RAPID PROTOTYPING OF JPEG ENCODER USING THE ASIP DEVELOPMENT SYSTEM: PEAS-III

Shinsuke Kobayashi, Kentaro Mita

Graduate School of Engineering Science,
Osaka University
{s-kobays,k-mita}@ics.es.osaka-u.ac.jp

Yoshinori Takeuchi, Masaharu Imai

Graduate School of Information Science
and Technology, Osaka University
{takeuchi, imai}@ist.osaka-u.ac.jp

ABSTRACT

In this paper, JPEG encoder application, one of the DSP applications, was implemented using the ASIP development system: PEAS-III. Instructions for JPEG encoder, such as DCT instruction, and butterfly instructions, were added to the initial design. Area, performance, and execution cycles of processors were calculated using generated HDL description, compiler, and assembler by PEAS-III. From experimental results, 12 architectures can be designed in 160 hours, and designer can select an optimal architecture that satisfies design constraints considering hardware cost, clock frequency and execution cycles.

1. INTRODUCTION

There are two approaches to realize application domain specific embedded systems. One is to use general purpose processors and ASICs (Application Specific Integrated Circuits), and the other is to use ASIPs (Application Specific Instruction set Processors). One of the advantages of the second approach is that better implementations can be realized by introducing cost-effective instructions suitable for specific applications. In the ASIP design, it is also important to search for a processor architecture that matches the target application. To achieve this goal, it is essential to estimate design quality of architecture candidates that have different instruction sets, pipeline stage counts, and combinations of hardware resources. Here, design quality means area, performance, and power consumption of a design. Because there are many architectural parameters, there exist a huge number of processor architecture candidates, which makes it difficult to find an optimal architecture in a short design time. In this case, the ASIP development system plays an important role to estimate design quality and develop target processors.

In this paper, JPEG encoder application was designed using the ASIP design methodology. Instructions for JPEG encoder, such as DCT instruction, and butterfly instructions, were added to the initial design. Area, performance, and execution cycles of processors were calculated using HDL descriptions, compiler, and assembler. From experimental results, various architectures can be designed in a short time, and designers can select an optimal architecture that satisfies design constraints.

The rest of this paper is organized as follows. In section 2, related work is surveyed, and the PEAS-III system, one of the ASIP development systems, is introduced. The case study and experimental result are discussed in section 3, and examined in section 4. Finally, section 5 concludes this paper and future work is described.

Thanks to Japan Novel and ACE Associated Compiler Experts bv for providing the CoSy compiler kit.

2. ASIP DEVELOPMENT SYSTEM

2.1. Related Work

Conventional approaches to ASIP development can be classified into two kinds. First one is a “parameterized generic processor core” such as PEAS-I[1], Satsuki[2], MetaCore[3], CASTLE[4], Xtensa[5] and so on. Their processor models usually have basic instruction sets and a synthesizable ASIP description is generated by adding predefined or user defined instructions to the basic instruction set. Architectures of these processors ease to develop the parameterized retargettable compiler, but in many cases have little flexibility on pipeline structure and instruction variations. Hence, the variety of architecture candidates by these systems is limited with respect to pipeline stage count, instruction format and micro-operation for each pipeline.

Another approach is based on “processor specification languages” such as nML[6], ISDL[7], LISA[8], FlexWare[9], EXPRESSION [10], AIDL[11], and Hamabe, *et al.*[12]’s approach. The processor specification languages nML, ISDL, LISA, FlexWare and EXPRESSION are originally developed to design a compiler, simulator and other tools for software development. The instruction behavior and the structure of the target processor are described in these specification languages. Compiler and other tools can be generated using these languages, but it is difficult to generate **synthesizable** HDL descriptions from these languages. Because it is not enough resource specification including timing specification, control signal information and so on to generate HDL descriptions. On the other hand, AIDL and Hamabe, *et al.*’s approach are developed to produce HDL descriptions. The instruction behavior, the timing relations of pipeline stages and the structure of the processor core are described in these languages. Using these languages, HDL descriptions of the target processor can be generated. However, the modification cost is larger than those of other approaches based on processor specification language, because detail information is described when designers use these languages.

2.2. PEAS-III

The PEAS-III system [13, 14] is one of the ASIP development systems, which generates not only synthesizable HDL descriptions but the target compiler and the target assembler. PEAS-III is based on processor specification language approach. Hence, wide range of processor architecture can be described using the PEAS-III sys-

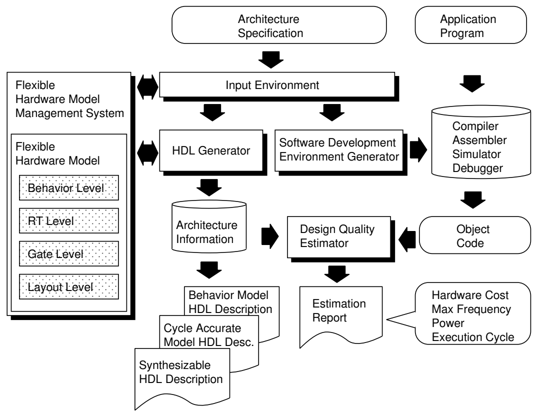

Fig. 1. Overview of the PEAS-III system.

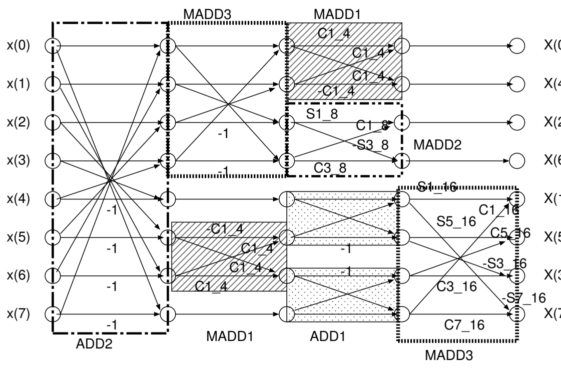

Fig. 2. Data Flow of Chen DCT (1-dimensional 8 points).

tem. The PEAS-III system has well parameterized resource models, Flexible Hardware Models (FHM). FHM-DBMS (DataBase Management System) produces the resource specification to generate HDL and the target compiler. When designers would like to change the features of resources, they only change the parameters of FHM. Moreover, designers describe processor specification through the PEAS-III input environment using GUIs. Resources and other architecture parameters are specified using GUIs. These features reduce the modification cost and encourage design reuse.

Overview of the PEAS-III system is shown in Fig. 1. Processor architecture specification is written in the input environment, which encourages efficiently input. The processor specification description includes: (1) architecture parameters such as pipeline stage counts, the number of delayed branch slots, (2) declaration of resources included in the processor, such as ALUs and register files, (3) instruction format definitions, (4) behavior and micro-operation descriptions of instructions, and (5) interrupt definitions including cause conditions and micro-operation description of interrupts. The architecture description is given from the input environment to the HDL generator and the software development environment generator. The HDL generator and the compiler generator uses FHM, which is parameterized resource model. Since FHM is used in HDL and compiler generation, designers can change the characteristics of resource only by changing the parameters of each resource.

It is the advantage of the PEAS-III design that a processor architect can design the synthesizable HDL and the target compiler rapidly. Since execution cycles, clock frequency and hardware cost can be evaluated in the early design step, designers can find an optimal architecture in the short design time.

3. CASE STUDY

3.1. Architecture Candidates

Several kinds of parameters are defined in JPEG specification. In this case study, 8 bit precision baseline algorithm was selected. Huffman coding was selected as VLC and VLD. In the following section, architecture candidates are described, and experimental results are explained.

DCT and IDCT are implemented using Chen DCT algorithm [15], which is one of the famous algorithms reducing multiplications and additions. Data flow of Chen DCT is shown in Fig. 2.

```
quantization (short int ∗input,
              short int ∗output,
              short int ∗qtable) {
  short int *inputPtr = input;
  for ( ; inputPtr < input + 64; inputPtr++ ) {
    if ( ∗inputPtr > 0 ) {
      ∗output = (∗inputPtr + (∗qtable >> 1)) / ∗qtable;
    } else {
      ∗output = (∗inputPtr - (∗qtable >> 1)) / ∗qtable;
    }
    output++; qtable++;
  }
}
```

Fig. 3. C Source Code of Quantization.

Here, the x(i) denotes element of input matrix, X(i) denotes transformed element. Ci_j and Si_j denote $\cos(\frac{i\times\pi}{j})$ and $\sin(\frac{i\times\pi}{j})$, respectively. Using Chen algorithm, multiplication times are reduced from 64 to 16, and addition times are reduced from 56 to 26 in 1 dimensional 8 points DCT. IDCT can be implemented using inverse of DCT. Hence, multiplication and addition times in IDCT are reduced as much as those of DCT.

In DCT and IDCT implementation, several approaches exist, and there approaches were studied in this case study: **Sequential Instructions Approach** is software approach that stands for software implementation. All of the algorithm is processed by primitive instructions of processors. **DCT Instruction Approach** is hardware approach that stands for hardware unit implementation. All of the algorithm is processed by hardware. **Butterfly Instructions Approach** is hybrid approach that stands for implementation using fine grain instructions. The part of the algorithm is processed by hardware, and the other part of the algorithm is processed by primitive instructions of processors. These approaches are expected to have trade-offs between hardware cost and performance.

In quantization implementation, several approaches exist, which is the same as DCT implementation. Fig. 3 shows the C source code of quantization. From Fig. 3, quantization divides the element by the element of quantization table. Hence, the performance of divider affects the execution cycles of quantization. In this case study, the implementation algorithm of divider was changed.

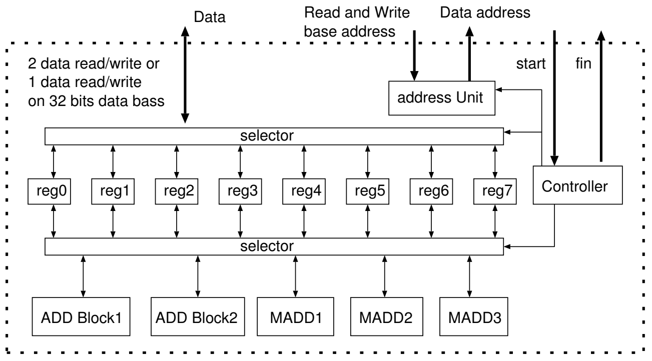

Fig. 4. DCT/IDCT Unit.

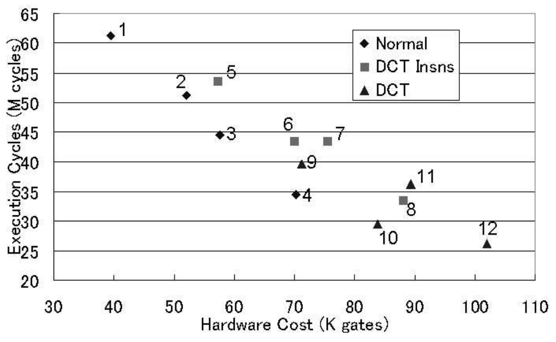

Fig. 5. Trade-offs Between Hardware cost and Execution Cycles When JPEG Encoder was Executed.

Table 2. Design Time.

	Time (hour)
C source code implementation	130
DCT unit design	60
Total	190
Base processor design	12
Registration of DCT unit and Convolution blocks to FHM-DBMS	1
Instruction addition	1
Implementation algorithm selection	0.1
Others	150
Total	164.1

3.2. Input Image

In this evaluation, a standard image (Lenna) was used as an input image. The image size was 256 × 256 pixels and the sampling factors of each component were as follows: horizontal sampling factors of Y, U, V were 4, 1, 1, and vertical sampling factor were 4, 1, 1, respectively.

3.3. DCT/IDCT Unit

Fig. 4 shows the DCT/IDCT unit that processes 2 dimensional (2-D) 8 points DCT/IDCT. The DCT/IDCT unit fetches the data from the data memory to the internal registers. Calculation units are executed sequentially using these registers. Each block processes the data in one cycle, then DCT/IDCT is executed in four steps. Because this unit processes 2-D 8 points DCT/IDCT, each step is executed twice.

3.4. Additional Instructions

Additional Instructions are as follows: (1) **DCT** instruction executes the procedure of DCT. This instruction uses the DCT unit described in section 3.3. (2) **MADD1** instruction calculates the MADD1 block in Fig. 2. MADD1 instruction takes 2 operands as input and write back to same operand registers. (3) **MADD2** instruction calculates the MADD2 block in Fig. 2. MADD2 instruction takes 2 operands as input and write back to same operand registers. (4) **MADD3** instruction calculates the MADD3 block in Fig. 2. MADD3 instruction takes 4 operands as input and write back to same operand registers. MADD3 unit has 2 operation mode to change coefficients.

3.5. Processor Organization

Processor organization in this case study is shown in Table. 1. Normal denotes base instruction set that is sub set of MIPS-R3000 instruction set. DCT insns denotes instruction set added MADD1, MADD2 and MADD3 instructions. DCT denotes instruction set added DCT instruction. The implementation algorithm of multiplier is sequential type that executes 32 cycles and array type that executes 1 cycle. On the other hand, the implementation algorithm of divider is sequential type that executes 34 cycles, and array type that executes 1 cycle.

3.6. Experimental Results

Fig. 5 shows trade-offs between hardware cost and execution cycles when JPEG encoder has been executed. Horizontal axis is hardware cost, and vertical axis is execution cycles. The number of each point corresponds to Table 1. From Fig. 5, the trade-off between hardware cost and execution cycles exists when instructions are added and the hardware unit implementation algorithms are changed.

The design time of the case study is shown in Table 2. From table 2, about ten hours were spent using the PEAS-III system. Here, the reason why the implementation algorithm selection time is short is only changing FHM parameters to select implementation algorithm. From this result, the hardware description and the target compiler can be designed in a short design time. 130 hours were spent implementing JPEG codec using C source code. 60 hours were spent DCT unit design. Others include debug time and simulation time and synthesizing time to evaluate the processor core. The time of JPEG codec implementation and DCT unit design do not depend on our environment.

4. DISCUSSION

When an application such as DSP application is implemented using ASIPs, designers consider trade-offs between hardware cost and performance. In this case study, Normal (1, 2, 3, 4) and DCT (10, 12) architecture candidates can be selected when these pro-

Table 1. Processor Cores and Their Execution Cycles of JPEG Application.

	multiplier	divider	area (K gates)	Max Freq. (MHz)	Exec Cycles (M cycles)
1. Normal	seq(32)	seq(34)	39.43	151	61.28
2. Normal	seq(32)	array	52.1	22.5	51.19
3. Normal	array	seq(34)	57.59	44.5	44.54
4. Normal	array	array	70.19	43.3	34.45
5. DCT insns	seq(32)	seq(34)	57.3	149	53.57
6. DCT insns	seq(32)	array	70.0	23.0	43.48
7. DCT insns	array	seq(34)	75.5	44.5	43.52
8. DCT insns	array	array	88.0	23.0	33.43
9. DCT	seq(32)	seq(34)	71.17	151	39.62
10. DCT	seq(32)	array	89.35	22.4	29.53
11. DCT	array	seq(34)	83.86	43.3	36.25
12. DCT	array	array	101.93	43.3	26.17

Library: 0.14 μm CMOS Standard Cell Library.

cessors execute the same clock frequency. Generally, it is well known that the design time of hardware description, compiler and assembler requires several months or at least several weeks. However, it is too long to meet a requirement of the design time in design space exploration. On the other hand, when designers use the ASIP development systems that have been explained in section 1, either software development environment or hardware description is produced in a short time, but the other part, for example processor cores for software development environment, must be developed separately. The advantage of the PEAS-III system is that compiler, assembler and hardware description are generated at the same time. Furthermore, the modification cost of the design is low, and hardware modules such as DCT unit can be reused easily, because designers only select modules from FHM-DBMS as resources. Using the PEAS-III system, designers can evaluate processors and select an optimal architecture in a short design time.

5. CONCLUSION

In this paper, JPEG encoder implementation using the PEAS-III system is described, which is one of the ASIP development system. Instructions for JPEG encoder, such as DCT instruction, and butterfly instructions, were added to the initial processor. Area, performance, and execution cycles of processors were calculated using the generated HDL description, compiler, and assembler. From experimental results, 12 architectures can be designed in a short time. Moreover, the design quality of each processor including hardware cost, execution cycles of application, and clock frequency was evaluated using the PEAS-III system efficiently. Future work includes instruction set simulator, profiler, and debugger generation.

6. REFERENCES

[1] J. Sato, A. Y. Alomary, Y. Honma, T. Nakato, A. Shiomi, N. Hikichi, and M. Imai, "PEAS-I: A Hardware/Software Codesign System for ASIP Development," *IEICE Trans. Fundamentals*, vol. E77-A, no. 3, pp. 483–491, Mar. 1994.

[2] B. Shackleford, M. Yasuda, E. Okushi, H. Koizumi, H. Tomiyama, and H. Yasuura, "Satsuki: An Integrated Processor Synthesis and Compiler Generation System," *IEICE Trans. Inf. & Syst.*, vol. E79-D, no. 10, pp. 1373–1381, Oct. 1996.

[3] Jin-Hyuk Yang, Byoung-Woon Kim, Sang-Jun Nam, Jang-Ho Cho, Sung-Won Seo, Chang-Ho Ryu, et al., "MetaCore: An Application Specific DSP Development System," in *35th DAC*, 1998, pp. 800–803.

[4] Raul Campasano and Jörg Wilberg, "Embedded System Design," *Design Automation for Embedded Systems*, vol. 1, no. 1-2, pp. 5–50, Jan. 1996.

[5] Tensilica, "Xtensa," http://www.tensilica.com.

[6] Andreas Fauth, "Beyond tool-specific machine descriptions," in *Code Generation for Embedded Processors*. 1995, pp. 138–152, Kluwer Academic Publishers.

[7] George Hadjiyiannis, Pietro Russo, and Srinivas Devadas, "A methodology for accurate performance evaluation in architecture exploration," in *36th Design Automation Conference*, June 1999, pp. 927–932.

[8] Stefan Pees, Andreas Hoffmann, Vojin Zivojnovic, and Heinrich Meyr, "LISA - Machine Description Language for Cycle-Accurate Models of Programmable DSP Architecture," in *36th Design Automation Conference*, 1999, pp. 933–938.

[9] Pierre G. Paulin, Clifford Liem, Trevor C. May, and Shailesh Sutawala, "Flexware: A flexible firmware development environment for embedded systems," in *Code Generation for Embedded Processors*, 1995, pp. 65–84.

[10] Ashok Halambi, Peter Grun, Vijay Ganesh, Asheesh Khare, Nikil Dutt, and Alex Nicolau, "EXPRESSION: A Language for Architecture Exploration through Compiler/Simulator Retargetability," in *DATE 99*, Mar. 1999, pp. 485–490.

[11] T. Morimoto, K. Saito, H. Nakamura, T. Boku, and K. Nakazawa, "Advanced Processor Design Using Hardware Description Language AIDL," in *ASP-DAC'97*, 1997, pp. 387–390.

[12] M. Hamabe, A. Nose, N. Togawa, M. Yanagisawa, and T. Ohtsuki, "A Generation System for Hardware Description of Pipelined Processors," in *Tech. Report of IEICE, VLD97-117*, 1997, pp. 33–40, (in japanese).

[13] 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.

[14] Shinsuke Kobayashi, Kentaro Mita, Yoshinori Takeuchi, and Masaharu Imai, "Design Space Exploration for DSP Applications using the ASIP Development System PEAS-III," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing*, May 2002, vol. 3, pp. 3168–3171.

[15] W. H. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, pp. 1004–1009, 1977.