# Adaptive explicitly parallel instruction computing for embedded systems

LIAO Ji-rong[1], DONG Hai-tao[2]

(1. School of Computing, National University of Singapore, Singapore 117543; 2. National Laboratory of Computational Fluid, Beijing University of Aeronautics and Astronautics, Beijing 100083, China)

**Abstract**: Reconfigurable hardware offers the embedded systems the potential for significant performance improvements by providing support for application-specific operations. Adaptive Explicitly Parallel Instruction Computing is a prototype model such that fine-grain and dynamically reconfigurable structure is tightly coupled with a generic EPIC machine. AEPIC allows application programs to add specialized functional units yielding a dynamically varying instruction set interface to the running application without compromising current compatibility model. Two advantages of AEPIC are ① dynamic configuration support; ②application specific instruction set synthesis. In order to investigate the ideaof AEPIC's potential realistic experiments are conducted in an environment that incorporates the AEPIC simulator and actual reconfigurable hardware of Xilinx FPGA. Results show that AEPIC can achieve the similar or higher performance at a much lower execution frequency, compared with EPIC.

**Key words**: adaptive EPIC; instruction synthesis; dynamic reconfiguration

**中图分类号**: TP3　　**文献标识码**: A　　**文章编号**: 1000-274X(2003)06-663-06

In traditional view, instruction set architecture (ISA) is designed to provide a fixed set of primitives that enables low-complexity implementations of various applications. A consequence of the ISA-based processors is that an application implementation can use the instructions from the fixed instruction set only. In many cases, specialized operations tailored toward specific application domains can result in significant performance benefits. For example, digital signal processing applications contains many multiply-accumulate operations. The implementation of these application will be more efficient if multiply-accumulate instruction is available. However, extending instructions sets directly may cause bloated instruction sets and lead the processors design more complex and expensive[1] because they should be specialized enough to allow significant performance benefits, and at the same time, be general enough so that they are useful for a variety of applications. Reconfigurable hardware is one alternative to realize specialized instructions to simplify processor designs and improve application performance. Moreover, as the specialized instruction sets provided by reconfigurable hardware is flexible, reconfigurable hardware has the potential to evolve with the applications and tailor the instructions on a per application basis. Adaptive Explicitly Parallel Instruction Computing (AEPIC)[2] is proposed to augment the EPIC architecture[3] with a reconfigurable component on the same die that is amenable to compiler optimizations. The AEPIC architecture combines the advantages of the EPIC with simpler architecture and that of programmable logic that exploits fin-grain

**Reviced date**: 2003-07-15

**Authors**: Liao Ji-rong(1979-), Man, Guandong Zengcheng, Master of science candidate at the National University of Singapore, his research interests focus on computer application and computer system architecture.

parallelism through explicit control over micro-architecture features. It allows multiple instructions to be processed on each cycle and allow application programs to dynamically alter the functional unit composition of the data-path of the processor using the programmable logic resources. In this paper, we describe the AEPIC architecture and the software environment support for this architecture. Realistic experiments on actual FPGA hardware are conducted to evaluate the potential of the AEPIC idea.

The rest of this paper is organized as follows: In Section 2 we review the AEPIC architecture to see how it supports dynamic configuration. In Section 3 we discuss the software environment supporting the architecture and instruction synthesis that produces the specialized instructions. In Section 4 we present our results of realistic experiments. Finally, the paper concludes with Section 5.

## 1 AEPIC

An abstract execution model for an AEPIC processor is shown in Figure 1.

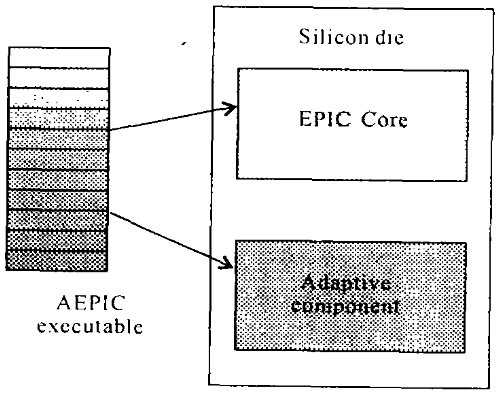

Fig. 1 AEPIC execution model

The processor can be viewed as composed of one EPIC core and one adaptive component. The EPIC core is intended for performing non data-processing tasks. The adaptive component consists of functional units that have been configured into the datapath by some reconfiguration instructions. Operations performed by the configured functional units are triggered by specific AEPIC instructions invoked on the EPIC core. One side-effect of this execution model is the extra cost for loading a configuration before the instructions are executed. In this section, we describe how AEPIC architecture supports dynamic configuration to minimize this extra cost.

### 1.1 AEPIC architecture

Figure 2 shows in detail an abstract model of an APEIC architecture. The EPIC core is a standard EPIC machine with additional functional units to execute the non-EPIC instructions. These non-EPIC instructions cause execution control switch to or state changes on the adaptive component. The instruction cache, Fetch/Decode, and the generation of control signals comprise the control unit. Control unit is not only responsible for the fetching and processing of instructions in program order, but is also responsible for flushing or stalling the pipelines of the functional units caused by program events such as branch operations or interruptions. The adaptive component of the AEPIC processor consists of the Configuration Cache Hierarchy, Multi-context Reconfigurable Logic Array (MRLA), Array Register File (ARF) and Configuration Register File (RCF) connect together via bus configurable interconnect. The MRLA is the primary resources used for hosting the Configured Functional Units (CFU) which execute specialized instructions. Like a typical FPGA, the MRLA is a two dimensional region of programmable logic and interconnect bocks. In order to mask the reconfiguration overhead, the programmable elements in MRLA are associated with multiple CFUs. This allows multiple logic designs to be simultaneously resident on the MRLA. The desired logic design can be activated by selecting the appropriate CFUs for each programmable element. Thus the MRLA allows rapid switching between two different sets of CFUs so that context switching (reconfiguration overhead) of two different logic designs is very fast. In some sense, MRLA can be effectively viewed as an array of FPGAs. The memory system of AEPIC architecture consists of the standard instruction and data cache memory hierarchies. The instruction cache and data cache are disjoint at the highest level of the memory hierarchy but share the subsequent levels of the memory hierarchy. The first level cache is followed by a large second level cache and then the main memory. In addition, AEPIC architecture supports a novel memory hierarchy for configuration data. The Configuration Cache Hierarchy consists of two hierarchies, namely C-Cache and C1 Cache. It simulates the functions of register and cache in general processor for program data except that it is explicitly controlled by the compiler. C-Cache serves as a temporary cache for configurations before they are instantiated on the MRLA, which way is analogous to that registers serve as storage for program values. The next level C1 Cache is larger than C-Cache and is connected to the external memory. The configuration is loaded to C1 Cache from external memory if a request for a particular configuration is not located in C-Cache or C1 Cache. This is analogues to the way that cache is used in instruction/data memory hierarchies of standard microprocessors.

The Configuration Register File (CRF) consists of a set of configuration registers (CR). Each CR serves as an alias to either a CFU allocated in the C-Cache. Configuration register is used as an operand in AEPIC instructions to refer to a CFU.

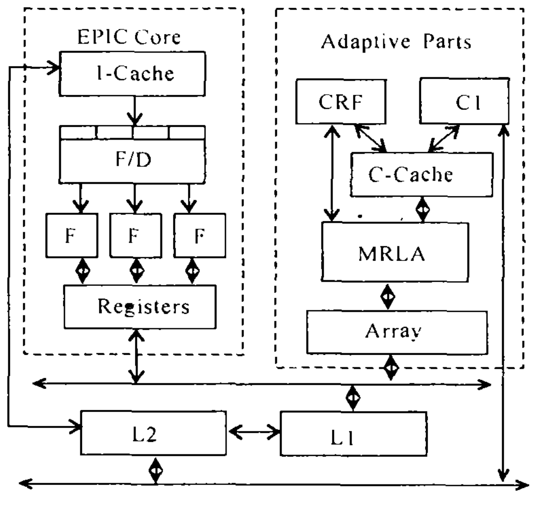

Fig. 2　AEPIC architecture

Besides the features mentioned above, the AEPIC architecture provides the following supports to enable efficient dynamic configuration:

1) Architecturally transparent reconfigurable resource assignment.

2) Implicitly specified operands for CFUs.

3) Explicitly specified operation latencies.

4) Parametric description of the architecture.

5) Inherited EPIC features.

### 1.2　Comparing AEPIC with other architecture

AEPIC is an extension of EPIC. It inherits the innovative features of EPIC architectures and reconfigurable hardware. EPIC is the evolution of very-long-instruction-word (VLIW) with addition of features such as predicated execution and support for software pipelining, etc. The advantage of AEPIC over EPIC is that it enables dynamic specialized instruction. For the execution of sequential code, there is no difference between these two architectures. For parallel code, the AEPIC can exploit far more parallelism and do not compete for function units because it has an extra reconfigurable hardware. Superscalar processor can exploit parallelism in sequential code, and they can adjust their execution on the fly for operations with variable latency. However, the hardware complexity of dynamically determining dependencies between instructions prevents superscalar processors from scaling well beyond a modest number of instruction issue slots. AEPIC can identify the parallelism during compilation. AEPIC targets fine-grain specialization. It is capable of building functional units that can implement the synthesized operation of several instructions. This differ AEPIC from GARP[4], PipeRench[5] and Rapid[6] which target coarse-grain parallelism in loop level. Another distinct aspect of AEPIC is that it support dynamic configuration.

## 2　Software environment

AEPIC compiler and simulator have been created to enable research and study for AEPIC. Both of them are built upon the Trimaran[7] compilation infrastructure which is based on the HPL-PD[8] EPIC architecture.

### 2.1　Compilation framework

The AEPIC compiler's inputs include the application source program, the architecture description of the AEPIC processor and a library containing parameterized configurations for popular computational routines. After a series of standard lexical and syntactic analysis, the source program will be partitioned two parts. One will be executed on EPIC core and the other one in adaptive component. The two parts will then be processed in parallel by high-level optimization phase and the instruction synthesis phase respectively. The high-level optimization targets on the EPIC core and is similar with a standard ILP compiler. The instruction synthesis phase combines several instructions to generate a new operation which will be realized in the adaptive component. Subsequent phases of the compilation are similar in structure to back-end phases of a typical ILP compiler adapted to the characteristics of configurations. One distinct phase is added to the AEPIC compiler is configuration allocation[2]. The configuration allocation phase is aimed at optimizing allocation of resources for configurations. The resource for reconfiguration referred to the C-Cache and the MRLA.

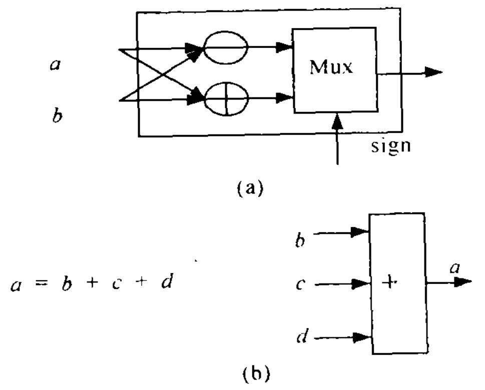

Fig. 3 Instruction synthesis

### 2. 2 Instruction synthesis

Instruction synthesis is a systematic technique for defining new instructions for a given micro-architecture. In the AEPIC, the new instructions are realized in the reconfigurable hardware. The process of instruction synthesis typically involves analyzing the program to infer the most suitable operation repertoire based on its computational characteristics for the intended micro-architecture. The Instruction Synthesis phase takes a list of candidate partitions that have been identified from previous partitioning analysis phase, combine several instructions to a new operation and map it onto the programmable logic, and then synthesizes a set of functional units that can implement all the computations of the code partitions. Figure 3 shows two examples. The combined operation can reduce the execution cycles and alleviate the memory traffic and register pressure by data reuse.

To accelerate the procedure, a library of pre-synthesized macros for various basic operators and frequently used kernels can be used as a guide for this phase. The library should be applicable to a wide range of the target programmable logic parameters, and also accommodate variations in the structure of the input partitions.

### 2. 3 Simulator

A complete hardware implementation of AEPIC does not yet exist, so AEPIC program must be executed on a simulator. The AEPIC simulator is based on the cycle level simulator of the HPL-PD EPIC architecture. It translates the intermediate representative generated by the AEPIC compiler to the AEPIC format and simulates the AEPIC semantics for a user-defined architecture. AEPIC generates run-time information such as clock cycles taken for execution, static instruction count, etc. It also provides detailed information about the execution profile on the adaptive component of AEPIC such as time spent for data-path reconfiguration, computation time on MRLA, effectiveness of configuration caches, etc.

## 3 Realistic evaluation

To evaluate the idea of AEPIC's potential, we conducted realistic experiments on actual reconfigurable hardware to compare AEPIC against EPIC. The Xilinx XCV1000 FPGA chip on the Celoxcia RC1000 board is used instead of the MRLA as no actual MRLA hardware exists The EPIC architecture is a 9-slot width architecture consisting of four integers, two floating point, two memory and one branch units. The EPIC processors assumes running at 200MHz. The FPGA programs for AEPIC are coded manually. The benchmarks we used include the Adpcmdec, IDEA and DCT. Adpcmenc is Adaptive Differential Pulse Code Modulation Encryption. It is widely used in speech compression. IDEA implements a complete 8-round International Data Encryption Algorithm. DCT is 2D discrete cosine transform. It is widely used in image compression application. Table 1 shows details of the application implementation in

AEPIC and EPIC platform respectively. A total of six implementations were made of these benchmarks. The column of Slices No. is the source requirement for the FPGA implementation. All of the FPGA implementations use less than 10% of total slices on the RC1000 board. The Max frequency of AEPIC design is the FPGA's frequency returned by the place and route tools. From the Operations and Time column, we can see that AEPIC achieve similar performance with EPIC in benchmark Adpcmdec and IDEA. As for benchmark DCT, AEPIC's speedup is more significant, reaching as twice fast as the EPIC. These results indicate that an AEPIC processor running at a lower frequency can achieve the similar or even higher performance of an EPIC processor running at a 10 times higher frequency.

Author names and affiliations are to be centered beneath the title and printed in Times Roman 12-point, non-boldface type. Affiliations are centered, italicized, not bold. Include e-mail addresses if possible. Follow the author information by two blank lines before main text. For papers with multiple authors always group authors from the same institution together, that is, list each institution only once. When authors are from more than one institution center the name and addresses across both columns.

**Tab. 1　Evaluation results**

| Benchmark | Architecture | Configuration File Size/kb | Slices No. | Max. Freq /MHz | Operations and Time |
|---|---|---|---|---|---|
| Adpcmdec | AEPIC | 749 | 915 | 19.752 | 0.033 8 s |
| | EPIC | — | — | 200 | 0.028 5 s |
| IDEA | AEPIC | 749 | 975 | 14.913 | 0.094 8 s |
| | EPIC | — | — | 200 | 0.106 9 s |
| DCT | AEPIC | 749 | 612 | 17.504 | 7 589 dct/s |
| | EPIC | — | — | 200 | 3 890 dct/s |

## 4 Conclusion

In this paper, we review a fine-grain, dynamically reconfigurable AEPIC architecture that consist of an EPIC core tightly coupled with a reconfigurable hardware that implements compiler synthesized instructions. Preliminary evaluation using actual FPGA hardware shows that AEPIC of low execution frequency can achieve the same or higher performance with those EPIC processors of 10-times higher execution frequency. As the work for AEPIC is mainly focused on the dynamic configuration, the future emphasis would be placed on compiler work to find efficient instruction synthesis algorithm.

## References:

[1] BURGER D, GOODMAN J R. Guest editors' introduction: Billion-transistor architectures[J]. IEEE Computer, 1997, 30(9): 46-49.

[2] TALLA S. Adaptive explicitly parallel instruction computing[D]. New York: New York University, 2000.

[3] SCHLANSKER M S, RAU B R. EPIC: An architecture for instruction-level parallel processors[R]. Palo Alto: Hewlett Packard Laboratories, 2000.

[4] HAUSER J R, WAWRZYNEK J. GARP: A MIPS processor with a reconfigurable coprocessor[A]. POCEK K L, ARNOLD J. IEEE Symposium on FPGAs for Custom Computing Machines [C]. Los Alamitos: IEEE Computer Society Press, 1997. 12-21.

[5] GOLDSTEIN S C, SCHMIT H, MOE M, *et al.* PipeRench: A coprocessor for streaming multimedia acceleration[A]. DEGROOT D. Proceedings of the 26th Annual ACM/IEEE International Symposium on Computer Architecture[C]. Los Alamitos: IEEE Computer Society Press, 1999. 28-39.

[6] CRONQUIST D, FRANLKIN P, BERG S, *et al.* Specifying and compiling applications for RaPid[A]. POCEK K L, ARNOLD J. IEEE Symposium on FPGAs for Custom Computing Machines[C]. Napa: IEEE Computer Society Press, 1998. 116-127.

[7] TRIMARAN R G. An infrastructure for research in instruction-level parallelism [OL]. http://www.trimaran.org, 2003-03-15/2003-05-18.

[8] KATHAIL V, SCHLANSKER M, RAU B R. HPL-PD architecture specification: Version 1.1[R]. Palo Alto: Hewlett Packard Laboratories, 2000.

（编　辑　姚　远）

# 嵌入式系统的适应性显式并行指令技术

廖继荣[1],董海涛[2]

(1.新加坡国立大学 计算机学院,新加坡 117543;2.北京航天航空大学 国家计算流体力学实验室,北京 100083)

**摘要**:利用可重构技术可以显著改善系统的性能。重点分析探讨了支持可重构技术的适应性显式并行指令技术(AEPIC)的系统模型。该系统模型由一个显式并行指令技术(EPIC)处理器和一个精细且可动态重构结构紧密连接而成,其特点在于支持动态可重构和指令合成,因此可以为不同的应用程序提供不同的动态指令集。通过AEPIC模拟器和可重构硬件Xilinx FPGA进行模拟分析以验证其有效性。实验结果表明:比起显式并行指令技术,此系统模型能够以同样的运行频率得到更高的运行速度。

**关 键 词**:适应性显式并行指令技术;指令合成;嵌入式系统

·学术动态·

## 我校取得一批有重要影响的科研成果

5年来,全校共获得各类科研奖励207项,其中省部级以上奖励107项,省部级一等奖以上的标志性科研奖励11项;发表各类研究论文8 200余篇,其中被SCI内圈(光盘版)收录论文数从1997年的37篇上升至2001年的109篇,在全国高校的排名由1997年的第36位跃升至2001年的第26位,在省内高校中排名第二。其中,特别是以侯伯宇教授为带头人的理论物理研究所和以史启祯教授为带头人的物理无机化学研究群体,被SCI收录论文数分别占到全校的28%和13%。高鸿院士由于在分析化学研究领域的重要贡献,2002年荣获"第九届何梁何利基金科学与技术进步奖";张国伟院士主持完成的国家自然科学基金重大项目"秦岭造山带岩石圈结构、演化及其成矿背景研究"获1999年国家自然科学二等奖;李继闵教授的"九章算术及其刘徽注研究"获1999年国家科技进步三等奖;舒德干教授在脊椎动物起源研究方面取得重大进展,研究论文多次在*Nature*杂志上发表,成果先后入选1999,2001年"全国十大科技进展"和1999年"中国基础研究十大新闻",其本人荣获"第二届长江学者成就奖"一等奖和"全国杰出专业技术人才"荣誉称号。在哲学社会科学研究方面,张岂之教授主编的《中国思想史》及其在弘扬中华优秀传统文化方面的系列成果、何炼成教授主编的《中国发展经济学》、彭树智教授主编的《中东国家通史》、周伟洲教授主编的《英国、俄国与中国西藏》、黄留珠教授主编的《周秦汉唐文明》、李浩教授主编的《唐代关中士族与文学》,以及考古领域里《扶风案板遗址发掘报告》等著作,在学术界引起了较大反响。

(薛 鲍)