An Elastic Architecture Adaptable to Millions of Application Scenarios

Yunji Chen, Tianshi Chen, Qi Guo, Zhiwei Xu, Lei Zhang

An Elastic Architecture Adaptable to Millions of Application Scenarios

Yunji Chen[1,2], Tianshi Chen[1,2], Qi Guo[1,2], Zhiwei Xu[1], and Lei Zhang[1]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China
[2] Loongson Technologies Corporation Limited, Beijing 100190, China

{cyj,chentianshi,guoqi,zxu,zlei}@ict.ac.cn

Abstract. With the rapid development of computer industry, the number of applications has been growing rapidly. Furthermore, even one application may correspond to different application scenarios which impose different requirements on performance or power. This trend raises the following question: how to design processors that best suit millions of application scenarios? It is impractical to design a dedicated processor for each single application scenario. A better alternative is to design a general-purpose processor architecture that can generate different architecture instances on demand. This paper proposes a novel CPU architecture called Elastic Architecture (EA), which can be dynamically configured into different architecture instances to suit different application scenarios. By employing reconfigurable architecture components (instruction set, branch predictor, data path, memory hierarchy, concurrency, status & control, and so on), the EA can achieve considerable elasticities on each application, which enables the EA to meet the performance or power requirements associated with each application scenario. We validate the effectiveness of the EA on a prototype implementation called Sim-EA. We demonstrate that Sim-EA exhibits large elasticities over 26 benchmarks of SPEC CPU2000, enabling Sim-EA to reduce the average energy-delay product (EDP) by 31.14% of a fixed baseline architecture.

1 Introduction

Since the appearance of the first electronic computer, millions of computer applications have emerged. Each single application (e.g., an internet game application running on hand-held terminals or desktop computers) may correspond to several *application scenarios,* which specify not only the concrete application but also the requirements imposed on the responses (response refers to some execution expenditure such as performance, power, performance-power tradeoff). To cope with the requirements of millions of application scenarios, more and more processor types are devised and used. For instance, the processor products of Intel increased from 5 types to near 30 types between 1999 and 2009 [10, 11]. However, the problem of

having a processor matching different application scenarios at the runtime and maintaining low costs in processor design and manufacture is not solved.

To address this problem, we propose a novel CPU architecture called the *Elastic Architecture* (EA) whose architecture components can be dynamically reconfigured to suit (adapt to) different application scenarios. The reconfigurable components in an EA may include instruction set, branch predictor, data path, memory hierarchy, concurrency, status & control, and so on. Through dynamically reconfiguring these components, the processor can be adjusted on demand. The EA can potentially be configured to a considerable number of *architecture instances*[1], which is significantly larger than the number of architecture instances that can be activated by conventional architectures. For example, a modern processor with the Dynamic Voltage/Frequency Scaling (DVFS) technique, whose frequency can be dynamically configured to 1 GHz, 1.5 GHz, or 2 GHz, only has three different architecture instances. In contrast, this paper shows that the EA can be reconfigured into more 70 million architecture instances. Consequently, the EA can offer higher *elasticity* (defined as *the ratio of the worst-case response to the best-case response*) for each application scenario, compared with conventional designs. Taking again the processor with DVFS as an example, its *performance elasticity* is at most 2 (the ratio of execution times when running at 1 GHz and 2 GHz) for all applications. Such a small elasticity of the conventional architecture makes it hard to meet sophisticated requirements on responses (e.g., the power consumptions should be less than 100 Watt while the execution should be as fast as possible), while the EA is capable of tackling such requirements for various application scenarios.

We implement a prototype design of EA, named Sim-EA. Sim-EA can be reconfigured into more than 70,000,000 architecture instances. We evaluate the elasticity of performance-power tradeoff with EDP (Energy-Delay Product) of the EA, using 26 benchmarks of SPEC CPU2000. The arithmetic average EDP elasticity of Sim-EA is 5.41, and such a large elasticity indicates that selected reconfigurable components have crucial impacts on the performance-power tradeoff. To demonstrate the effectiveness of EA, we also compare the achieved EDPs of Sim-EA with those of a traditional fixed baseline architecture. Experimental results show that the EA can provide near-optimal EDPs for all benchmark applications, which results in 31.14% average EDP reduction.

2 Elastic Architecture

2.1 Concept

The EA is a flexible CPU architecture whose architecture components can be dynamically reconfigured so as to adapt to different application scenarios. The reconfigurable components may include instruction set, branch predictor, data path,

[1] An architecture instance is a runtime configuration of an architecture, in which all architecture components are fixed, and an application can be executed with fixed performance and power (if random disturbance is ignored).

memory hierarchy, concurrency, status & control, and so on. For an application scenario with high-performance requirement, the EA can be configured into a high-performance CPU architecture instance. For an application scenario with low-power requirement, the EA can work as a low-power CPU. For an application scenario without a specific requirement, the EA can select an architecture instance with efficient performance-power tradeoff. The concrete configurable features of an EA may include:

1. The configurable instruction set mainly requires a configurable instruction decoder, which is sufficient to support instruction set variations. It enables an EA to support applications developed for different but closely-related computer families.
2. The configurable branch predictor enables the EA to adopt suitable branch prediction strategies (e.g., global history predictor, local history predictor, and so on) for applications with different types of branch behaviors.
3. The configurable data path provides flexible computational ability for the EA. Through configuring data path, the number and functionality of computational units can be adjusted to meet specific performance and power requirements.
4. The configurable memory hierarchy is crucial to the EA, since the memory hierarchy may consume half of the area in a state-of-art CPU. There are many important memory hierarchy parameters, such as cache size, cache line size, cache way, cache replacement strategies, and so on. Each of these parameters has a non-negligible impact on the performance and power.
5. The configurable concurrency includes not only TLP, but also ILP. Concretely, the concurrency configurations can include core number, issue width, instruction window size, and so on.
6. The configurable status & control include voltage adjustment, frequency adjustment, kernel-mode resource adjustment, and other miscellaneous CPU configurations.

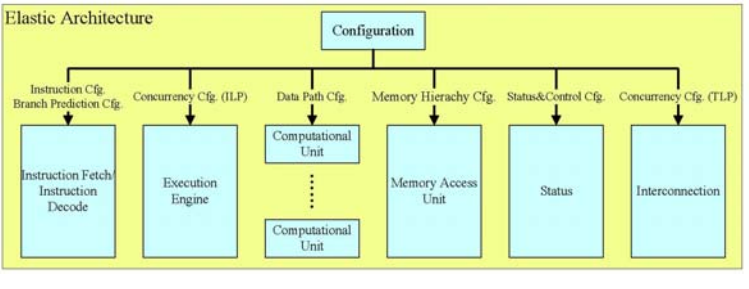

Fig. 1. In an elastic architecture, the instruction fetch&decode model, execution engine, computational units, memory access unit, status, and interconnection can be dynamically configured.

Figure 1 illustrates the concept of EA, in which a central configuration module controls the other modules (instruction fetch & decode module, execution engine, computational units, memory access unit, status and core interconnection) through several configuration buses (instruction config bus, branch prediction config bus,

concurrency config bus, data path config bus, memory hierarchy config bus, and status & control config bus).

It is worth noting that an EA does not require that all the above features of the CPU are reconfigurable. In other words, for a certain EA, it always consists of a part of fixed features and a part of reconfigurable features. Hence, determining which part to reconfigure and how to reconfigure should cautiously trade off between the design complexity and obtained benefits. An empirical guideline to determine the reconfigurable features is that such reconfigurable parts can provide large elasticities for applications, which can offer great adaptivity to a wide range of application scenarios.

2.2 Implementation of EA

To demonstrate the feasibility and merit of EA, we implement Sim-EA, a prototyping of EA on a simplescalar-like C simulator [2]. As shown in Table 1, there are ten configurable parameters in the EA, which include the issue width (WIDTH), the number of floating-point functional unit (FUNIT), the number of integer functional unit (IUNIT), the size of L1 data cache (L1DC), the size of L1 instruction cache (L1IC), the size of L2 cache (L2UC), the size of gshare branch history table (GSHARE), the size of branch target buffer (BTB), the size of reorder buffer (ROB), and the size of load store queue (LSQ). We choose the above parameters to be configurable, not only because these parameters can be conveniently configured, but also because they are critical to control the overall performance/power of the processor, i.e., most of such parameters are closely related to the ILP, e.g., WIDTH, FUNIT, IUNIT, GSHARE, BTB and ROB etc. Among these parameters, WIDTH, FUNIT, IUNIT, ROQ and LSQ can be reconfigured in about ten cycles. Once the pipeline is flushed, they can take effect immediately. L1DC, L1IC, L2UC, GSHARE and BTB need relatively long time to be reconfigured, since the corresponding RAMs need to be flushed before reconfiguring these parameters. The detailed costs of reconfiguring such parameters are also shown in Table 1.

Table 1. Reconfigurable parameters in processor with EA.

Abbr.	Parameter	Value	Reconfiguration Costs
WIDTH	Fetch Width	2,4,6,8	~10 cycle (flush pipeline)
FUNIT	FPALU/FPMULT Units	2,4,6,8	~10 cycle (flush pipeline)
IUINT	IALU/IMULT Units	2,4,6,8	~10 cycle (flush pipeline)
L1IC	L1-ICache	8-256KB: step 2*	~2000 cycle (flush L1D cache)
L1DC	L1-DCache	8-256KB: step 2*	~2000 cycle (flush L1D cache)
L2UC	L2-UCache	256-4096KB: step 2*	~10000 cycle (flush L2 cache)
ROB	ROB size	16-256: step 16+	~10 cycle (flush pipeline
LSQ	LSQ size	8-128: step 8+	~10 cycle (flush pipeline)
GSHARE	GShare size	1-32K: step 2*	~200 cycle (flush GShare table)
BTB	BTB size	512-4096: step 2*	~100 cycle (flush BTB)
Total	**10 parameters**	**70,778,880 options**	

3 Experiments

3.1 Experimental Methodology

To evaluate the EDP elasticity and the EDP reduction of Sim-EA, we employ the 26 benchmarks of SPEC CPU2000 as the representative applications for real life applications in different fields. As shown in Table 1, 10 crucial features of Sim-EA are variable, resulting in a design space consisting of more than 70 million architecture instances. As defined in Section 1, the EDP elasticity is the ratio of the worst-case EDP to the best-case EDP, which means that we should determine such two extreme architecture instances (i.e., achieving the best and worst EDPs) from the design space for each application. Furthermore, we also want to obtain the EDP reduction compared with a baseline architecture, which also requires us to obtain the optimal architectures from such a design space for each application via machine learning techniques (e.g., model tree algorithm [17]).

For fair comparison, we employ average Cycle-per-Instruction (CPI) to measure the performance of each architecture instance in the design space. In addition to the performance metric, we also estimate the *average power consumption* of each architecture instance with a widely-used metrics on performance-power tradeoff, i.e., Energy Delay Product (EDP) [7], which equals to $CPI^2 \times power$.

3.2 Elasticity

As the first step of experiments, for each of the 26 benchmark applications we employ predictive modeling techniques to explore the design space. For each application, we first randomly sample 500 architecture instances as the training set, that is, we simulate each application with 500 different architecture instances to obtain the corresponding performance/power responses. Then, such information is collected as the training data to build two predictive models i.e., model trees, for performance and power, respectively. After that, the performance/power with respect to a given architecture instance can be rapidly deduced by such models. Since we can easily know the performance/power responses with the help of predictive models, it is straightforward to find the two architecture instances with the best and the worst EDP for each application.

Once we can obtain the best and worst case EDP of Sim-EA, we can show the elasticity over 26 applications in SPEC CPU2000 in Fig. 2. We observe that the elasticity ranges from 3.31 (*sixtrack*) to 14.34 (*art*), and the arithmetic average elasticity is 5.41. Briefly, an application corresponding to larger elasticity implies that the applications are more sensitive to Sim-EA, which provides larger freedom for the EA to dynamically reconfiguring the architecture features on demand. For example, the elasticity of *art* is 14.34, which indicates that *art* is most sensitive (among all investigated 26 applications) to 10 reconfigurable parameters in Sim-EA. Moreover, the average elasticity as 5.41 indicates that the selected 10 reconfigurable parameters offer considerable freedom for Sim-EA to obtain an appropriate EDP for real-life scenarios. Similar situations can be observed when using performance or power as the

alternative response for estimating the elasticity (i.e., performance elasticity or power elasticity).

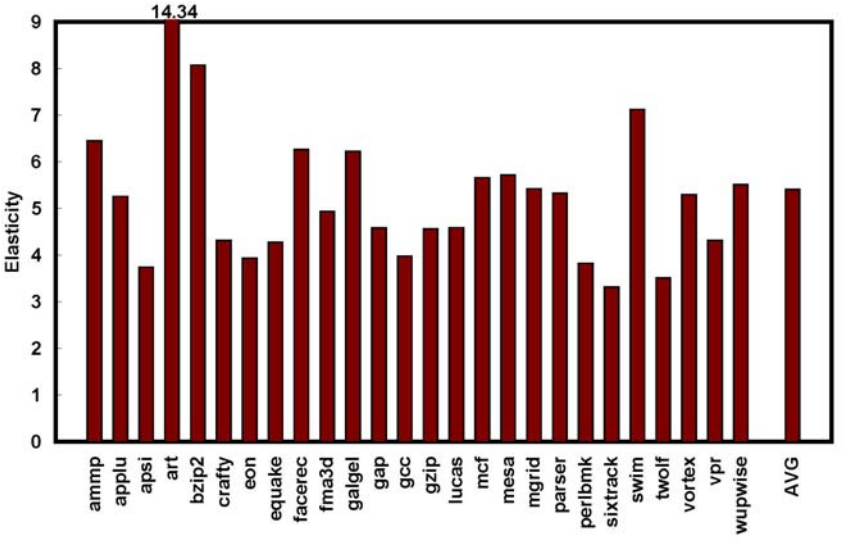

Fig. 2. EDP elasticities of the Sim-EA over different applications.

3.3 EDP Reduction

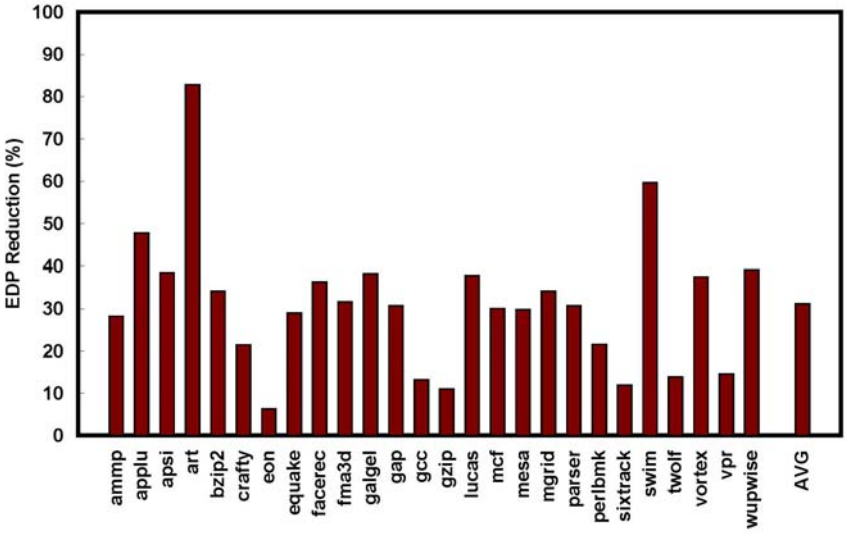

Fig. 3. EDP reduction over the baseline architecture. The arithmetic average EDP reduction is 31.14%, ranging from 6.26% (eon) to 82.84% (art) for 26 applications in SPEC CPU2000.

By integrating the optimal architecture instances (obtained for 26 benchmarks respectively) as the candidate running modes, Sim-EA can reconfigure its architecture to achieve promising EDP over different applications. The default architecture as a classical superscalar architecture in SimpleScalar Tool Suite, as shown in Table 2, is employed as the baseline. Fig. 3 shows the EDP reduction of Sim-EA to the baseline architecture, i.e., $(1 - EDP_{Sim-EA}/EDP_{baseline}) \times 100\%$. It can be observed that, for all applications, Sim-EA can reduce the EDP of baseline architecture significantly, and the arithmetic average EDP reduction is 31.14%. The benefit of Sim-EA is highlighted by application *art*, which can reduce 82.84% EDP compared with the baseline architecture. However, for *eon*, it can only achieve 6.26% EDP reduction, an insignificant improvement on EDP. To be specific, the CPI and *Power* of Sim-EA on *eon* are 0.79 and 14.43 respectively, while the CPI and *Power* of the baseline are 0.69 and 19.14, respectively. In fact, the baseline architecture has already been at the Pareto Frontier of the performance-power tradeoff function [16], and it is a near-optimal architecture instance with respect to EDP. In this case, the EDA reduction of the Sim-EA is not significant.

Table 2. Baseline architecture

Parameter	WIDTH	FUNIT	IUNIT	L1IC	L1DC	L2UC	ROB	LSQ	GSHARE	BTB
Value	4	4	4	16KB	16KB	128KB	16	8	2048	2048

4 Conclusion

With the advance of computer industry, the programmers and end users have put more and more requirements on processors. For different application scenarios, they hope that the processors can adapt to specific requirements on performance/power. Such requirements cannot be satisfied with a processor diversity approach, and may lead to excessive processor market segmentation, which increases the overall cost of processor design and manufacture. A promising solution is to make the processor elastic, which has many configurable features (e.g., instruction set, data path, memory hierarchy, concurrency, and so on). In this paper, we propose a Elastic Architecture and a prototype design of EA, which is named Sim-EA. Experimental results show that with respect to the SPEC CPU2000 benchmark suite, the elasticity of Sim-EA ranges from 3.31 to 14.34, with 5.41 in arithmetic average, provides great flexibility to fulfill the different performance/power requirements in different scenarios. Moreover, Sim-EA can significantly reduce the energy-delay product by 31.14% in average compared with a baseline fixed architecture.

In the future development of EAs, it is possible that more architecture features are designed to be reconfigurable, resulting in advanced EAs. An advanced EA is said to be *downward compatible* with a former EA if the advanced one can reconfigure all

architecture features that can be reconfigured by the former. Driven by the rapid development of the microprocessor industry, a new concept called *computer tribe* may emerge, which is the set of consecutively-developed processors adopting downward compatible EAs.

Acknowledgement

This work is partially supported by the National Natural Science Foundation of China (under Grants 61003064, 61100163, 61173006, and 60921002), the National S&T Major Project of China (under Grant 2010ZX01036-001-002), the National 863 Program of China (under Grant 2012AA012202), and the Strategic Priority Research Program of the Chinese Academy of Sciences (under Grant XDA06010401-02).

References

1. J. Abella and A. Gonzalez. On reducing register pressure and energy in multiple-banked register files. In ICCD, 2003.
2. T. Austin, et al. Simplescalar: An infrastructure for computer system modeling. Computer, 35:59–67, 2002.
3. R. Balasubramonian, et al. Memory hierarchy reconfiguration for energy and performance in general-purpose processor architectures. In MICRO-33, 2000.
4. G. Blaauw, F. Brooks Jr. "The structure of SYSTEM/360: Part I: Outline of the logical structure". IBM Systems Journal, Vol. 3, No. 2, 1964.
5. C. Dubach, et al. A predictive model for dynamic microarchitectural adaptivity control. In MICRO-43, 2010.
6. D. Folegnani and A. Gonz´alez. Energy-effective issue logic. In ISCA, 2001.
7. R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. IEEE Journal of Solid-State Circuits, 31(9):1277–1284, 1996.
8. K. Hoste and L. Eeckhout. Microarchitecture-independent workload characterization. IEEE Micro, 27(3):63–72, 2007.
9. C. J. Hughes, et al. Saving energy with architectural and frequency adaptations for multimedia applications. In MICRO-34, 2001.
10. Intel 1999 annual report. `http://download.intel.com/museum/archives/pdf/1999 AR.pdf`
11. Intel 2009 annual report. `http://www.intc.com/intelAR2009/`
12. E. Ipek, et al. Core fusion: accommodating software diversity in chip multiprocessors. In ISCA, 2007.
13. E. Ipek, et al. Efficiently exploring architectural design spaces via predictive modeling. In ASPLOS-XII, 2006.
14. V. Kontorinis, A. Shayan, and D. M. Tullsen, et al. Reducing peak power with a table-driven adaptive processor core. In MICRO-42, 2009.
15. F. Liu, et al. Understanding how off-chip memory bandwidth partitioning in chip multiprocessors affects system performance. In HPCA, 2010.
16. G. Mariani, et al. An industrial design space exploration framework for supporting run-time resource management on multi-core systems. In DATE, 2010.
17. J. R. Quinlan. Learning with continuous classes. In AI, 1992.