AE32000B: a Fully Synthesizable 32-Bit Embedded Microprocessor Core

Hyun-Gyu Kim, Dae-Young Jung, Hyun-Sup Jung, Young-Min Choi,
Jung-Su Han, Byung-Gueon Min, and Hyeong-Cheol Oh

In this paper, we introduce a fully synthesizable 32-bit embedded microprocessor core called the AE32000B. The AE32000B core is based on the extendable instruction set computer architecture, so it has high code density and a low memory access rate. In order to improve the performance of the core, we developed and adopted various design options, including the load extension register instruction (LERI) folding unit, a high performance multiply and accumulate (MAC) unit, various DSP units, and an efficient coprocessor interface. The instructions per cycle count of the Dhrystone 2.1 benchmark for the designed core is about 0.86. We verified the synthesizability and the area and time performances of our design using two CMOS standard cell libraries: a 0.35-μm library and a 0.18-μm library. With the 0.35-μm library, the core can be synthesized with about 47,000 gates and operate at 70 MHz or higher, while it can be synthesized with about 53,000 gates and operate at 120 MHz or higher with the 0.18-μm library.

Keywords: Integrated circuit, embedded microprocessor, synthesizable core, SoC.

Manuscript received Jan. 15, 2003; revised July 26, 2003.

This work has been supported by the COSAR.

Hyun-Gyu Kim (phone : +82 2 545 4898 ext. 222, email: babyworm@adc.co.kr) is with R&D Center, Advanced Digital Chips Inc., Seoul, Korea, and Graduate School of Korea University, Seoul, Korea.

Dae-Young Jung (email: dyjung@adc.co.kr), Hyun-Sup Jung (email: hsjung@adc.co.kr), Young-Min Choi (email: choiym@adc.co.kr), Jung-Su Han (email: jshan@adc.co.kr), and Byung-Gueon Min (email: bgmin@adc.co.kr) are with R&D Center, Advanced Digital Chips Inc., Seoul, Korea.

Hyeong-Cheol Oh (email: ohyeong@korea.ac.kr) is with School of Engineering, Korea University at Seo-Chang, Korea.

I. Introduction

In the era of deep sub-micron technologies, a single chip, known as a system-on-a-chip (SoC), integrates multiple systems that were previously constructed as one or more printed circuit boards. An SoC may include a couple of microprocessors, various functional blocks, a bus system, and several I/O elements. Today's trend toward an extremely short time-to-market is forcing SoC designers to reuse available design blocks, called cores [1]. In the design of an SoC, selecting the microprocessor cores is an especially important part of the design process because they control the whole system.

Reusable cores are divided into hard IPs and soft IPs. The hard IP is a complete physical circuit description of the core. The hard IP is optimized and usually the best solution for a specific process, but it is less portable than its soft counterpart. In these days, all IPs start off as soft IPs, because soft IPs can be more flexibly ported to and implemented in any process. A soft IP is a synthesizable register transfer level (RTL) code that has been functionally verified and has to be written in such a good coding style that it is reusable. In general, it takes more effort to optimize a soft IP to the target process than its hard counterpart [2].

In this paper, we present the architecture and the performance of a fully synthesizable 32-bit microprocessor core called the AE32000B. Three objectives motivated the AE32000B project. First, we intended to achieve high performance using the so-called "brainiac" approach [3], because it is generally hard for a synthesizable core to operate at a high frequency. Second, we aimed at designing a core that is fully synthesizable within a reasonably small area without losing much the power of its instruction set. Third, we intended to develop an interfacing scheme that could be used to integrate

various types of coprocessors with the microprocessor core.

The AE32000B is based on the extendable instruction set computer (EISC) architecture, so it has high code density and a low memory access rate [4], [5]. It is equipped with a 32-bit ALU, 32-bit barrel shifter, and a multiply and accumulate (MAC) unit that can handle a MAC operation (32×32+64=64) in a single cycle. It also has various functional units to support DSP applications.

In the embedded microprocessor system, a relevant coprocessor is often an efficient solution for a specific application. Therefore, in order to operate the system properly, an efficient coprocessor interface is essential. The main objective of the coprocessor interface for the AE32000B system was to support various types of coprocessors efficiently.

To present the performance of our design, we use the Dhrystone 2.1 benchmark. The instructions per cycle (IPC) count of the Dhrystone 2.1 benchmark for the designed core is about 0.86. We verified the synthesizability and the area/time performances of our design using two CMOS standard cell libraries: a 0.35-μm library and a 0.18-μm library. With the 0.35 μm library, the core could be synthesized with about 47,000 gates and operate at 70 MHz in the worst case, while it could be synthesized with about 53,000 gates and operate at 120 MHz in the worst case with the 0.18-μm library.

The rest of the paper is organized as follows. We briefly discuss the EISC architecture in section II and introduce the microarchitecture of the AE32000B in section III. Then, our verification method and the performance of AE32000B are described in sections IV and V. Lastly, section VI summarizes the paper.

II. The EISC Architecture

In an embedded microprocessor system, code density and chip area are two major design issues, since these costs are more important in this arena. However, many 32-bit embedded microprocessors suffer from poor code density. In order to address this problem, some RISC-based 32-bit microprocessors adopt 16-bit compressed instruction set architectures, such as ARM THUMB [6] and MIPS16 [7]. This approach provides better code density but needs some mechanisms to extend the insufficient immediate field and to provide backward compatibility with their previous architectures, which can result in extra hardware overhead. Moreover, these architectures have difficulty in utilizing their registers efficiently [6], [7].

The EISC architecture takes a different approach while achieving high code density and a low memory access rate [4], [5]. The EISC uses an efficient fixed length 16-bit instruction set for 32-bit data processing. To resolve the problem of insufficient immediate operand fields in a concise way, EISC uses an independent instruction called load extension register (LERI), which consists of a 2-bit opcode and a 14-bit immediate value. The LERI instruction extends the immediate field by loading an immediate value to a special register called the extension register. By using LERI instructions, the EISC architecture can make the program code more compact than the competing architectures, since the frequency of LERI instructions is less than 20% in many programs. In addition, EISC does not require instructions for switching its processor mode between the compressed instruction mode and the normal instruction mode. (For competing architectures, extra mode-changing instructions are added to use specific instructions such as MAC instructions.)

As a result, the EISC architecture has higher code density than ARM THUMB and MIPS16 architectures. The code density of the EISC is 6.5% higher than the ARM THUMB and 11.5% higher than the MIPS16. This is a considerable improvement in two aspects. First, the EISC core can save a large portion of the fetch-related power consumption. Second, the EISC core and its program memory can be integrated into a smaller die. In addition, the EISC architecture reduces its data memory access rate by fully utilizing its 16 registers. The data memory access rate of the EISC is 35.1% less than that of the ARM T HUMB and 37.6% less than that of MIPS16 [4]. Thus, the EISC can reduce both instruction references and data references. Reducing the memory accesses would bring reduction in power consumption related to the memory accesses and also lessen the performance degradation caused by the speed gap between the processor and the memory. Moreover, the EISC-based system can be implemented in a smaller die, because the memory circuit can be made smaller.

III. Microarchitecture

The AE32000B core uses a simple 5-stage scalar pipeline (Fig. 1). The five stages are instruction fetch (IF), instruction decode (ID), execution (EX), memory access (MEM), and write-back (WB) stages.

1. Instruction Fetch and LERI Instruction Folding

The AE32000B core has an eight-entry instruction queue between the IF stage and the ID stage. The instruction queue decouples the instruction fetch stage and the execution portion (ID, EX, MEM, WB) of the pipeline. Instructions are prefetched into the instruction queue to minimize the number of the pipeline stalls caused by instruction cache misses. In addition, the AE32000B uses the queue to fold in LERI instructions.

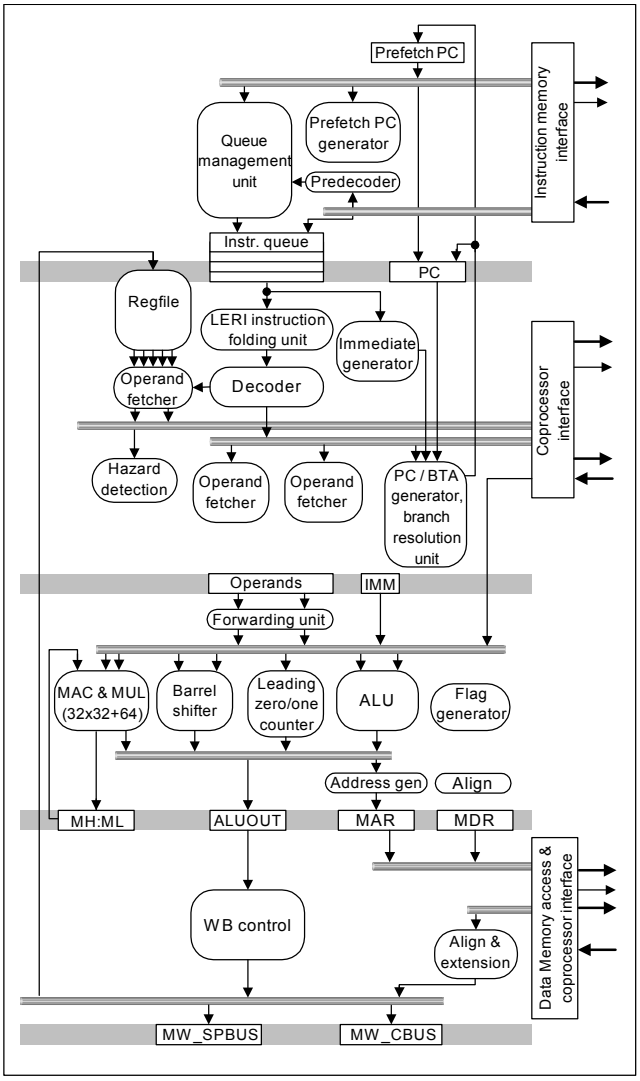


Fig. 1. Microarchitecture of the AE32000B core.

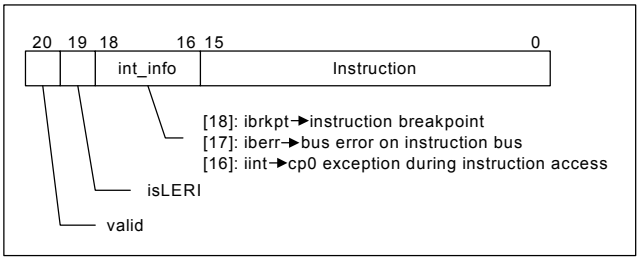


Fig. 2. An entry of the instruction queue.

Figure 2 shows the structure of each entry of the instruction queue. It consists of a valid bit, a LERI checking bit, an information field to indicate whether an interrupt occurred during an access to the instruction memory, and an instruction. A prefetched instruction is predecoded to determine whether it is a LERI instruction. The LERI instruction folding unit fetches up to four valid instructions from the instruction queue. Then, by checking the *isLERI* field, this unit checks whether and where non-LERI instructions exist. If any valid non-LERI instruction exists among the fetched instructions, the folding unit sends a non-LERI instruction to the instruction register and extends the LERI instructions to the extension register. This unit also checks the *int_info* field for generating the requested exceptions.

2. DSP Features

As the need of signal processing grows, many 32-bit embedded microprocessors tend to be equipped with DSP functions. Many DSP algorithms perform multiply and accumulation operations intensively [14]. To accelerate the operation, the AE32000B core has a powerful MAC operation unit that can process a MAC operation (32×32+64=64) in a single cycle. The MAC unit can also be used for a multiply operation (32×32=64). Moreover, the core is equipped with a 32-bit barrel shifter and a leading one/zero counter. With this DSP functionality, the AE32000B core can process DSP applications efficiently without an additional coprocessor.

3. Debugging Capability

In order to debug an SoC system, it is sometimes necessary but difficult to watch the internal state of the embedded microprocessor in the system. Over the past few years, a number of studies have been carried out on the method of debugging embedded microprocessor cores. A common approach is to use an in-circuit-emulator (or ICE breaker), for which an extra processor emulates the embedded microprocessor. The use of an expensive ICE breaker increases the overall verification cost. Hence, currently, some microprocessors are integrated in a chip with an ICE unit such as the background debug mode debugger [9]. This approach can decrease the cost of the verification phase but increases the cost of the chip itself.

The AE32000B has a special operation mode, called the on-silicon ICE (OSI)) mode, to inexpensively provide debugging capability. In the OSI mode, the debugger of the AE32000B can access the internal registers including special purpose registers.

The OSI module can be implemented in the system coprocessor called the coprocessor 0 (CP0). This unit is comprised of simple comparators, registers, and a communication port. The OSI breaker module can trace up to eight breakpoints or watchpoints. The breaker module monitors the addresses accessed during the instruction fetch stage and the data memory access stage, and it generates a proper break exception if any break condition is met. When the

core detects a break exception, it switches its operating mode to the OSI mode to activate the debugging capability. The OSI module can be configured to use a parallel port or a serial port to communicate with the host computer that is used for debugging the system remotely.

4. Coprocessor Interface

The coprocessor interface of the AE32000B is a passive one. Figure 3 shows the coprocessor interface for the AE32000B. The *cpctrl* signal is used to identify the operation of the coprocessor. There are four types of instructions for the coprocessor interface: the instructions for specifying commands to the coprocessors, the instructions for directing data transfer between the AE32000B and a coprocessor, the instructions for directing data transfer between a coprocessor and a memory (under the control of the AE32000B), and the instructions for polling.

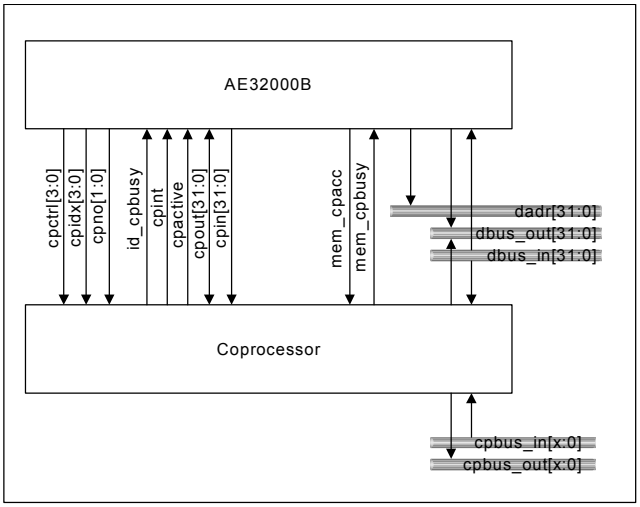


Fig. 3. Coprocessor interface.

In order to send a coprocessor instruction to a coprocessor, the AE32000B must fetch the instruction from the instruction memory and feed it to the coprocessor, since the coprocessor is a passive device. A coprocessor instruction is encapsulated in the *CPCMDn* instruction. The width of the coprocessor instruction can be up to 32 bits.

During the data transfer operation between the AE32000B and a coprocessor, the pipeline of the coprocessor is synchronized with the pipeline of the AE32000B to avoid structural hazards. The coprocessor uses the *id_cpbusy* signal to stall the pipeline in the core.

All the memory access operations of the coprocessor are managed by the AE32000B. When the core decodes an *LDCn* or an *STCn* instruction, it sends a synchronization request to the coprocessor. The coprocessor detects hazards about the destination (source) register and determines whether it has to send the *id_cpbusy* signal to the AE32000B. In the MEM stage, the AE32000B sends the *mem_cpacc* signal as a synchronization request and controls the memory bus to access the memory. Then, the coprocessor sends the *mem_cpbusy* signal to the microprocessor as an acknowledgement to complete the memory access operation. If a wider bus is needed in a coprocessor, the *cpbus_in* bus and the *cpubs_out* bus can be connected to the coprocessor.

Sometimes a coprocessor operates independently to increase the overall performance. The coprocessor commands are passed to the coprocessor in the ID stage. Then, the coprocessor can run without stalling the pipeline in the AE32000B. Two methods can be used for checking results: polling and interrupts. In order to support polling, the AE32000B has two instructions: *GETCn* and *EXECn*. The former modifies the zero flag in the AE32000B by setting a specific status bit in the coprocessor. The latter generates coprocessor exceptions. In order to generate an interrupt, the coprocessor sends the exception information to the external interrupt controller, which then can also be used to check the completion of an operation.

Another synchronization issue in the coprocessor interface is how to handle the exception on the microprocessor when the coprocessor runs independently. For this purpose, we use the

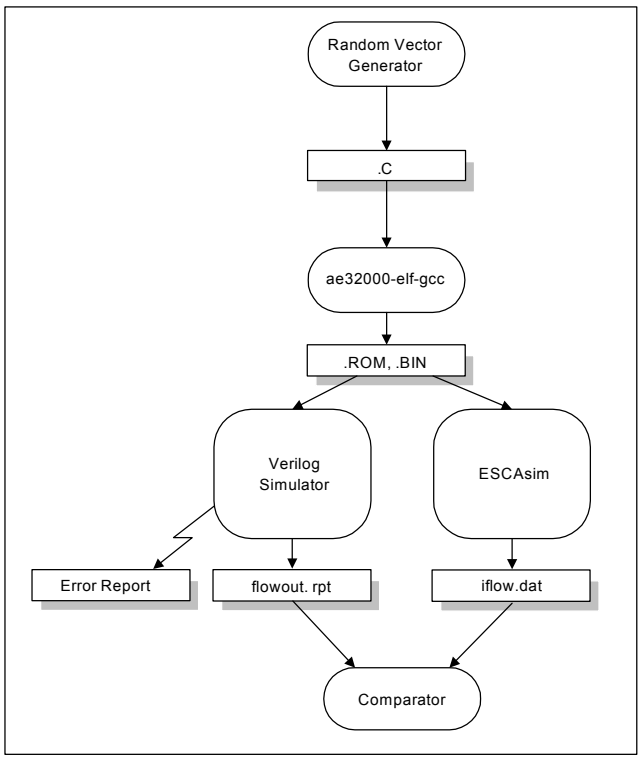


Fig. 4. Flow of the constrained random test.

cpactive signal. The AE32000B cannot enter the exception state for interrupts when *cpactive* is asserted. Some exceptions of higher priority can abort the operation in the coprocessor to initiate the exception handling routine.

IV. Verification

In order to verify the functionality and the synthesizability of the designed AE32000B core, we used three verification techniques. First, we used a static code checking method, called Lint, to detect syntax error and coding misses in the early stage. Second, we used a focused method by using a set of test vectors based on a sophisticated checklist. Lastly, we used a random checking with a constrained random vector generator.

1. Lint Checking

Lint is a static code checking method [2]. In the design of the AE32000B, the lint method was widely used in the early design stage to check syntax errors and frequently occurring coding misses such as inconsistencies in assign statements or port widths. Moreover, we used the lint method to check RTL coding style, as specified in [2] and [10], to increase reusability and synthesizability. A good coding style reduces coding misses and improves the quality of the design. As a result, we were able to detect many errors in the early design stage and make the AE32000B model consistent in coding style. It made our code easy to verify and modify.

2. Focused Test

In the functional verification phase, we used a focused test [1]. Since the quality of the test vectors in this test closely depends on the test engineer's intention, we developed a sophisticated checklist, based on the specification and the bug list from the previous project. We also added some self-checking codes to reduce verification efforts.

3. Constrained Random Test

Random checking, commonly used for verifying the functionality of cores, can check the corner cases that are hard to find by the focused test. The drawback to random checking methods is that they need many test vectors to achieve the target coverage [11]. Thus, an automated checking method is needed in the random test. We decided to use an instruction-based test vector that examines the instruction specification and the constraints of memory accesses. Thus, we developed an in-house constrained random vector generator for the AE32000B.

The whole verification flow of the constrained random test is described in Fig. 4. The random vector generator makes the test vectors in .C files. The generator can confine the memory access range and the sequence of instructions to avoid unwanted (prevented in the specification) situations, such as the breakdown of a stack area. The generated test vector is included in an in-line assembly code and has a self-checking code. To support automated verification, we used a cycle accurate C simulator, called the ESCAsim, as a reference model. We fed the compiled test vectors to both the Verilog RTL model and ESCAsim. Both the Verilog simulator and ESCAsim trace the contents of the program counter during the runs. We compared the results to verify the correctness of the Verilog model. We also detected some errors by using self-checking code in the test vectors.

V. Performance Estimation

1. Impact of LERI Folding

In this section, we present the impact of the LERI instruction folding on the performance of the designed core. First, we compiled the Dhrystone 2.1 benchmark and the quicksort program to see the frequency of the LERI instructions. The frequency of the LERI instructions was closely related to the offset length and the size of immediate values. From our experiment, we found that the frequency was about 11.2% in the compiled programs.

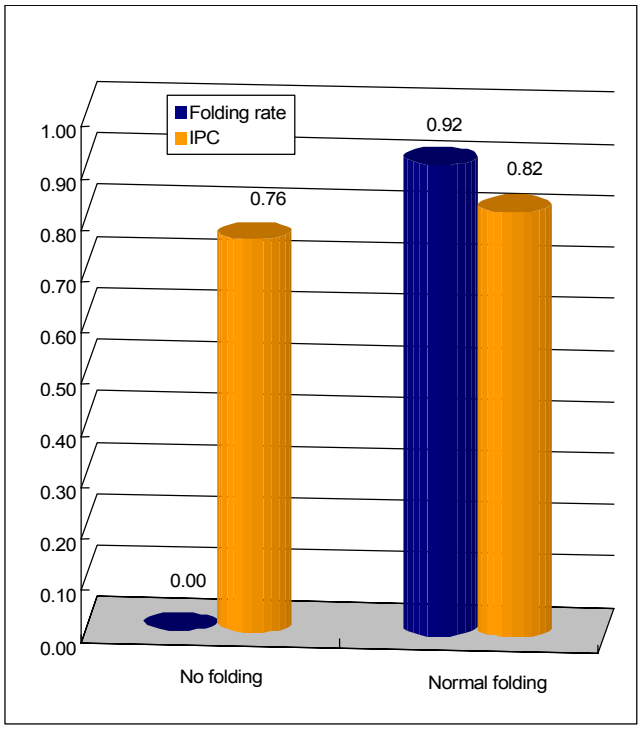


Fig. 5. Impact of the LERI instruction folding.

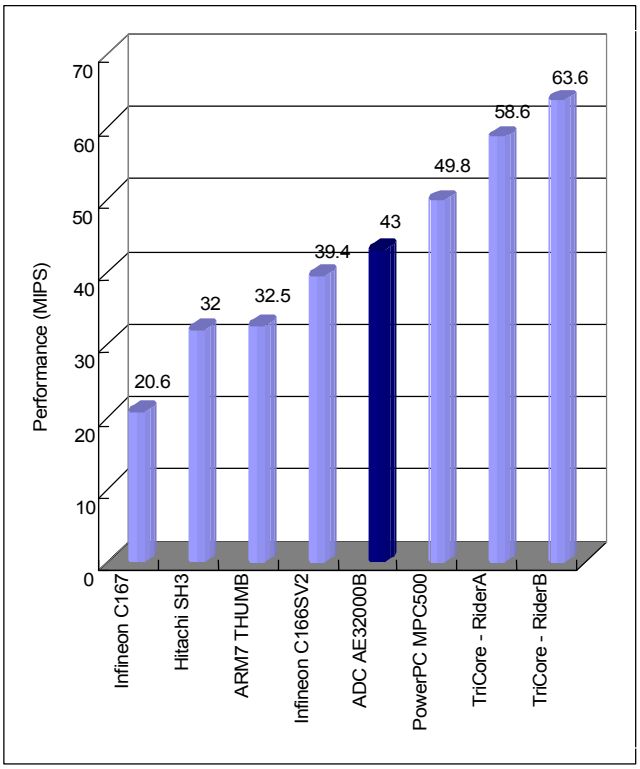


Fig. 6. IPCs of Dhrystone 2.1 for the AE32000B core and other embedded cores under a perfect memory system environment (50 MHz normalized, zero-wait memory system).

The impact of the LERI folding is summarized in Fig. 5. Without the LERI instruction-folding unit, the average IPC count is about 0.76, while it is about 0.82 with the LERI folding unit. The designed LERI folding unit can fold and hide about 92% of the LERI instructions and increase the IPC count by about 7.9%.

The cost effectiveness is more important in the embedded market. We synthesized the LERI instruction-folding unit using a 0.35-μm static CMOS standard cell library and found that the unit can be implemented with about 1900 gates, which is only 4% of the whole AE32000B.

2. Performance

In order to compare the performance of the AE32000B with those of other microprocessors, we use the Dhrystone 2.1 benchmark, which is widely used for embedded processors. We compiled the benchmark using a GNU C compiler for the AE32000B, called the *ae32000-elf-gcc*, with a –O2 optimization option. The IPC of the Dhrystone 2.1 benchmark for the designed core was about 0.86, which we normalized for 50 MHz, to compare it with those for other processors and compared to those for other processors in million instructions per second (MIPS) metric [12]. These results are derived under perfect memory configuration (a zero-wait memory system). The result is summarized in Fig. 6.

3. Synthesis Results

In order to verify the synthesizability of our design and to suggest its operation frequency and area guideline, we synthesized the design core using two target libraries: a static 0.35-μm CMOS standard library (*STD90*) by Samsung Electronics and a static 0.18-μm CMOS standard cell library (*sb18os120_anam*) by Avanti. We compiled the designed the AE32000B core using a Synopsys synthesis and optimization tool [13]. Table 1 shows the results of the synthesis and the simulation with the worst case condition—slow process, 125 OC junction temperature, and 10% lower supply voltage than the normal condition. Under the best case condition, our design could operate at 280 MHz with 1.98 V in the 0.18-μm process.

Table 1. Operation frequency and implementation cost.

Technology	Operation frequency	Gate usage
0.18 μm	Higher than 120 MHz	Less than 53,000 Gates
0.35 μm	Higher than 70 MHz	Less than 47,000 Gates

VI. Conclusions

This paper presented a 32-bit fully synthesizable microprocessor core called the AE32000B. This microprocessor has high code density and low memory access rate. It has some unique features: the LERI folding method for improving performance and the OSI for debugging internal status. It also has DSP functionality, which can be used in various DSP applications. The AE32000B has an efficient coprocessor interface to support various coprocessors.

Our experimental results showed that the LERI folding unit considerably improves performance. Analysis of the Dhrystone 2.1 benchmark showed that the AE32000B achieved an IPC count of about 0.86. This result demonstrates that the performance of the AE32000B is above middle range in the embedded market. In the synthesis result, the AE32000B can operate higher than 120 MHz in 0.18 μm and higher than 70 MHz in 0.35 μm. It uses about 47,000 to 53,000 gates in the implementation.

In the design of the AE32000B, we focused on increasing IPC count with a small die size. Hence, operating frequency was a little slower than competitors, but we believe the overall

performance of the AE32000B outperforms competitors. ADChips, Inc. plans to use the new synthesizable core using the "speed-demon" approach to achieve higher operation frequency.

References

[1] P. Rashinkar, P Paterson, and L. Singh, *System-on-a-Chip Verification: Methodology and Technique*, Kluwer Academic Publishers, Boston, 2001.

[2] M. Keating and P. Bricaud, *Reuse Methodology Manual: For System-on-a-Chip Designs*, 2nd ed., Kluwer Academic Publishers, Boston, 1999.

[3] L. Gwennap, "Brainiacs, Speed Demons, and Farewell," *Microprocessor Report*, vol. 13, Issue 17, Dec. 1999.

[4] H. Lee, P. Beckett, and B. Appelbe, "High-Performance Extendable Instruction Set Computing," *Proc. of 6th ACSAC* 2001, Jan. 2001, pp. 89-94.

[5] H.C. Oh, H.G. Kim, H.S. Jung, J.W. Lee, B.J. Kim, J.Y. Jung, B.G. Min, J.Y. Lim, H. Lee, and K.H. Kwon, "AE32000: An Embedded Microprocessor Core," *Proc. of 2nd AP-ASIC* 2000, Aug. 2000, pp. 255-258.

[6] *Introduction to Thumb*, ARM Ltd., http://www.arm.com/Documentation/Overviews/Thumb_intro/.

[7] K.D. Kissell, *MIPS16: High-Density MIPS for the Embedded Market*, MIPS Tech.. Inc., http://www.mips. com/Documentation/MIPS16whitepaper.pdf.

[8] L.T. Clark, E.J. Hoffman, J. Miller, M. Biyani, Y. Liao, S. Strazdus, M. Morrow, K.E. Verlarde, and M.A. Yarch, "An Embedded 32-b Microprocessor Core for Low-Power and High-Performance Applications," *IEEE J. of Solid-State Circuits*, vol. 36, no. 11, Nov. 2001, pp. 1599-1608.

[9] *Using Background Debug Mode for the M68HC12 Family*, Motorola, http://www.motorola.com/.

[10] SIPAC, *Verilog Coding Guideline v1.0*, http://www.sipac.org.

[11] N. Dohm, C. Ramey, D. Brown, S. Hildbrandt, J. Huggins, M. Quinn, and S. Taylor, "Zen and the Art of Alpha Verification," *Proc. of ICCD*-98, Oct. 1998, pp. 111-117.

[12] K.D. Maier, "C166S V2–A Single Cycle 16-Bit Microcontroller and DSP Core for Next Generation Systems on Chips," *Proc. of 4th COOL Chips*, Apr. 2001, pp. 79-93.

[13] *Synopsis, Version 2000.11*, Synopsis Inc., Mountain View, CA, Nov. 2000.

[14] J. Lee, J. Lee, M.H. Sunwoo, S. Moh, and S. Oh, "A DSP Architecture for High-Speed FFT in OFDM Systems," *ETRI J.*, vol. 24 no. 5, Oct. 2002, pp. 391-397.

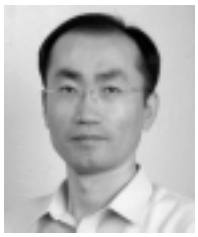
Hyun-Gyu Kim received his BS and MS degrees in electronics and information engineering from Korea University in Seoul, Korea in 1998 and 2000. He is currently working toward the PhD degree in Korea University. He joined Advanced Digital Chips Inc. in Seoul, Korea, in 2002 where he is currently an Associative Research Engineer of R&D Center. His research interests include computer arithmetic and architecture, SoC design, and verification methodologies.

Dae-Young Jung received his BS in information and communication engineering from Chung-Buk National University in Cheong-Ju, Korea, in 2000. He Joined the Advanced Digital Chips, Inc. in Seoul, Korea, in his graduate years where he is currently a Research Engineer of R&D Center. His research interests include microprocessor design, SoC verification, and design.

Hyun-Sup Jung received his BS and MS degrees in electronics and information engineering from Korea University in Seoul, Korea, in 1999 and 2002. He is currently an Associate Research Engineer at Advanced Digital Chips, Inc. in Seoul, Korea, where he has worked since his graduate years. He is currently involved in the research and development of an EISC microprocessor.

Young-Min Choi received the BS and MS degrees in telecommunication and information engineering from Hankuk Aviation University in Gyeonggi-do, Korea, in 1999 and 2001. He joined Advanced Digital Chips, Inc. in Seoul, Korea in 2001 where has been engaged with the design and verification of the EISC processor. He is currently studying and designing TLB for virtual memory system.

Jung-Su Han received the BS degree in control and instrumentation engineering and MS degree in information and communication engineering from Chonbuk National University, Korea, in 1995 and 1997. From 1997 to 1999, he was with Anam S&T, where he worked on digital circuit design and ASIC design. He joined Advanced Digital Chips, Inc. in Seoul, Korea, in 1999 where he is a Research Engineer of R&D Center. His research interests include microarchitecture of EISC and ASIC design.



Byung-Gueon Min received the BS degree in electronic engineering from Korea University in Seoul, Korea in 1987. From 1987 to 1997, he was with the Semiconductor Division of Samsung Electronics, where he was engaged in research and development and developed Super VGA ICs, 2D/3D multimedia graphic accelerator ICs for PC's graphic adapter, and digital still camera chips including a 32-bit embedded microprocessor. From 1997 to 1999, he was with the Silicon Modular Network Inc. in Korea, where he designed circuits for micro-controller and developed a solid state hard disk controller IC. In 1999, he joined the Advanced Digital Chips, Inc. in Korea, where he has been working on the development of new embedded microprocessors, extendable instruction set computer (EISC), and integrated circuits using the EISC for application specific products, such as, video/graphic application, network appliance, and general purpose microprocessor ICs. At present, he is the Director of the Research Laboratory at ADChips.

Hyeong-Cheol Oh received the BS degree in electronics engineering from Seoul National University in 1982 and the MS degree in electrical and electronics engineering from the Korea Advanced Institute of Science and Technology in 1984. He received the PhD degree in electrical engineering from the University of Maryland at College Park in 1993. He joined Korea University at Seo-Chang in 1994 where he is currently a Professor of electronics and information engineering. He also worked for three years at Goldstar Semiconductor Ltd. in Korea. His research interests include parallel computation, computer arithmetic and architecture, and VLSI design.