both architecture have many similar instructions. Required time for modification is about 3 hours.

The amount of descriptions for both the R3000 compatible processor and DLX is shown in Table II. The amount of input description for PEAS-III is about less than one sixth of the case

TABLE I
RESULTS OF LOGIC SYNTHESIS FOR R3000

component	#	Cell Area (gates)	Total Area (gates)	Frequency (MHz)
user specified resources	12	21,833.5	33,232.2	70.97
registers	23	6,305.8	7,772.6	934.58
selectors	10	1,163.5	1,822.6	437.80
controller	1	1,662.3	2,474.8	110.01
sum of the above	45	30,965.0	45,302.1	70.97
processor	1	30,883.0	49,643.2	50.28

using Design Compiler (0.5 μm CMOS library)

TABLE II
COMPARISON OF THE AMOUNT OF DESCRIPTIONS FOR R3000 AND DLX

	Input description	generated HDL description
R3000	1181	5811
DLX	1196	6259

(unit: word)

of the corresponding generated HDL description. It is clear that PEAS-III reduces the designer's load.

B. A RISC controller

This experiment is aimed for comparison between designs with conventional method and designs with the method used in PEAS-III. The original controller which is used for image processing was designed by manual RT-level description. A compatible controller was designed with PEAS-III in this experiment.

This RISC controller has Harvard architecture. The instruction width is 24 bits. The number of instructions is 54. The controller consists of three stage pipeline. It has synchronous interrupt facility.

An undergraduate student designed this controller with PEAS-III. He had no experience of processor design with PEAS-III at the beginning of this experiment.

Design proceeded as the following way.

1. He learned the usage of PEAS-III.
2. He designed the controller with 32 bits for instruction width.
3. He modified the design to fit 24 bits for instruction width.

The time required for learning PEAS-III is about seven hours. The learning includes reading manuals and trying design with a sample processor attached to PEAS-III.

In the first design, he designed the 32 bits instruction width for ease of the code assignment, because the code assignment

TABLE III
WORK LOAD FOR DESIGNING A RISC CONTROLLER

works	first design (hour)	modification (hour)
selecting resources	3	1
determining instruction set architecture	12	8
writing micro operation description	40	2
modifying errors	2	2
total	58	13

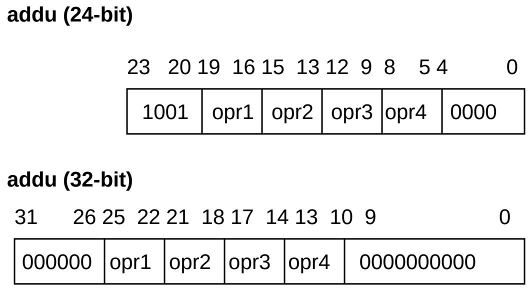

Fig. 2. Difference of instruction code of ADDU between 24-bit and 32-bit in RISC controllers.

of the original instruction set was not given. He implemented all 54 instructions. The work load for this work is shown in the column "first design" of Table III. Each row corresponds to the design phase described in Section II. The total required time is 58 hours. Though he was not familiar with PEAS-III, he designed a processor in a few days.

In the second design, he modified the first design concerning about the instruction width. The main work was the modification of instruction format. While some trivial modifications were required, the most part of the micro-operation description was reused.

Examples of instruction code assignment of both 24-bit and 32-bit are shown in Fig. 2. In this example, code assignments of ADDU (add unsigned) instruction are shown. Named fields like "opr1" in Fig. 2 is referred in the micro-operation description.

An example of a portion of a micro-operation description is shown in Fig. 3 In this example, the micro-operation description of ADDU instruction is shown. It consists of behavior of each stages. At the stage 2, the value of operands are referred using the names "opr2" and "opr3." As shown in this example, modification of instruction codes can be done without modification of the micro-operation description.

The column "modification" of the Table III shows the required time for this work.

The design quality in terms of area and available clock frequency are also examined in this experiment. The generated HDL description of 32-bit version of a RISC controller and the

stage 1	IR := IMEM[PC];
	PC.inc();

stage 2	DECODE(IR); $sr1 := freg.read0(opr2); $sr2 := freg.read1(opr3);

stage 3	($result, $aluflg) := ALU.addu($sr1, $sr2, ’0’);
	aluflg := $aluflg(2) & $aluflg(3);
	freg.write0(opr1, $result);

Fig. 3. Micro-operation description of ADDU in RISC controllers.

TABLE IV
COMPARISON OF THE DESIGN WITH PEAS-III AND WITH CONVENTIONAL METHOD FOR A RISC CONTROLLER

		PEAS-III	conventional method
instruction width (bit)		32	24
work load (hour)		58	420
gate size	[50MHz]	12.7k	14.3k
	[108MHz]	12.9k	14.6k

(using CMOS 0.25 μm library)

HDL description of real RISC processor were synthesized under the same condition. Table IV shows the result. Two target frequencies 50 MHz and 108 MHz was set up for logic synthesis. Given proper constraint for logic synthesis, both controller have achieved these frequencies. Note that the original controller has several instructions that were added to the original instruction set for extension, and they were not implemented in the controller designed with PEAS-III. Though rough comparison of the values for the areas is not justified enough, there seems no remarkable difference.

C. PEAS-I core

PEAS-I core is a processor generated with the PEAS-I system [7]. PEAS-I system can generate an optimal processor for a given application program from predefined instruction set. Predefined instruction set consists of a primitive instruction set and optional instructions. The primitive instruction set contains basic instructions which most processors have. Instructions in the primitive instruction set can be categorized into arithmetic instructions, data transfer instructions, and execution sequence control instructions.

In this experiment, the existing design and new one designed with PEAS-III are compared. First, a PEAS-I core from a primitive instruction set was designed with PEAS-III. The instruction set contains 85 instructions. Then, this processor was extended by adding multiply instructions.

The result of the first step is shown in Table V. The column “original” corresponds to the case of the original design, and the column “with PEAS-III” corresponds to the case of the design with PEAS-III. Work load for the design with PEAS-III is about one third compared to the original one. Others are almost same.

TABLE V
RESULT OF PEAS-I CORE DESIGN

	original	with PEAS-III
work load (hour)	96	32 $^{(*1)}$
lines in the HDL description	6431	7194 (1038 for MOD)
maximum delay (ns)	9.80	9.74
# of gates	22,247	26,970

$^{(*1)}$ includes the time of learning about the system.

TABLE VI
DELAY AND SIZE OF PEAS-I CORE WITH MULTIPLY OPERATION

Design	delay (ns)	Area (gate)
under 100MHz		
SR	17.93	49567.8
SL	9.77	49946.5
CR	9.78	67905.8
CL	9.72	75089.6
under 200MHz		
SR	17.69	50784.4
SL	7.68	51828.9
CR	6.93	69351.8
CL	6.07	76577.2

S: sequential circuit implementation, C: combinational circuit implementation; R: using ripple carry adder, L: using carry-lookahead adder.

Next in this experiment, this processor was extended with additional multiply with signed/unsigned operations using PEAS-III. The result of the logic synthesis is shown in Table VI. To implement the multiply instructions, several functional units for multiply operation can be selected. Using PEAS-III, this selection is done by specifying the parameters for the operation. The designer can evaluate several designs easily.

IV. DISCUSSIONS

In this section, the effectiveness of the method used in PEAS-III is discussed with experimental results. There are following advantages in the method compared with conventional methods.

- shorter design period
- easiness of design space exploration
- easiness of design

Since the method uses automatic HDL generation, the quality of the design in terms of the speed and area may be inferior to the design with conventional methods. The above issues are confirmed and perspectives for future improvement are given in the followings.

A. Design Period

Design period for ASIP with PEAS-III is about three to seven times shorter than that for conventional RT-level design as shown in examples. This improvement owes to abstraction of a description of a processor.

Higher abstraction also contributes the easiness of modification of the design. As the description of the design is more concrete, it is difficult for the designer to grasp the whole design in detail. Thus, in view of validation of the design, abstraction is essential. Furthermore, easiness of validation leads to shortening design period.

B. Easiness of design space exploration

Since the objective of design space exploration is to find a best processor configuration for the target application, design space exploration is quite difficult. There are few approach to overcome this problem. One is to restrict the flexibility and to narrow the design space. The other is to keep the flexibility and to support the designer. The approach taken in PEAS-III is based on the latter one.

In the approach, designers can design various processors in a short time as described in experiments. The flexibility of processor architecture and module configuration enable to design various kinds of processors.

Support for more general processor architecture which includes VLIW architecture could lead to expansion of the design space.

For more efficient support of design space exploration, various kinds of effective estimation are needed. It enables several kinds of optimization. That can be great help for the designer.

C. Easiness of design

Easiness of design is another important issue since designers tend to have little time to learn the usage of CAD tools. In the experiment of a RISC controller described in Section III.B, an undergraduate student was able to design a processor with seven hours for learning. It indicates that the usage of PEAS-III is not so difficult.

With more improvement of PEAS-III such as better error handling, design with PEAS-III can be easier than the case of using the current PEAS-III system.

D. Quality of the design

As shown in the example of Section III.B and Section III.C, the number of gates in the design with PEAS-III tends to increase compared to the design with conventional methods. However, the difference is not crucial for most applications.

Because most part of the processors is shared by existing modules in the database, the disadvantage originated in automatic generation is rather small. Flexible modules with better quality are required for designs under strict constraints.

V. Conclusions

In this paper, the effectiveness of the design method used in PEAS-III have been examined. Through experiments, it has been confirmed that the design method is effective to architecture exploration for straightforward pipelined processors. While design quality in terms of area and performance is slightly inferior compared to the processor designed by experienced designers, the turn around time or design man-hour is quite shorter than conventional methods.

Future work includes a support for more general architecture, e.g., support of pipelined operations which are often appeared in media processors, or support of VLIW architecture which is often appeared in DSPs. Development of model generation algorithm and optimization algorithm is one of major problems.

Acknowledgments

We greatly appreciate to Mr. Takayuki Minemaru, Mr. Hiroshi Imanishi, Mr. Hiroyori Morimoto, Mr. Osamu Ogawa, Mr. Masanobu Mizuno, and Mr. Michiaki Muraoka at Matsushita Electric Industrial Co., Ltd. for cooperation about the evaluation of a RISC controller.

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

References

[1] R. Gonzalez, “Xtensa: A configurable and extensible processor,” *IEEE Micro*, vol. 20, no. 2, Mar./Apr. 2000.

[2] Makiko Itoh, Yoshinori Takeuchi, Masaharu Imai, and Akichika Shiomi, “Synthesizable HDL generation for pipelined processors from a micro-operation description,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E83-A, no. 3, pp. 394–400, Mar. 2000.

[3] Makiko Itoh, Shigeaki Higaki, Jun Sato, Akichika Shiomi, Yoshinori Takeuchi, Akira Kitajima, and Masaharu Imai, “PEAS-III: An ASIP design environment,” in *Proceedings of 2000 IEEE International Conference on Computer Design: VLSI in Computers & Processors (ICCD2000)*, Sept. 2000, pp. 430–436.

[4] Masaharu Imai, Yoshinori Takeuchi, Takafumi Morifuji, and Eiichiro Shigehara, “Flexible hardware model: A new paradigm for design reuse,” in *APCHDL ’98*, Seoul, Korea, July 1998, Invited Talk.

[5] Gerry Kane, *mips RISC Architecture*, Prentice-Hall, Inc., New Jersey, 1988.

[6] John L. Hennessy and David A. Patterson, *Computer Architecture a Quantitative Approach*, Morgan Kaufmann Publishers, INC., second edition, 1996.

[7] Jun Sato, Alauddin Y. Alomary, Yoshimichi Honma, Takeharu Nakata, Akichika Shiomi, Nobuyuki Hikichi, and Masaharu Imai, “PEAS-I: A hardware/software co-design system for ASIP development,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E77-A, no. 3, pp. 483–491, Mar. 1994.