Chapter 10
ERA – Embedded Reconfigurable Architectures

Stephan Wong, Luigi Carro, Mateus Rutzig, Debora Motta Matos, Roberto Giorgi, Nikola Puzovic, Stefanos Kaxiras, Marcelo Cintra, Giuseppe Desoli, Paolo Gai, Sally A. Mckee, and Ayal Zaks

Abstract In a scenario where the complexity and diversity of embedded systems is rising and causing extra pressure in the demand for performance at the lowest possible power budget, designers face the challenge brought by the power and memory walls in the production of embedded platforms. The focus of the ERA project is to investigate and propose new methodologies in both tools and hardware design to break through these walls, and help design the next-generation embedded systems platforms. The proposed strategy is to utilize adaptive hardware to provide the highest possible performance with limited power budgets. The envisioned adaptive platform employs a structured design approach that allows integration of varying computing elements, networking elements, and memory elements. For computing elements, ERA utilizes a mixture of commercially available off-the-shelf processor cores, industry-owned IP cores, and application-specific/dedicated cores. These are dynamically adapted regarding their composition, organization, and even instruction-set architectures, to provide the best possible performance/power trade-offs. Similarly, the choice of the most-suited network elements and topology and the adaptation of the hierarchy and organization of the memory elements can be determined at design-time or at run-time. Furthermore, the envisioned adaptive platform must be supported by and/or made visible to the application(s), run-time system, operating system, and compiler, exploiting the synergism between software and hardware. Having the complete freedom to flexibly tune the hardware elements allows for a much higher level of efficiency, riding the trade-off curve between performance and power compared to the state of the art. An additional goal of the adaptive platform is to serve as a quick prototyping platform in embedded systems design.

L. Carro (✉)
Universidade do Rio Grande do Sul, Passo Fundo, Brazil
e-mail: carro@inf.ufrgs.br

J.M.P. Cardoso and M. Hübner (eds.), *Reconfigurable Computing: From FPGAs to Hardware/Software Codesign*, DOI 10.1007/978-1-4614-0061-5_10,

10.1 Project Partners

1. Technische Universiteit Delft, The Netherlands (Coordinator)
2. Industrial Systems Institute, Greece
3. Universita' degli Studi di Siena, Italy
4. Chalmers University, Sweden
5. University of Edinburgh, UK
6. Evidence, Italy
7. ST Microelectronics, Italy
8. IBM, Israel
9. Universidade do Rio Grande do Sul, Brazil

- Project Coordinator: Stephan Wong, Technische Universiteit Delft, The Netherlands
- Start Date: 2010-01-01
- Expected End Date: 2012-12-31
- EU Program: 7th Framework Programme, FP7-ICT-2009.3.4 Embedded System Design, STREP Project No. 249059Global Budget: 4.014.511,00 €
- Global Funding by EU: 2.800.00,00 €
- Contact Author: Luigi Carro, Email: carro@inf.ufrgs.br

10.2 Introduction

The embedded systems market has become a strong focus in Europe that distinguishes itself from the more high-performance systems market in the USA, and the consumer electronics and the semiconductor market in Asia. Strong application areas in Europe are spread among different application markets like in automotive, aerospace, industrial automation, medical/healthcare, and telecommunication. A key (hidden) element in embedded systems is the embedded processor that determines most of their functionality. Traditionally, embedded applications with very specific requirements (power, size, cost, speed, or any combination thereof) were implemented in dedicated application-specific integrated circuits, leading to long design times and re-designs for new applications. In recent years, the key driver in the design of the embedded processor has been integration in order to keep (design) costs low, reduce time-to-market, and improve functionality of embedded systems. A striking example is the mobile phone, in which the number of separate chips was significantly reduced because of the utilization of a full embedded platform. The integration is still continuing on a single chip towards a more structured (on-chip) design by combining multiple IPs, and utilizing more complex and diverse programmable processor cores. Additionally, mobility is translated into limited power budgets that must satisfy all the required functionality. Several examples show that specialized hardware performs better and consumes less power than general-purpose processors. The trend towards specialization is also apparent in large –albeit specific– markets, such as the

DSP and Graphics Processing Unit (GPU) markets, where specialized architectures are fast gaining ground because their organization is very efficient for a certain class of problems. This has been the key issue of embedded systems platforms until very recently. Once again, embedded systems are nowadays moving from very specialized devices, targeting a single function, to more general-purpose platforms that must deliver high performance and low energy for a broad range of applications, which should cover all the important application domains where embedded platforms are used. Hence, the embedded processor and its platform are key to the EU development in these diverse and competitive fields.

To sustain the high-performance and low-energy behavior expected by consumers, the important dilemma which arises is whether to have dedicated specialized sub-systems in a fixed multi-core organization, or strive to do better. Creating a different accelerator for each new application that reaches the market is costly in terms of design time, and introduces a delay before its adoption by the whole community (compromising time-to-market). In contrast, adapting the processor architecture on-the-fly while an application is running would yield significant performance and power benefits, while avoiding the cost and time-to-market pitfalls. This is an old dream that as of yet has not come to fruition, but now three conditions exist that can bridge the gap between intention and reality. The first is the maturity of reconfigurable computing, since there are several different options to chose from (coarse- and fine-grain, homogeneous or heterogeneous, etc.); the second is dynamic adaptation, since this allows for fine-tuning at run-time, without user intervention; the third one is advanced compiler and operating systems technology, which have greatly evolved in the past years to cope with heterogeneous platforms and the need for on-line modifications and resource management.

In the embedded systems market, continued innovation is driving the need to improve existing functionalities and to introduce new functionalities. This is needed to differentiate products and to entice consumers/customers to purchase new or improved products. Whilst in the past product cycles span several years, the competition nowadays is cut-throat and product design cycles on average are more in the range of 1 year. This led to a more structured, platform-based design approach of embedded systems, and in turn of the embedded processors. Use of standardized IP blocks and re-use of existing designs has become common practice. Unfortunately, although a platform greatly reduces design effort and time, it precludes ultimate optimizations which are not foreseen at design time for niche markets. This way, a platform can very easily become obsolete, since new standards covering communication, audio and video processing are constantly emerging. Hence, a platform inherits all the benefits and all the inefficiency that a general-purpose processor could have when targeting specific functionalities. This scenario of changing applications in short times calls for some sort of adaptation during the lifetime of a product. Although FPGAS and eFPGAs can provide some adaptation for what concerns changing applications, their programming times preclude their adaptation during a program phase, without touching important points like memory and communication. As embedded systems must rely on limited energy supplies, most likely these optimizations must also be performed

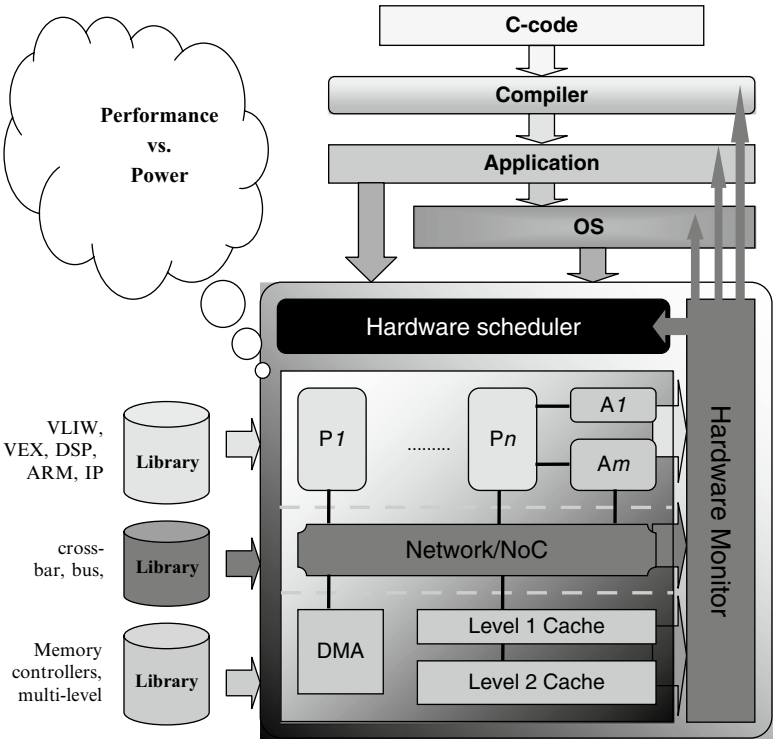

Fig. 10.1 General overview of the ERA project

at the hardware level, in order to extract the most out of the underlying technology. Due to the complexity of such a task, innovative solutions must be sought to ensure adaptability targeting optimization for different markets and for different applications within the same market. The ERA project covers exactly this problem by introducing a reconfigurable fabric able to perform adaptation at different moments (design time, application deployment, even during the lifetime of a product) to adapt to new standards without losing its power efficiency advantages, as shown in Fig. 10.1. Finally, reconfiguration will be critical for products developed for high-reliability markets (automotive, health) or for products developed in state-of-the-art processes, that are likely to have many defects, since we are reaching the limits of nature itself. In this case, without having total control of the fabrication process, one must devise clever fault-tolerant mechanisms at design time that can cope with potential low yield and aging, in order to add or maintain value to the product.

Adaptability is the key to develop embedded platforms for the new, heterogeneous and multi-applications embedded market. One of the ways to adapt is to have a reconfigurable fabric available, so that hardware changes can be done whenever necessary. Although designs made with reconfigurable hardware are gaining popularity in the embedded systems market, there are several issues that must be

addressed in order to allow for the breakthrough. Some of these issues stem from the excessive power dissipation of reconfigurable devices, since their programmability takes lots of power to read form an external memory, and also their operation draws extra power due to the extra reconfigurable wiring that must be present to allow for reconfiguration itself. Moreover, one still needs a design flow that, from the final user (the embedded system designer) perspective, is simple and hides all optimization bottlenecks from view.

To cope with the reconfiguration power problem, in ERA the focus is on the development of accelerators using a coarse-grain reconfigurable fabric, composed of a reconfigurable VLIW processor, a flexible memory organization and an interconnection network that can provide better usage of power resources by distributing its routing resources online. A software stack consisting of a compiler and OS will provide the means to drive both static and dynamic reconfiguration decisions according to the application characteristics and the user objectives (in terms of power and performance).

The reason that a VLIW processor was chosen for this project is because of its excellent power/performance trade-off. However, as embedded systems target new applications, the optimal size of the VLIW is clearly an issue. Developing traditional static tools such as those available today, based on a set of benchmarks, would only produce a general-purpose machine, without the power and performance benefits one would expected to have in dedicated markets with stringent requirements. Therefore, in the ERA project we target the usage of multiple and variable issue VLIW, as well as a multidimensional VLIW, so that for each different application, and for different parts of each application, an optimized processor can be constructed on-the-fly to obtain the fastest computation (for the application) with the smallest possible power budget within the constraints of the design.

As a variable issue processor is used, extra stress is placed in the memory subsystem. This requires the adaptation of the memory as well, in order to cope with variable instructions and variable amount of parallelism available at any given time during program execution. The memory hierarchy plays a significant role in the end performance of a system and its power consumption. Our goal in the ERA project is to tailor the memory system for executing a target application to meet specific (user) objectives for performance and power. The innovation in our case is that this is going to happen in concert with the processor reconfiguration (as well as NoC reconfiguration) so as to be synergistic towards meeting the user objectives. Moreover, all modifications are to be developed during online operation. Memory reconfiguration entails a vast array of techniques: resizing on-chip memory to fit program needs, optimizing cache architectures for dynamic and static power consumption, optimizing shared-memory communication via custom cache coherence protocols, modifying the replacement policies for minimizing miss rates in caches, compressing data and removing data redundancies, managing caches shared by multiple applications, partitioning the on-chip memory for different functionalities (caches, scratchpads, loop buffers, etc.) and any combination of the above. While the reconfiguration space is vast in this project, we will combine the mechanisms that work synergistically and provide global, application-driven policies for applying

these mechanisms. Our goal is to provide a malleable memory system that can be efficiently adapted to the needs of an application using both static (application information) and run-time information (hardware monitoring & feedback).

In the same manner, as the processor-cache and processor-memory bandwidth and throughput varies, so the communication needs also vary. Devising a single strategy for the communication network would mean either extra power without equivalent performance gains, or degradation in the quality of services. Therefore, in this project a reconfigurable NoC is also proposed as a manager of the communication needs.

At the software level, in the ERA project the consortium proposes a tool chain that comprises not only the application APIs needed to guide reconfiguration, but also run-time managing of the reconfiguration process via the OS. Hence, from the final user perspective, the embedded system designer himself, the adaptivity of the platform will be seen as a seamless software development process. One can envisage that the user could control the reconfiguration, since having a multitude of possibilities for tuning the system performance leads to a set of possible implementations of the same hardware accelerator, each with different performance characteristics in terms of execution time and hardware resources used. In this project, we envision the possibility of performing a coordinated system-wide dynamic reconfiguration that will be able to provide the needed hardware resources to the application in a "just-in-time" fashion, hence, providing adaptation for a wide range of application requirements. The online adaptation is seen as a user-controllable trade-off between the execution time of the software and the hardware resources – and consequently power consumption– required by the various subsystems.

Another important issue raised by the adaptation process concerns the production of optimized code. In current fixed VLIW architectures, the amount of ILP that can be exploited is fixed at design time of the architecture, and at compilation time of the application However, since the ERA project is based on a variable-issue VLIW, the compiler must be aware that run-time reconfiguration of the VLIW schedule can also be performed. The biggest challenge is performing this schedule change with minimal cost, which will involve a combination of multiple code versions and fast schedule, and code generation algorithms. In this project, we propose a compilation tool-chain that can act as a mediator between the features and requirements exposed by the applications (e.g., memory locality, memory bandwidth requirements, and processing needs, as well as execution phases, multi-threaded behavior, communication, and synchronization) and the reconfiguration potential of the architecture. In particular, different modules of the architecture will have the ability to reconfigure themselves according to the (in-hardware) perceived runtime conditions and resource usage. Our approach complements this hardware monitoring and adaptation by leveraging software development and tuning tools that are able to implement a range of optimization strategies of different complexities, aiming to induce the most suitable architectural reconfigurations that match the application requirements during run-time. Using this approach, the optimization tools can take advantage of the global view of the architecture together with the application structure and needs, in order to trigger profitable reconfiguration actions.

Most complex embedded systems use an operating system, which is responsible to load an application to the execution memory, decide how much CPU time each application will have and manage interrupt requests. In this project, we propose to modify the OS so that, knowing the task distribution in advance, manages the reconfiguration, decides when to reconfigure, and schedules the work in concert with reconfiguration decisions. This would not be possible without the usage of hardware monitors that can communicate information to the operating system level so that global reconfiguration decisions can be made.

Preliminary work shows that measuring only four statistics per core (instructions retired, main memory accesses, floating point utilization, and instruction stalls [1]) via appropriate performance monitoring counters (PMCs) and knowing how these statistics correlate with power consumption is sufficient to predict per-core power (e.g., when only whole-chip power can be measured). Such information has been leveraged for local task scheduling decisions to maintain user-specified power budgets. In this project, the adaptation based on analytic models using PMC data from real applications will be used in many forms: (i) it will be used in a profiling step to feed per-phase information back to an optimization system (which may simply choose alternate, lower power-consuming instruction sequences during critical phases, or may perform arbitrarily sophisticated power/performance optimizations, possibly utilizing hardware assists); (ii) model data will be used by hardware adaptation policies to reconfigure micro-and macro-architectural structures; (iii) the PMC data and analytic models will be adapted to perform system-wide adaptation in hardware and software.

Finally, in the context of a multicore implementation of the ERA reconfigurable platform, the OS will be called to schedule concurrent applications across multiple cores. Hardware will provide key monitoring information on application behavior (functional unit utilization, memory intensity, temperature, and the like). In this case, reconfiguration and scheduling decisions must be taken in concert since one affects the other. Simple scheduling decisions (e.g., not to run a particular application on a particularly configured core) can be determined in hardware, for instance if the core has more resources than the application can fully use, or far too few resources for the application to perform well, but deciding which applications to place where or how to configure the overall available resources to maximize the benefits for the workload as a whole will likely require a more global view of the system. The OS can keep monitored application information as part of each process state. It can then adapt an application's current core to best suit its performance and power requirements. On a system level, the OS can employ simple economic models to co-optimize reconfiguration and scheduling for power and performance across all running applications.

All these modifications are in line with the overall strategy of the industrial partners of the consortium. The usage of different accelerators is a common place in STMICRO products, but the problem of adapting the platform for different markets with different accelerator in the shortest possible time is unfortunately also present. Hence, the ERA project is targeting a real life problem, whose solution might have a major impact in the way embedded systems are designed in the EU.

10.3 Main Applications

Objective ICT-2009.3.4 "Embedded Systems Design" puts a strong focus on the development of a novel (generic) embedded systems design method that can be applied to several application areas. In the ERA project we develop a platform that can adapt itself through coarse-grain reconfigurable hardware to tailor the hardware itself for changing environments and needs of the applications running on the platform for different application markets and platform usage.

The proposed ERA platform can provide adaptability at different abstraction levels: optimization of application software at design time, OS control and optimization at run time to cope with changing conditions, and hardware adaptation at run time to efficiently tune its performance to the application or OS needs and taking into account power budgets. However, the hardware resources, thanks to their regularity, can also be used as adaptive spare parts, and hence increase the fault tolerant capabilities of the platform. This additional capacity can then be used either in highly reliability demanding markets, like the automotive one, or in consumer markets when the hardware uses the ultimate technology, which is likely to have many errors and a low yield (that should be increased). Finally, to exploit the adaptability of the proposed ERA platform, software tools will be made available to achieve this.

As for the application set, the ERA platform targets those that present the characteristics described as follows.

Heterogeneity: as an adaptive platform is the target, the goal of the ERA project itself concerns covering different heterogeneous applications. Heterogeneity in the present context refers to applications that require different hardware resources during different times. For example, a modern cell phone must deal with audio, video, and baseband processing, and for each of these tasks different accelerators are called in. The demonstrators chosen to validate the platform prototype are a good example of such heterogeneity coverage since we plan to optimize not only different applications that compose a modern cell phone, but also the operating system that is running on top of the platform. Additionally, the platform will allow for a mixture of heterogeneous components.

Predictability of non-functional properties such as performance and power consumption: one of the main goals of the ERA platform is to determine the precise performance needs of the applications that we investigate after which we will determine the "best" possible set of adaptive hardware component to achieve those goals. Monitoring of the performance and power are key elements within the project to allow adaptability at the three mentioned levels (application, OS, and hardware) to reach the required performance levels with the provided power budget.

Adaptability and self-awareness for coping with uncertainty, upgrades of components, and self configuration concepts: besides being able to be controlled by the OS (and hence based on a previously defined and coded in the OS optimization strategy), in the ERA project the platform can adapt to different code execution

scenarios also in an automatic fashion. This hardware based adaptation will be performed at different blocks, covering communication and processing. The overall strategy for dynamic optimization of the communication blocks is to use performance monitors that can advise the internal control system to increase resources of a certain channel of the used NOC in order to avoid congestion, while reducing resources that are not being effectively used in other channels. For the processing blocks, the idea is to allow monitoring of the executed code, and by using dynamic strategies like hardware DLLs or binary translation change the amount of resources that are being used at a certain moment of time. This adaptation may demand extra resources from the memory subsystem that must also respond with adequate bandwidth for the new performance or power node.

10.4 Experimental Results

In this section, we will show some results that sustain the basic claim of this project, that is, the clear need for an adaptive and dynamically reconfigurable platform. Most of the initial examples have been taken from the MiBench set.

10.4.1 Processor (ILP and TLP)

Currently, there is a big discussion about the amount of ILP and TLP that should be supported by the underlying platform. In the ERA project both are supported. Variable ILP can be supported by changing the VLIW issue width, while TLP is supported by activating several heterogeneous VLIWs.

Table 10.1 shows, for different applications, how TLP and ILP change for different applications, and hence the need for a reconfigurable fabric. The benchmark set covers general-purpose, embedded and parallel applications. The second column in Table 10.1 shows the mean basic-block size, as an indicator of potential ILP. It can be observed that each application has a different number of potentially parallel instructions, and hence changing the available ILP could lead to either performance or power gains.

The last column in Table 10.1 shows the total amount of executed instructions in an in-order processor. One can see that the distribution of code among threads has a huge variability (even in the same domain), and hence to cover several applications one must also provide a different number of active processors. Moreover, when sequential code is being executed, turning off processors and their clock network might lead to power efficiency, since no acceleration is possible with simple and low-power techniques.

This scenario claims for an adaptive platform as in the ERA proposal to find the optimum spot of the power-performance curve.

Table 10.1 ILP and TLP opportunities

Benchmark	Mean BB size (#instr)	Mean thread size: 4 threads #instr	4 threads %	8 threads #instr	8 threads %	16 threads #instr	16 threads %	64 threads #instr	64 threads %	Sequential (#instr)
General purpose										
equake	4.80	1,066,842	0.11	648,980	0.06	396,256	0.04	107,967	0.01	1,007,548,408
art	6.86	642,769,700	11.26	640,642,746	11.22	640,646,160	11.22	642,795,870	11.26	5,710,455,583
apsi	14.56	77,393	0.00	69,530	0.00	65,645	0.00	59,677	0.00	6,170,539,927
Embedded										
susan_e	16.60	2,397,335	1.11	1,917,022	0.89	1,680,975	0.78	1,519,993	0.71	215,489,859
patricia	5.04	5,139,690	5.00	3,485,357	3.39	4,020,714	3.91	4,194,469	4.08	102,793,656
susan_c	17.36	1,477,499	1.68	957,759	1.09	700,196	0.80	502,259	0.57	88,061,549
susan_s	12.10	5,835,705	0.42	2,931,041	0.21	1,528,875	0.11	468,044	0.03	1,374,058,841
Parallel										
swaptions	5.92	5,542,634	1.09	2,773,691	0.54	1,389,156	0.27	350,775	0.07	509,508,947
blackscholes	4.83	14,659,706	3.13	7,339,119	1.57	3,686,241	0.79	928,638	0.20	468,936,073
md	6.51	1,994,602	2.50	1,004,970	1.26	509,355	0.64	132,462	0.17	79,625,991
jacobi	6.94	6,346,588	8.44	3,171,016	4.22	1,586,607	2.11	401,982	0.53	75,191,816
fft	8.10	2,899,996	5.04	2,006,890	3.49	1,559,014	2.71	1,199,242	2.08	57,580,983
IU	8.32	1,552,984	1.54	1,065,276	1.05	1,024,484	1.01	1,026,825	1.01	101,169,484

10.4.2 Memories

In order to optimize memory power, one must have an idea about the access pattern present in different applications. This can be achieved by observing the locality of the references of the target applications. In the initial screening of the MiBench applications we realized that the locality of references allows one to reduce the leakage power consistently in comparison to previous solutions ("improved drowsy cache" [2]).

Therefore, a detailed investigation of the behavior of the *djpeg* application was done to verify whether it contained the potential for reconfiguration based on the distribution of strides when accessing data memory. Once the access pattern (the frequency of strides) is known, it can be used to reconfigure the cache by switching off parts of it [2], or by changing its size.

In order to extract the phases that may exist in a program, the execution is divided into intervals – sections of continuous execution of a program. In our example, we used intervals with length varying from one million instructions to one billion instructions. Intervals are then grouped into phases using SimPoint tool [3] in a way that each phase contains intervals with similar behavior (similar memory access patterns in this case). SimPoint uses k-means clustering in order to split program behavior into phases. K-means clustering takes a set of points in n-dimensional space and splits them into clusters by using distance between points as a metric for similarity (points with small distance are considered similar).

To characterize each interval as a point, we use frequency vectors: The *djpeg* binary that is executing in the simulator is instrumented at translation time so that each time an access to memory occurs the difference in address between the current access and the previous access is calculated (in words), and the corresponding element of the frequency vector is incremented. For example, if current access is a load from the address $0 \times 1{,}000{,}000$ and the previous was the store to $0 \times 10{,}000{,}004$ the difference is one word and the element at position one is incremented. The maximum stride that is recorded is 1,024: the element at the position 1,023 will contain the number of memory accesses that had the stride 1,024 or bigger. At the end of each interval, the vector is saved, and all its elements are reset to zero. The set of these vectors is then passed to SimPoint. To perform the clustering, SimPoint considers each vector as a point 1,024-dimensional space and performs k-means clustering of the set of intervals. The analysis performed with the interval length of one million instructions discovered *nine different clusters* (Fig. 10.2). As the length of intervals increases, less phases are discovered - for the interval length of one billion instructions SimPoint does not discover any difference among intervals of execution.

The data shown in Fig. 10.2 clearly sustain the fact that, as there are different memory access patterns as the program is being executed, there is a requirement for variable memory bandwidth and locality policy, and thus changing these parameters one might achieve huge gains on power efficiency in the ERA platform.

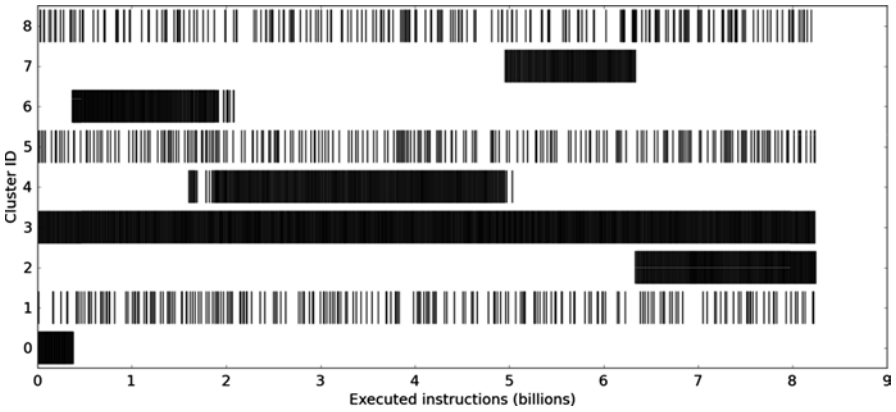

Fig. 10.2 Phases detected during execution of djpeg using the interval of one million instructions. X-axis shows the number of executed instructions while the y-axes shows the cluster that the interval belongs to (nine clusters are discovered in total)

We plan to further expand this analysis by collecting the statistics from>multiple sources and combining them to get a comprehensive view of the application behavior [4]. Data will be collected from measurements taken from emulated system (QEMU), from existing hardware and from ILP statistics generated by the compiler.

10.4.3 Network

In order to show detailed behavior of the communication patterns inside an MPSOC, we simulated four examples of real applications that can be found in Fig. 10.3. Each node represents a task, and each arc is weighted by the communication rate among tasks in Mbytes/s. In all simulations we used the SOCIN NOC [5], and used a fixed size buffer capable of storing four flits per output channel. The applications used were the MPEG4, VOPD [6], MWD (Multi-Window Display) [7] and Xbox [8], all with 12 cores, but with different communication patterns, as represented in the bandwidth of each link depicted in Fig. 10.3. In this figure, arcs in MPEG4, VOPD and MWD show rates in MB/s, while arcs in the XBOX application show rates in GB/S. A cycle-accurate traffic simulator in Java was utilized to evaluate the network hotspots and the average latency using the reconfigurable and original routers. The distribution of the cores in the NoC was specified in accordance with the communication needs of the cores, reflecting a design time choice, being based on the original application.

Figure 10.4 shows the mean efficiency of MPEG4, VOPD, MWD and Xbox when mapped to a 4×3 NoC with homogeneous buffer sizes. In this report, efficiency represents how many buffer units are being appropriately used, in accordance with

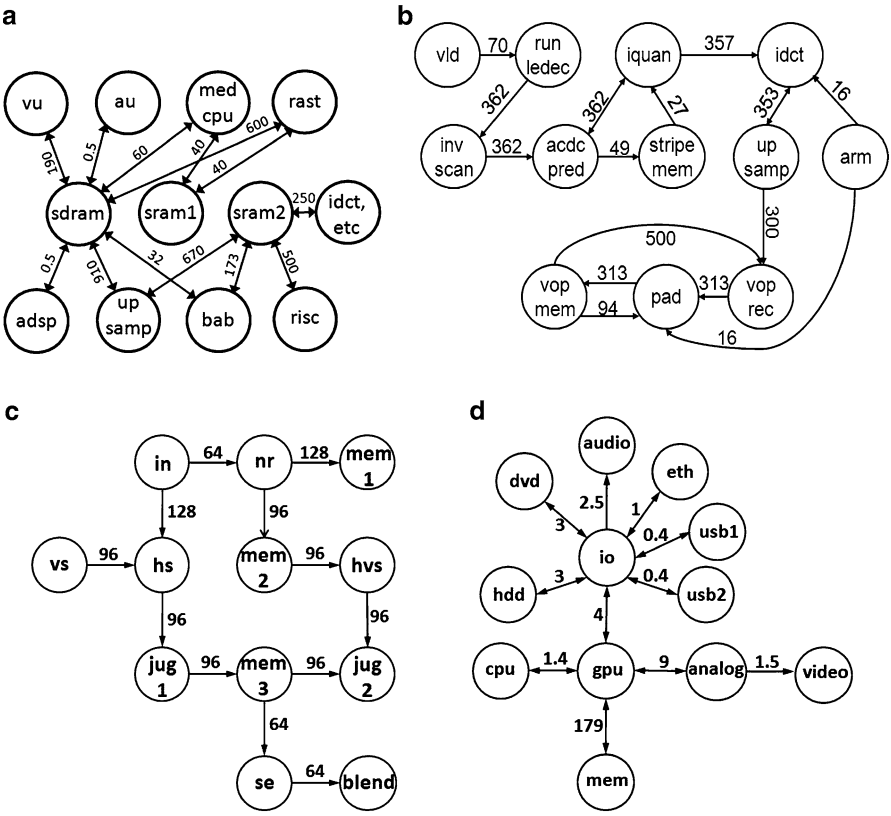

Fig. 10.3 MPEG4 (**a**), VOPD (**b**), MWD (**c**) and Xbox (**d**) task graphs

the necessity of the application. The efficiency results in Fig. 10.4 were obtained in accordance with Eq. 10.1,

$$\eta = \frac{\sum_{i=1}^{i=\#routers} \frac{\#buffers\_used\_router_i}{\#total\_buffers\_router_i}}{\#routers} \tag{10.1}$$

where the reference value has been obtained by using the best buffer distribution for a certain performance level.

Figure 10.4a presents the efficiency considering all channels of the network with the same buffer depth, and Fig. 10.4b shows the efficiency using an heterogeneous NoC, where each router might have a different number of buffers compared to others router, but all channels inside a router having the same number of buffers (achieving optimal communication throughput for a given power budget). In both cases, one can observe that the routers use excessive buffers in some channels, since not all channels present the same communication rate. In such cases, the extra buffer

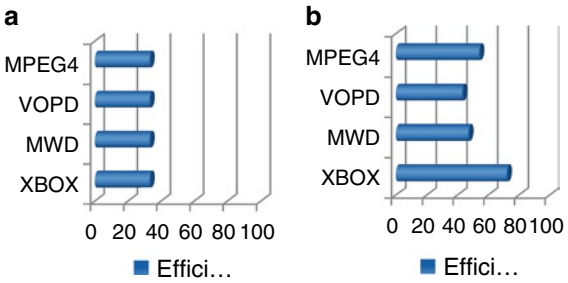

Fig. 10.4 Efficiency results (**a**) using the same buffer depth for all channels of the NoC and (**b**) using the same buffer depth for all channels of the same router, but with a different buffer depth specified to each router

units of the channel will consume unnecessary area and power. The network used in these experiments is a mesh-2D with XY-routing algorithm, handshake flow control and wormhole-switching mechanism [5]. Each input channel has a FIFO for storage of the flits. The FIFO size is defined at design time, and all channels have the same FIFO size.

One can see in Fig. 10.4a that, using a homogeneous router with the buffers sized to the best performance case, around 33% of the buffers slots are utilized. Similarly, in Fig. 10.4b only 54% of buffer slots have been used in the mean. However, they are nevertheless consuming power, but are not contributing to the reduction of the latency or the number of hotspots in the network.

Again, the concept of adaptability must be present for one to use the best communication fabric with the lowest possible power budget.

10.5 Summary of Proposed Innovations

As a means to show the effectiveness of the adaptive platform, in the ERA project commercially available FPGAs are used to demonstrate the concept. It is not our goal to devise new low-power FPGA structures, but instead propose a platform that can benefit from different types of reconfigurable hardware. The reasons for this prototyping option are as follows:

Cost-effectiveness: The NRE (non-recurring engineering) costs of FPGA designs is an order of magnitude lower than the NRE costs of ASIC designs and this is only expected to increase with newer technologies. Additionally, the costs of design tools targeting FPGAs are much lower than those targeting ASICs. This simply means that ASIC designs are increasingly only viable for mass-market embedded products and FPGA-based products will become more commonplace in the near future from a pure design cost perspective.

Time to market: The utilization of reconfigurable hardware is already commonplace as in the ASIC design trajectory; reconfigurable hardware is being utilized to perform

design verification. To shorten the time to markets windows, it is increasingly becoming more economical to ship FPGAs in actual products and forego the ensuing ASIC design trajectory. This is a natural progression from hand-designing ASIC chips towards cell-based ASIC designs with the cells now being placed on reconfigurable hardware structures. This also fits perfectly with the programmability trend as the larger more complex circuit designs and processor cores can nowadays be accommodated on the latest FPGA chips.

Performance: The performance of FPGAs is increasing at a much faster rate than their ASIC counterparts due to their very regular structure. Therefore, even though FPGAs are much slower than ASICs when implemented in the same technology, it is common that in actual product development, the utilized FPGA chips are at least one generation ahead of ASICs. This considerably alleviates the speed disadvantage. Especially when one considers that FPGAs nowadays also contain many dedicated circuitry to further improve the speed of commonly used functional elements, e.g., multipliers and on-chip memory. Moreover, the increased logic density also translates into increased memory bandwidth. This could allow the industrial partners to use the prototype of the proposed platform in a commercial way much before any dedicated ASIC is ready.

Power consumption: The power consumption of FPGAs is still considerably higher than their ASIC counterparts, but methods to save power targeting FPGAs start to gain focus, and our project intends to deal with power issues from different perspectives. Finally, low-power FPGAs are starting to make their appearance.

Flexibility: Flexibility by utilizing reconfigurable hardware can be achieved in many ways and many research projects have demonstrated this. The functionality of the hardware can be changed through reconfiguration or reconfigurable hardware structures or programming the softcore(s) (or in some cases: fixed processing cores), This allows for "in-field" updates of functionality and maybe more importantly enabling the possibility for designers to address design and process variability later on in the product cycle. This further reduces the design and verification costs.

From the perspective of an integrated design manufacturer such as STMICRO, one key factor for the success of future consumer products design, is to provide a solution to the faltering ability of existing silicon design and verification methodologies to keep up with the exponential increase of chip capacity and the increased pressure on architects and designers alike to develop very complex hardware in a decreasing amount of time. As a result the number of products that will rely on a set of predefined architecture template will greatly increase. The availability of a pre-characterized parametric architecture and programmable engine platform enables faster design space exploration and minimize critical late design respins during the physical implementation and validation phases. In order to make this possible, industrial design teams would need to have access to technologies such as the one focus of this proposal; so that programmable accelerator architectures best suited to an application domain could be easily prototyped and selected from a range of well defined features whose fast-prototyping platform is already available. This would

make a significant difference in terms of cost reduction for the total NRE and allow gaining a competitive advantage over designs that adopt a more traditional conservative approach to system level design.

The architecture proposed in the ERA project will allow Evidence to develop innovative solutions in the market of reconfigurable systems, allowing Evidence to acquire advanced knowledge in the usage of FPGA for signal processing and multimedia systems together with Embedded Linux microcontrollers. The framework developed as part of the ERA Project will be exploited by Evidence with the prototype study of an embedded board based on a Xilinx FPGA, as well as with applications in the signal processing area with key Evidence customers.

The technology developed for software support of reconfigurable embedded architectures in the ERA project will allow IBM to offer advanced tool-chain solutions that target changes in the underlying platform efficiently, thereby increasing the overall utilization and performance-per-power characteristics of the system. In particular, IBM will benefit from advanced and innovative compiler optimizations which leverage the capabilities of target platforms.

Additionally, we will combine all the solutions proposed in this project into a demonstrator platform that we expect will allow the industrial partners fast access to new products developed on top of it. The intended platform will serve several purposes:

Quick development platform for the industry: the clear interfaces defined in this project should allow the industrial partners to take from the platform everything they need and still incorporate their own IPs. Moreover, for low volumes even the prototype can be used as a commercially viable product, since the consortium will use available FPGA technology to validate its contribution.

Academic purposes: the ERA platform can be easily used to build different instances of embedded processing solutions and we foresee and will actively pursue the possibility of incorporating the ERA platform as a teaching tool in embedded courses or labs.

Having stated the main objective of the ERA project and the sub-objectives in order to reach it, we can now summarize the main deliverables of the ERA project as follows:

1. A hardware prototype reconfigurable multi-core system: a hardware platform prototype, encompassing a reconfigurable VLIW processor, reconfigurable memory, and reconfigurable NoC, that can be used as a demonstrator of the adaptive properties.
2. A software platform prototype, that working together with the hardware can provide adaptability in a seamless fashion for the platform user and software designer:

    - A supervisor coordinating the reconfiguration operations, providing the needed trade-off between application quality-of-service and resource usage;
    - Exploration of profile-directed compiler optimizations to increase performance and reduce power by scheduling reconfiguration instructions, and scheduling several cores;

- Portable compilation scheme capable of adapting to changes in the underlying architecture efficiently.

3. A set of applications identified that will benefit the proposed ERA platform. As the main test case of the platform, several programs that are used in portable phones will be used as our driving benchmark.

Here we list some of the main areas where we expect that the ERA project is going to advance the state of the art.

Reconfigurable processors. Most embedded devices will employ complex system-on-a-chip (SoC) platforms, based on several different sub-systems, where each one is built to optimize a specific application domain (DSP, 3D-graphics, etc.). However, there is no easy way to anticipate all the software applications for an embedded system, which means that a means for dynamic adaptation must be available.

Fine-grain reconfigurable computing where one can create many different circuits on an FPGA-like substrate available on chip, seemed a very promising approach to solve the need of dynamic adaptation. Indeed, many fine-grain reconfigurable approaches have proven to be a good option for reducing the design time of specialized functions. Many successful examples have shown good performance improvements and energy savings [9–15]. However, the fine-gain reconfigurable approach has been plagued by two major problems: virtualization of the reconfigurable hardware (especially managing its large state) for use in a multi-programmed environment (multiple concurrently-running applications) and a blurring or breaking of the hardware/software interface at the Instruction Set Architecture (ISA) level that complicates programming, compilation, toolsets and run-time systems. It is exactly in these areas where most of the effort is spent for the development of fine-grain reconfigurable systems. With the advent of multi-core processors and MPSoCs the situation became increasingly complex since the need to manage several different accelerators and at the same time minimize energy and ensure performance at the global system level became a necessity. Piperench [9], Molen [10], GARP [11] and Raw [12] are well known examples that extend the underlying instruction set. Such reconfigurable machines rely on the compiler to detect application "hotspots" and optimizing them by creating new specialized instructions. Despite the implications of its use, static code detection of potential reconfigurable accelerators is still being explored. However, when a compiler is responsible for finding possible candidates for optimization, two main problems emerge, particularly concerning the OS: besides the necessity of having the OS source code, the room for optimization is restricted to static code – dynamic library optimization is not possible, since when and how much the libraries will be used is not known before run-time.

More recently, reconfigurable approaches have been presented, where the focus was on dynamic hotspot detection. The pioneer approach is Warp [13]. This technique uses a non-trivial CAD algorithm to transform the sequences of instructions to a control flow graph. Then, it synthesizes the code and maps the circuit onto a simplified FPGA structure. As another example, CCA [14] (Configurable Computing

Accelerator) provides both dynamic and static detection. CCA is a tightly coupled coarse grain reconfigurable array, composed by simple functional units, coupled to an ARM processor. These examples of reconfigurable hardware are restricted to optimize standalone user applications, with specific hotspot optimizations and with hotspots known at design time –something that is becoming less and less frequent. In [15] one can find a coarse grain system that covers acceleration of the whole software code, using dynamic techniques and a coarse grain fabric.

The ERA project advances the state-of-the art by taking another route in reconfigurability: instead of trying to accelerate applications by reconfigurable functionality (which is hard to use) ERA relies instead on architectural reconfigurability for optimizing applications in terms of performance *and* power. The processor (as a single core or as a collection of cores on the chip) is reconfigured to match the computation needs of the application.

Memory Hierarchy Reconfiguration. Similarly to processor reconfiguration, the memory system is also reconfigured for the application. There is a significant body of work concerning the reconfiguration of the memory hierarchy, primarily for high-performance computing but also with some sparse examples for embedded computing. Similarly to processor reconfiguration, caches have been a prime target for **resizing** to save power. Although it is always better to have more cache, in many cases using much less cache can save considerable power while giving up very little in performance. There are several proposals for resizing the cache to fit program needs, most important among them: Variable L1/L2 division [16], Selective Cache Ways [17], the Accounting Cache [18], and Miss Tag Resizing for CAM-Tag Caches [19] (which is well suited for embedded architectures). These techniques differ on the memory partitioning technique used (e.g., partitioning at the memory bank/segment level, cache-way level, or even within a bank using bit-line segmentation), and the policy used to drive the partitioning. Both within and across such techniques there are significant differences in power and performance depending on the target application. The policies proposed to drive such techniques are tuned to deliver the desired results for some benchmark suite so there is no clear consensus what works and what does not for our target application domain. Furthermore, these cache resizing techniques were proposed in isolation without taking into account any possible processor reconfiguration. There are very few proposals for holistic reconfiguration of both the processor and the cache hierarchy, but even those do not take fully into account the complex interaction of the joint reconfiguration of these two components; they use instead blanket performance/power feedback policies.

Beyond resizing, critical role in the power/performance of the cache hierarchy play the techniques for reducing static power dissipation (DVFS [20] and/or decay techniques [21]) and techniques to optimize the switching activity (dynamic power) for well known cache architectures. The toolkit available for optimizing the memory hierarchy includes [20]: various power-efficient associative cache architectures (phased caches, way prediction, etc.), filter caches, loop caches, and trace caches, data compression techniques (value locality compression, zero compression, etc.), and coherence optimization techniques.

In the ERA project the reconfigurable memory architecture is based on the needs and requirements of our target application domain, i.e., embedded applications. Mechanisms for memory reconfiguration will be unified according to the synergy they show (again in relation to embedded applications). Furthermore, the management policies for such reconfiguration techniques will be application driven, rather than blanket policies that cater to the general case but can be rather inefficient for specific applications or even individual application phases. We will use the necessary hardware monitoring coupled with the application-fed requirements to construct application-driven policies that can achieve various user objectives. Of critical importance is that these policies will work in concert with the processor reconfiguration proposed in our project. Our holistic approach will be expanded to cover multiprocessor SoCs and their memory hierarchies, multiprogramming in such environments (e.g., multiple programs running concurrently and sharing the chip resources), and reconfiguration of the communication infrastructure (including cache coherence protocols and networks-on-chip) specifically for parallel applications running on multiple heterogeneous cores and/or accelerators.

In the ERA project the whole compilation tool-chain can act as the mediator between the features and requirements exposed by the applications (e.g., memory locality, memory bandwidth requirements, and processing needs, as well as execution phases, multi-threaded behavior, communication, and synchronization) and the reconfiguration potential of the architecture. In particular, different modules of the architecture will have the ability to reconfigure themselves according to the (in-hardware) perceived runtime conditions and resource usage. In the integrated approach of ERA we complement this hardware monitoring and adaptation by leveraging software development and tuning tools that are able to implement a range of optimization strategies of different complexities, aiming to induce the most suitable architectural reconfigurations that match the application requirements during runtime. Using this approach, the optimization tools can take advantage of the global view of the architecture together with the application structure and needs, in order to trigger profitable reconfiguration actions.

10.6 Conclusions

In the ERA project, we are addressing the power and memory walls that bottlenecks in the design of embedded systems to satisfy the demand for more performance. More specifically, we are focusing on the design of the embedded processor and its environment. We identified a clear trend in the design of embedded processors that is moving away from application-specific implementations to more general-purpose approaches. This is not to say that application-specific implementations are no longer important, but they are only (economically) viable when there is a large mass market to amortize the design costs or when the “best” solution is sought after (independent of costs).

In this trend, we strongly believe that reconfigurable hardware will play an important role as it is capable of providing adequate performance (for most embedded applications) while maintaining a good level of flexibility that is needed in current-day embedded systems market, in which time-to-market times are shrinking. Of course, the mentioned power and memory walls are also present in the reconfigurable embedded processor scenario, and in the ERA project, we are addressing them at the same time. We believe that reconfigurability can bring forward new solutions to allow us to scale the walls. As the embedded market is moving towards a more structured design approach, we focus in the ERA project on three main elements: computing elements, memory elements, and networking elements. For each type of element, we will propose new solutions on how to exploit reconfigurability to provide more performance, but at the same time limit power consumption and better scale memory requirements. In this chapter, we also provided some evidence why we believe that our approach will be successful, and the effective need of reconfiguration in different structures to bypass the memory and power wall. In the end, one of the goals of the ERA project concerns building a proof-of-concept demonstrator to show the benefits of our approach and which can be immediately used by the industry for prototyping purposes.

Acknowledgments We would like to acknowledge the Valorisation Centre in Delft, and in particular two persons: Linda Roos and Theresia Twickler. They are responsible for the (financial) management of the project and their contributions/efforts allowed all the researchers involved in the ERA project to properly focus on their research tasks.

References

1. Bhavishya Goel, Sally A. McKee, Roberto Gioiosa, Karan Singh, Major Bhadauria, Marco Cesati, "Portable, scalable, per-core power estimation for intelligent resource management," Greencomp, International Conference on Green Computing, pp.135–146, 2010.
2. M. Alioto, P. Bennati, R. Giorgi "Exploiting locality to Improve Leakage Reduction in Embedded Drowsy I-Caches at Same Area/Speed", In Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), pp. 37–40, May 2010.
3. G. Hamerly, E. Perelman, J. Lau, B. Calder, "Simpoint 3.0: Faster and more flexible program phase analysis", In Journal of Instruction Level Parallelism, vol 7, 2005.
4. N. Puzović, S. A. McKee, R. Eres, A. Zaks, P. Gai, S. Wong, R. Giorgi, "A Multi-Pronged Approach to Benchmark Characterization" in IEEE International Conference on Cluster Computing Workshops and Posters (CLUSTER WORKSHOPS), pp.1–4, 2010.
5. C. Zeferino, A. Susin, "SoCIN: A Parametric and Scalable Network-on-Chip" in 17th Symposium on Integrated Circuits and System (SBCCI), 2003, pp. 169–174.
6. Bertozzi, D. et al., "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip", IEEE Transaction on Parallel and Distributed System, 2005, pp. 113–129.
7. K. Srinivasan and K. S. Chatha, "A Low Complexity Heuristic for Design of Custom Network-on-Chip Architectures," in Proceedings of Design, Automation and Test in Europe Conf.,vol. 1, 2006, pp. 1–6.
8. J. Andrews, N. Baker, "Xbox 360 System Architecture", IEEE Micro, vol. 26, no. 2, 2006, pp. 25–37.

9. Goldstein S. C., Schmit H., Budiu M., Cadambi S., Moe M., and Taylor R. R., “PipeRench: A Reconfigurable Architecture and Compiler”. Computer 33, 4, 70–77, 2000.
10. Vassiliadis S., Wong S., Gaydadjiev G., Bertels K., Kuzmanov G., and Panainte E.M., “The MOLEN Polymorphic Processor”. IEEE Transactions on Computers, 53, 11, 1363–1375, 2004.
11. Hauser J. R., and Wawrzynek, J., “Garp: a MIPS Processor with a Reconfigurable Coprocessor”. In Proceedings of the 5th IEEE Symposium on FPGA-Based Custom Computing Machines (FCCM), 1997.
12. Waingold E., Taylor M., Srikrishna D., Sarkar V., Lee W., Lee V., Kim J., Frank M., Finch P., Barua R., Babb J., Amarasinghe S., and Agarwal A., ”Baring It All to Software: Raw Machines”. Computer 30, 9, 86–93, 1997.
13. Lysecky R., Stitt G., Vahid F., “Warp Processors”. In ACM Transactions on Design Automation of Electronic Systems (TODAES), pp. 659–681, 2006.
14. Clark, N., Kudlur, M., Park, H. Mahlke, S., Flautner, K., “Application-Specific Processing on a General-Purpose Core via Transparent Instruction Set Customization”. In International Symposium on Microarchitecture (MICRO-37), pp. 30–40, Dec. 2004.
15. Beck A.C.S., Rutzig Mateus B., Gaydadjiev G., Carro, L., “Transparent Reconfigurable Acceleration for Heterogeneous Embedded Applications,” in Proceedings of the Design, Automation and Test Conference (DATE), pp.1208-1213, 2008.
16. D.H. Albonesi, “Dynamic IPC/clock rate optimization,” in Proceedings of the 25th International Symposium on Computer Architecture (ISCA-25), 1998.
17. D.H. Albonesi, “Selective Cache Ways: On-demand Cache Resource Allocation”, in Proceedings 32nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-32), pp. 248–259, 1999.
18. S. Dropsho, et.al., “Integrating adaptive on-chip storage structures for reduced dynamic power,” in Proceedings of the International Conference on Parallel Architectures and Compilation Techniques, 2002.
19. M. Zhang and K. Asanovi’c, “Highly Associative Caches for Low-power Processors”, Kool Chips Workshop, 33 rd International Symposium on Microarchitecture, 2000.
20. S. Kaxiras and M. Martonosi, “Architectural Techniques for Low Power”, Morgan & Claypool Publishers, 2008.
21. S. Kaxiras, Z. Hu, and M. Martonosi, “Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power”, in Proceedings of 28th International Symposium on Computer Architecture (ISCA-28), 2001.