Simultaneous Reconfiguration of Issue-width and Instruction Cache for a VLIW Processor

Fakhar Anjam and Stephan Wong
Computer Engineering Laboratory
Delft University of Technology
Mekelweg 4, 2628CD, Delft
The Netherlands
Email: {F.Anjam, J.S.S.M.Wong}@tudelft.nl

Luigi Carro, Gabriel L. Nazar, and Mateus B. Rutzig
Instituto de Informatica
Universidade Federal Do Rio Grande Do Sul
Av. Bento Gonalves, 9500 - Campus do Vale - Bloco IV
Bairro Agronomia - Porto Alegre - RS - Brasil
Email: {carro, glnazar, mbrutzig}@inf.ufrgs.br

***Abstract*—This paper presents an analysis on the impact of simultaneous instruction cache (I-cache) and issue-width reconfiguration for a very long instruction word (VLIW) processor. The issue-width of the processor can be adjusted at run-time to be 2-issue, 4-issue, or 8-issue, and the I-cache can be reconfigured in terms of associativity, cache size, and line size. We observe that, compared to reconfiguring only the I-cache for a fixed issue-width core, reconfiguring the issue-width and I-cache together can further reduce the execution time, energy consumption, and/or the energy-delay product (EDP). The results for the MiBench and the PowerStone benchmark suites show that compared to "2-issue + the best I-cache", "4-issue + the best I-cache" can reduce execution time, energy consumption, and EDP by up to 37%, 11%, and 36%, respectively, for different applications. Similarly, compared to "2-issue + the best I-cache", "8-issue + the best I-cache" can reduce execution time and EDP by up to 46% and 30%, respectively, for different applications.**

I. Introduction

Increasing the issue-width for a VLIW processor increases the performance for applications with inherent instruction level parallelism (ILP). Figure 1 presents the instructions per cycle (IPC) for some applications from different benchmark suites (MiBench [1], PowerStone [2]) for 2-issue, 4-issue, and 8-issue VLIW processors with a single load/store unit. As depicted in the figure, the IPC increases with the issue-width for applications with more ILP. Specializing a cache for a processor may improve the performance or energy consumption for one benchmark, but may perform poorly across others [3]. Studies have shown that more than half of the chip die is reserved for the on-chip caches and that the energy consumption in cache systems accounts for more than 50% of the total energy consumption [2][4][5][6][7]. Table I presents the instruction cache parameters for some commercial/research VLIW processors. As can be observed, there is a wide variation across different cache parameters (associativity, cache size, and line size). Compared to having a fixed cache, reconfiguring the cache for a processor at run-time can reduce the execution time, energy consumption, and/or the EDP across different benchmarks [3][5][7][8][9][10][11]. Compared to reconfiguring only the cache, reconfiguring the "issue-width + cache" can further improve the execution time, energy consumption, and/or the EDP.

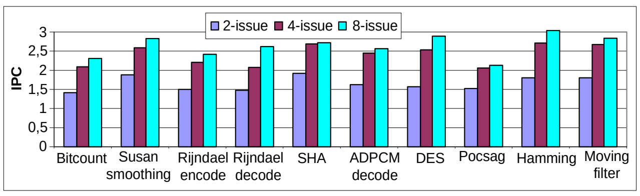

Fig. 1. Instructions per cycle (IPC) for different applications [1][2].

In this paper, we study the impact of I-cache reconfiguration on the performance, dynamic energy consumption, and the EDP when the issue-width of a VLIW processor is changed. We define the EDP as the product of energy consumed per application and the total execution cycles. When the issue-width is changed, a different schedule is followed by the compiler and a different request for instructions and data is generated. Since this request determines the performance, energy consumption, and the EDP, we analyze how this request can be better fulfilled by tuning the available I-cache. The fundamental question to answer here is: given reconfigurable processors, does it makes sense to perform I-cache reconfiguration? In particular, we try to answer the following questions:

- Given an I-cache configuration, is there a processor configuration (issue-width) that reduces the execution time and/or energy consumption?
- Given a processor configuration (issue-width), is there an I-cache configuration that reduces the execution time

TABLE I
Typical instruction cache parameters for some famous VLIW processors.

Processor	Issues	Assoc.	Size (Kbytes)	Line size (bytes)
TriMedia TM32A	5	8	32	64
TriMedia TM3270	5	8	64	128
TMS320C611	8	1	4	64
ST231	4	1	32	64
ST240	4	4	32	64
Transmeta TM5400	5	8	64	-
Fujitsu FR450	2	2	32	32
CoreVA	4	1	16	64

and/or energy consumption?

- Given a set of processor configurations (issue-widths), and assuming that one will reconfigure them for performance, is there an I-cache configuration that reduces the energy consumption for the same execution time?

We present a design and analysis where both the issue-width and the I-cache can be reconfigured simultaneously. Notice that if different “issue-width + I-cache” configurations have the same execution times, but reduced energy consumptions or vice versa, it may be beneficial to reconfigure the core issue-width, the cache, or both. We analyze different “issue-width + I-cache” configurations. We designed and implemented a ρ-VEX VLIW processor that can be configured to be 2-issue, 4-issue, or 8-issue at run-time. The unused issue-slots are clock-gated to reduce dynamic power consumption of the processor. We considered an instruction cache that can be reconfigured in terms of associativity, total cache size, and line size. We utilized the VEX simulator [12] to simulate different “issue-width + L1 I-cache” configurations. For energy calculation, we utilized the CACTI [13] and the Synopsis Design Compiler for a 90nm technology. We utilized the MiBench [1], PowerStone [2], and custom (16 small applications/kernels from different domains) benchmark suites. We found that given a processor configuration (issue-width), there is always an optimum I-cache configuration which performs better in terms of energy consumption, execution time, and/or the EDP compared to other cache configurations. Additionally, we found that, compared to tuning only the I-cache for a fixed issue-width core, tuning the “issue-width + I-cache” has more potential to reduce the energy consumption, execution time, and/or the EDP for different applications. The results show that compared to “2-issue + the best I-cache”, “4-issue + the best I-cache” can reduce execution time, energy consumption, and EDP by up to 37%, 11%, and 36%, respectively, for different applications. Comparing “2-issue + the best I-cache”, to “8-issue + the best I-cache”, execution time and EDP can be reduced by up to 46% and 30%, respectively, for different applications.

The remainder of the paper is organized as follows. Motivations for the paper are presented in Section II. Section III discusses some related work. Our reconfigurable issue-width VLIW processor and instruction cache, configuration parameters, and experimental setup are presented in Section IV. Section V presents in detail the experimental results and analysis. Finally, Section VI concludes the paper.

II. Motivations

Instruction cache reconfiguration plays an important role in the performance, energy consumption, and EDP for different applications. Figure 2 depicts the execution cycles, dynamic energy consumption, and EDP when the size of a direct-mapped I-cache with 16 bytes line size is changed from 4 Kbytes to 32 Kbytes for the ADPCM encode and the Susan edges applications from the MiBench benchmark suite. The issue-width of the core is kept fixed at 2. As depicted in the figure, the execution cycles for the Susan edges decrease with increasing the I-cache size, while that for the ADPCM encode remains almost the same. The reason is the decrease in the number of cache misses for the Susan edges. The cache size has no effect on the cache misses for the ADPCM encode. Normally, a larger cache consumes larger energy per access, hence, for the ADPCM encode, the energy consumption increases with increasing the cache size, as the number of misses remains the same. For the Susan edges, although the per-access energy increases with the larger cache size, but the number of misses reduces, hence, the dynamic energy consumption reduces with increasing the cache size. It is to be noted that compared to a cache hit, the energy consumption due to a cache miss is much higher. The result is that the EDP for the Susan edges decreases with increasing the I-cache size while that for the ADPCM encode increases.

Instead of reconfiguring only the I-cache, reconfiguring both the “issue-width + I-cache” can further improve the performance, energy consumption, and the EDP. Figure 3 depicts execution cycles, speedup, dynamic energy consumption, and EDP for the Rijndael decode application for a direct-mapped I-cache with varying cache size and line size for 2-issue and 4-issue VLIW processors. As depicted in the figure, changing the cache parameters changes the execution cycles for both the 2-issue and 4-issue cores. Compared to the 2-issue, the 4-issue core always performs better in terms of execution cycles for all of the cache configurations. The 4-issue core performs better up to 1.5x compared to the 2-issue core for a properly tuned I-cache. As depicted in the figure, considering a fixed cache line size (16 or 32 bytes), the energy consumption varies for the 2-issue and 4-issue cores with varying cache sizes (4 to 32 Kbytes). The energy consumption also varies with line size for a fixed cache size. For the same cache parameters, the energy also varies for the 2-issue and 4-issue cores. Similarly, the EDP varies largely with varying the “issue-width + I-cache” configurations. Hence, by properly tuning the “issue-width +

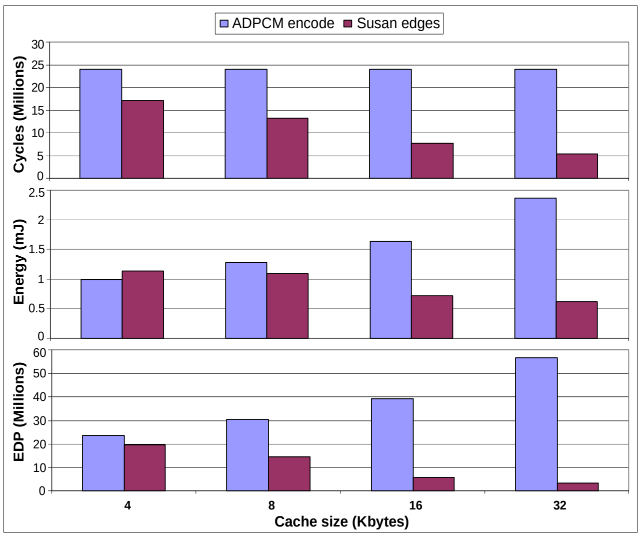

Fig. 2. Impact of cache size reconfiguration on execution cycles, dynamic energy consumption, and EDP for a direct-mapped instruction cache with 16 bytes line size for a 2-issue processor.

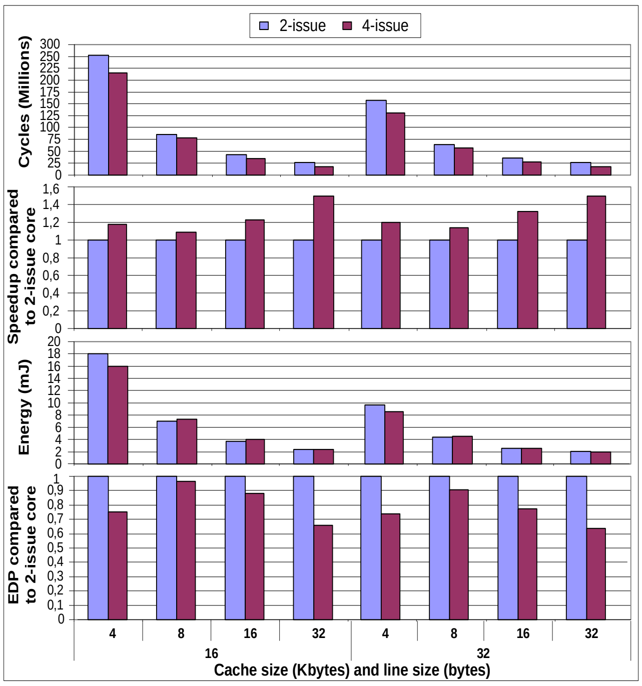

Fig. 3. Dynamic energy consumption, execution cycles, speedup, and EDP for the Rijndael decode.

I-cache" parameters in Fig. 3, the execution cycles and the EDP can be reduced by up to 33% and 40%, respectively, in the case of the best/optimal configurations.

III. Related Work

The impact of cache parameters such as total size, line size, associativity, replacement policies, etc., on performance and energy consumption for different levels of caches (L1 and L2) has been widely reported. A reconfigurable cache memory with heterogeneous banks to reduce the cache size and hence the power consumption at run-time is presented in [14]. Reconfigurable aspects of the cache system for the TMS320C6211 processor are discussed in [15]. The 4-way unified L2 cache can be used as either mapped RAM or as 1, 2, 3, or 4 ways cache. Each way or bank is 16 Kbytes. A reconfigurable data cache design with a hardware-adaptive line size for miss rate and memory traffic reduction is presented in [8]. The paper does not discuss energy consumption.

Selective cache ways [5] provides the ability to disable a subset of the ways in a set-associative cache to reduce the energy consumption for little performance overhead. A mechanism for tuning cache ways and voltage scaling for embedded system platforms to reduce energy consumption is presented in [16]. Way predictive set-associative caches [10][17] provides the ability to reduce energy consumption at the expense of longer average access time. The design presented in [11] dynamically divides the cache arrays into multiple partitions that can be used for different processor activities to increase the performance. A novel set and way management cache architecture for efficient run-time reconfiguration (Smart cache) is presented in [18], providing reconfigurability across cache size and associativity. A hybrid selective-sets-and-ways cache organization is proposed in [19] that always offers equal or better resizing granularity than both of the selective-sets and selective-ways organizations. The impact of line size on energy consumption and performance for instruction and data caches is presented in [9]. Designs of configurable caches where all the three parameters (associativity, cache size, and line size) can be configured are presented in [7][20]. It must be noted that all these papers present results for cache configurations with fixed issue-width processors.

Superscalar and VLIW are two main architectures that can exploit ILP. A superscalar processor determines the active issue-width at run-time by resolving dependencies among the in-flight instructions. A VLIW processor considers a fixed issue-width at compile time and then splits the incoming long instruction among the available functional units (FUs) at run-time, greatly simplifying the hardware and reducing the energy consumption [21]. The commonly used commercial VLIW processors such as the TriMedia series from NXP, ST231 from STMicroelectronics, TMS320C611 from Texas Instruments, Crusoe from Transmeta, and the FRxxx series from Fujitsu all utilize a fixed issue-width. Reconfiguring the issue-width at run-time improves the performance of applications with higher ILP. Core fusion [22], TRIPS [23] and Voltron [24] combine small cores and the on-chip memory to make larger issue-width cores at run-time to exploit the instruction, data, or thread-level parallelism. Smart memories [25] is a reconfigurable architecture capable of merging in-order RISC cores to form a VLIW machine. All these papers present results for performance/speedup for the available configurations of the system but do not discuss the energy consumption.

Our reconfigurable issue-width processor design [26] is based on the ρ-VEX VLIW processor [27], having small 2-issue in-order cores that can morph at run-time to make a larger issue-width core to exploit ILP. The processor issue-width can be configured to be 2-issue, 4-issue, or 8-issue at run-time. We consider a wide range of run-time I-cache configurations (associativity, cache size, and line size) for our adaptable issue-width processor. This paper analyzes the performance and energy benefits of simultaneous reconfiguration of issue-width and I-cache for our VLIW processor.

IV. The Reconfigurable Processor and Instruction Cache

In this section, we present the design and configuration parameters for our reconfigurable VLIW processor and instruction cache. In this paper, we present results only for instruction cache, hence, in the remainder of the paper, the word cache or I-cache is referring to an instruction cache.

A. *The VEX System: ISA and Toolchain*

The VEX (VLIW Example) instruction set architecture (ISA) is a 32-bit clustered VLIW ISA that is scalable and customizable to individual application domains developed by Hewlett-Packard (HP) and STMicroelectronics [28]. The ISA is scalable because different parameters, such as the number and type of functional units (FUs), number of registers, and different latencies can be changed. The ISA is customizable because custom instructions can be added. The VEX ISA is loosely modeled on the ISA of the HP/ST Lx (ST200) family of VLIW embedded cores [29]. Based on trace scheduling, the VEX C compiler is a parameterized ISO/C89 compiler. A flexible programmable machine model determines the target architecture, which is provided as input to the compiler. A VEX software toolchain including the VEX C compiler and the VEX simulator is made freely available by the Hewlett-Packard Laboratories [12].

B. The ρ-VEX Reconfigurable VLIW Processor

The ρ-VEX is a configurable (design-time) open-source VLIW softcore processor [27]. The ISA is based on the VEX ISA [28]. Different parameters of the ρ-VEX processor, such as the number and type of FUs, number of multi-ported registers (size of register file), number and type of accessible FUs per syllable, width of memory buses, and different latencies can be changed at design time. Figure 4 depicts the organization of a 2-issue ρ-VEX processor. The processor is a 5-stage pipelined processor consisting of *fetch*, *decode*, *execute0*, *execute1/memory*, and *writeback* stages. The available functional units are *arithmetic logic unit* (ALU), *multiplier unit* (MUL), *branch or control unit* (CTRL), and *load/store or memory unit* (MEM). The processor has a 64×32-bit multiported general-purpose register file (GR) and an 8×1-bit multiported branch register file (BR).

C. The Run-time Adjustable Issue-slots Processor

In [26], we designed and implemented a run-time reconfigurable processor utilizing 2-issue ρ-VEX cores. This processor has four 2-issue ρ-VEX cores as depicted in Fig. 4. Multiple 2-issue cores can be combined at run-time to make a larger issue-width core. With the four 2-issue cores, we can have a VLIW processor at run-time of the following type:

- 2-issue
- 4-issue or
- 8-issue

Figure 5 depicts the general view of the run-time adjustable issue-slots VLIW processor. There are different functional units and their type and composition can be changed. We adjusted the design presented in [26] for the analysis in this paper. The adjusted processor consists of 8 ALUs, maximum of 4 multipliers (MULs), one branch unit (CTRL), and one load/store (MEM) unit. The 2-issue, 4-issue, and 8-issue cores can use 2 ALUs and 2 MULs, 4 ALUs and 4 MULs, and 8 ALUs and 4 MULs, respectively. From profiling results of the considered benchmarks, we determined that increasing the number of MULs from 4 in case of the 8-issue core has no effect on performance; hence we kept the maximum number of MULs to 4. The processor has a 64×32-bit multiported register file with 8-write-16-read (8W16R) ports. The register ports are distributed among the 8 issue-slots in order to provide access to the unified registers as required by the ISA. The issue-slots reconfiguration is accomplished through a configuration register. Two bits in the configuration register control the issue-width of the processor. When the *issue_ctrl* bits are “00”, the system behaves as a 2-issue processor, when “01”, the system behaves as a 4-issue processor, and when “1x”, the system behaves as an 8-issue processor. The issue-width is changed in a single cycle. Additionally, the *issue_ctrl* bits are used to clock gate the unused functional units and parts of the processor system to reduce the dynamic power consumption. The 2-issue, 4-issue, and 8-issue core can issue a VLIW instruction consisting of 2, 4, and 8 RISC-style operations, respectively, every cycle.

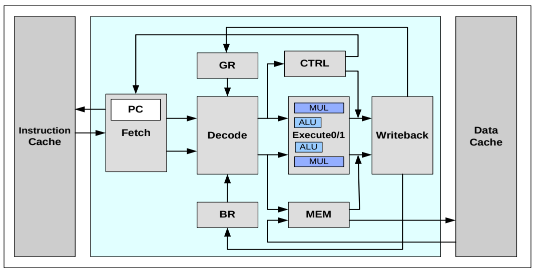

Fig. 4. A 2-issue ρ-VEX VLIW processor.

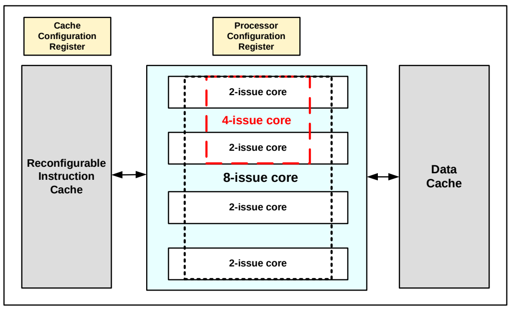
Cache Configuration Register
Processor Configuration Register
Reconfigurable Instruction Cache
2-issue core
4-issue core
2-issue core
8-issue core
2-issue core
2-issue core
Data Cache

Fig. 5. General layout of the run-time adjustable issue-slots VLIW processor.

Currently, our processor supports only single-tasking computation. Multitasking or multi-threading support is not available. When an application starts executing, it is allowed to finish completely and then a new application is started. Hence, we do not need any complex mechanisms for task preemption, and the design becomes very simple. The processor issue-width can be reconfigured in a single cycle when *issue_ctrl* bits are written to the configuration register. The reconfiguration is needed per application basis. The request to change the issue-width remains pending until the currently running application finishes execution. The request to change the issue-width for the new application can be placed in the currently running code or it can be placed in the start of the new application. This request is placed in a custom operation that writes *issue_ctrl* bits to the configuration register. When the configuration register is written, the processor reconfigures the issue-width.

Our processor can be configured to be 2-issue, 4-issue, or 8-issue with different number of load/store (LS) units, but for this paper, we kept the number of LS units to be 1 for every type of our processor issue-width. The reason is that we do not want to include the discussion about the data cache and only focus on the instruction cache. When the number of LS units is increased, the number of data words loaded/stored per cycle can be increased which can improve the performance of certain applications. We consider a single LS unit so that the energy due the data cache remains the same for every type of processor issue-width.

D. The Reconfigurable Instruction Cache

Our reconfigurable I-cache architecture includes three parameters: cache associativity, cache size, and line size. According to Table I, there is a wide variation across the cache parameters, therefore, we utilized the following parameters for our reconfigurable cache:

- Cache associativity: 1/2/4/8 ways
- Cache size: 4/8/16/32 Kbytes
- Cache line size: 16/32/64 bytes

In our system, the total cache (in all parameters) is available to all types of the configured issue-width cores. In this analysis, we only discuss the dynamic energy. We calculate the energy according to [20] which takes into account the energy consumption from the complete instruction memory hierarchy including the external memory. The architecture of our reconfigurable cache is also based on [20]. Following are the 3 cache parameters and their methods of reconfiguration.

1) Cache Associativity; Way Concatenation: For reconfiguration of cache associativity, the way concatenation technique is used [20]. The base cache includes eight banks that can operate as eight ways. By writing to the configuration register, the ways can be effectively concatenated, resulting in either a four-way, two-way, or direct-mapped 32 Kbytes cache.

2) Cache Size; Way Shutdown: For reconfiguration of cache size, the way shutdown technique is used. With way shutdown, the 32 Kbytes 8-way cache can be reconfigured as a 16 Kbytes cache that can be either 4-way, 2-way or direct mapped, an 8 Kbytes cache that can be either 2-way or direct mapped, or a 4 Kbytes direct mapped cache.

3) Cache Line Size; Line Concatenation: For reconfiguration of line size a base physical line size of 16 bytes is implemented, with larger line sizes implemented logically as multiple physical lines [20]. By writing to the configuration register, line size can be reconfigured as 16, 32, or 64 bytes.

E. Characteristics of the Instruction Cache Design

The I-cache can be reconfigured at run-time by writing a small number of bits in the cache configuration register. The cache reconfiguration can be done in a single cycle after the configuration register is written. Because our processor does not support multi-tasking, the cache reconfiguration is required only when application changes. There is no need for cache flushing, no reconfiguration overhead, and hence, the cache reconfiguration time is reduced. The cache design is simple as no complex cache flushing logic is needed. Applications are profiled statically utilizing the available issue-width and cache parameters, and the best cache for a particular issue-width can be selected based on energy consumption, performance, or EDP metrics. In our case, the best/optimal cache for a particular issue-width and application is a cache with optimal performance and reasonable energy consumption. Information about the best configuration (issue-width + I-cache) can be stored in a program executable, which are written to the issue-width and cache configuration registers, and the issue-width and the cache can be reconfigured before the application starts execution. We do not need any run-time methods and policies for reconfiguration of the processor and the cache, as we reconfigure the system per application basis, and we can do static profiling in order to determine the best configurations.

F. Experimental Setup and Benchmark Applications

We utilized the VEX toolchain [12] which includes a parameterized C compiler and a simulator. The compiler reads a machine configuration file and then compiles and schedules the code according to the machine specifications. For our analysis, there are 48 I-cache configurations and 3 issue-width configurations; hence the total search space for each application is 144 “issue-width + I-cache” configurations. The simulator generates an extensive log file containing different information, such as total memory accesses, total misses, execution cycles, stall cycles, function profiles etc. For energy calculation, we utilized CACTI [13] and Synopsis Design Compiler and targeted 90nm technology. We utilized the MiBench [1], PowerStone [2], and custom benchmark suites for the analysis. The custom benchmark suite includes the following 16 small applications/kernels from different domains: discrete cosine transform (DCT), discrete Fourier transform (DFT), finite impulse response filters (FIR), Floydwarshall graph, Hamming distance, Huffman compression and decompression, inverse discrete cosine transform (IDCT), matrix multiply, moving filter, run length encoding (RLE), different sorting applications such as bubblesort, quicksork, radixsort, and shellsort. Due to the space limit, we cannot present results for all of the benchmark applications.

G. Energy Calculation

To calculate dynamic energy consumption in I-cache, we utilized CACTI [13] and targeted 90nm technology. We obtained energy per access for all of the cache configurations with CACTI. We then calculated the total cache energy consumption taking into account both the hit and miss energies with the following equation:

$$\begin{aligned} Cache\_Energy &= Accesses * Energy/access + Misses * Energy/miss \\ &= Accesses * Energy/access + Misses * Kmiss * Energy/access \\ &= (Accesses + Kmiss * Misses) * Energy/access \end{aligned} \tag{1}$$

Kmiss is a factor representing a multiple of the cache hit energy consumption. According to [20] which takes into account the energy consumption from the complete instruction memory hierarchy including the external memory, the value of *Kmiss* ranges from 50 to 200. For this paper, we consider the *Kmiss* to be 50. We utilized the VEX toolchain [12] to simulate all the benchmark applications. Each application is simulated 144 times to generate total memory accesses, cache hits, cache misses, and execution cycles statistics for the 144 "issue-width + I-cache" configurations. Using equation (1), we calculated the I-cache energy consumption for each application with 144 different configurations. The cache energy/access is obtained from CACTI for each cache configuration.

For calculating energy consumption in the processor, we utilized the Synopsis Design Compiler (Synthesis-E-2010.12-SP1) to get the power consumption for 90nm technology. We kept the working frequency at 100 MHz. We then calculated the processor energy consumption for all applications with the following equation:

$$Processor\_Energy = Power\_consumed * Cycle\_time * Execution\_cycles \qquad (2)$$

The total energy consumption and energy-delay product (EDP) are calculated with the following equations:

$$Total\_Energy = Processor\_Energy + Cache\_Energy \qquad (3)$$

$$EDP = Total\_Energy * Execution\_cycles \qquad (4)$$

Execution_cycles, Total_Energy, and EDP for each benchmark application with 144 different "issue-width + I-cache" configurations are calculated and then analyzed in Section V.

V. Results and Analysis

As stated earlier, when the issue-width is changed, a different schedule is followed by the compiler and a different request for instructions and data is generated. Since this request determines the performance, energy consumption, and EDP, we analyze how this request can be better fulfilled by tuning the available cache parameters. In this paper, we investigate the following question: given several different cache configurations with same energy consumption for different processor configurations, is it possible that different programs run with less execution time or given several different cache configurations with same execution time for different processor configurations, is it possible that different programs run with less energy consumption. In order to analyze this question, we study three types of reconfiguration for our system:

1) Reconfiguration of issue-width with respect to I-cache
2) Reconfiguration of I-cache with respect to issue-width
3) Reconfiguration of both the issue-width and the I-cache

A. *Reconfiguration of Issue-width with respect to I-cache*

Changing the I-cache configuration for any core configuration (issue-width) affects the memory accesses and miss/hit rates. These parameters have a direct influence on the application's performance, energy consumption, and hence the EDP. Similarly, changing the issue-width for a cache configuration also impacts these parameters. Figure 6 presents an analysis for 3 applications; Basicmath, ADPCM decode (D-adpcm), and Rijndael encode (E-rijndael) for the 3 configurations of our processor issue-width with varying the I-cache configurations. Here, 1W8KB16B means a cache with 1 way associativity, 8 Kbytes total size, and 16 bytes line size. This is the base cache. We vary the cache in all its three parameters, i.e., doubling the size, the line size, and the associativity. The first graph in Fig. 6 depicts the speedup normalized to "2-issue core + 1W8KB16B I-cache" configuration. Focusing at the Basicmath application, we can observe that there is no effect of changing the issue-width; hence, for all issue-widths, the speedup is similar at varying I-cache configurations. We can observe that for any issue-width configuration, varying the I-cache configuration does vary the performance as well as the energy consumption, but this change remains same across all the issue-widths. When any of the execution time or energy consumption changes, the EDP changes accordingly.

Focusing at the D-adpcm application, we can observe that varying the I-cache configurations has no effect on the speedup for different issue-width configurations. The speedup increases only with the issue-width reconfiguration. On the other hand, the energy consumption varies with the issue-width and hence the EDP. Considering the Rijndael encode application, we can observe that both the issue-width and the I-cache configurations impact the performance and energy consumption. The "8-issue + 1W8KB32B I-cache" results in the highest performance, the least energy consumption, and hence the least EDP. This shows that the issue-width reconfiguration becomes necessary in certain cases to reduce the execution time and/or the energy consumption.

B. *Reconfiguration of I-cache with respect to Issue-width*

In this case, we study and analyze the effect of varying the I-cache configurations on the core's issue-width configuration. Figure 7 depicts the impact of I-cache reconfiguration on

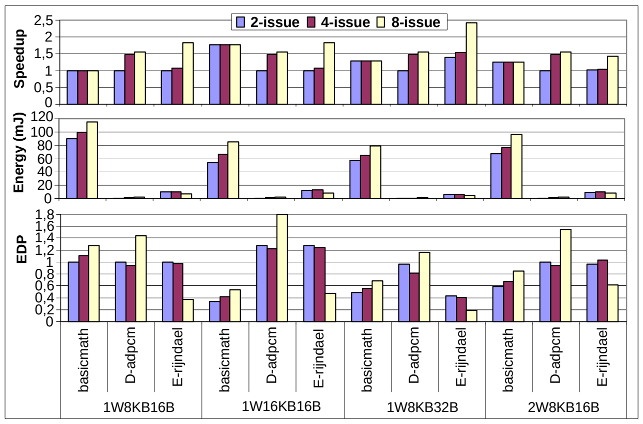

Fig. 6. Impact of issue-width with respect to I-cache; speedup, energy, and EDP normalized to 2-issue and 1W8KB16B I-cache.

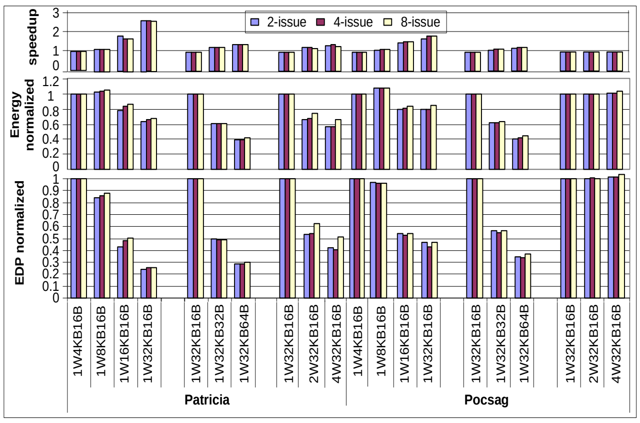

Fig. 7. Impact of I-cache on issue-width; speedup, energy and EDP for 2-issue, 4-issue, and 8-issue cores with varying I-cache normalized to own issue-width with 1W4KB16B I-cache.

the issue-width. We consider an I-cache of 1W32KB16B that is varied in different dimensions. The first, second, and third cache sets are 1W(4-8-16-32)KB16B (varying cache size), 1W32KB(16-32-64)B (varying line size) and (1-2-4)W32KB16B (varying the associativity), respectively. The speedup, energy, and EDP for each issue-width configuration are normalized to that of the "own issue-width + the smallest I-cache" configuration. The smallest I-cache in the considered caches is 1W4KB16B.

Considering the Patricia application, when the cache is varied for each type of the issue-width in the first cache set, the speedup, energy consumption, and EDP are improved compared to that of the same issue-width with 1W4KB16B I-cache. When the cache is varied in the second and third sets, there is a small variation in the speedup, but there is a big variation in energy consumption. It must be noted that the performance does not change with the issue-width; rather it only changes with varying the cache. In case of the Pocsag application, performance and energy consumption changes with varying the I-cache in first and second cache sets for each issue-width, but there is no effect for the third cache set, meaning that the associativity has almost no effect on any issue-width's execution cycles, energy consumption and EDP. Hence, we can say that for a fixed issue-width, changing the I-cache almost always affects the performance, energy consumption, and/or the EDP. The advantage in our case is that we can also change the issue-width, which can also produce some positive effect.

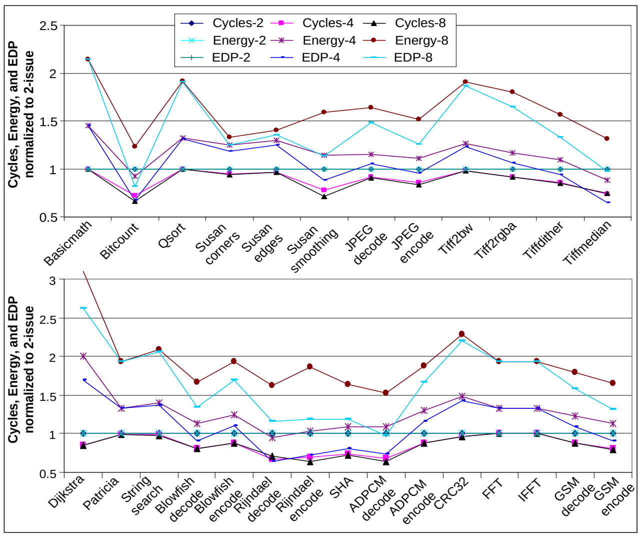

Fig. 8. Execution cycles, energy consumption, and EDP for the 4-issue and 8-issue cores normalized to 2-issue core (all with their best I-caches) for the MiBench benchmarks.

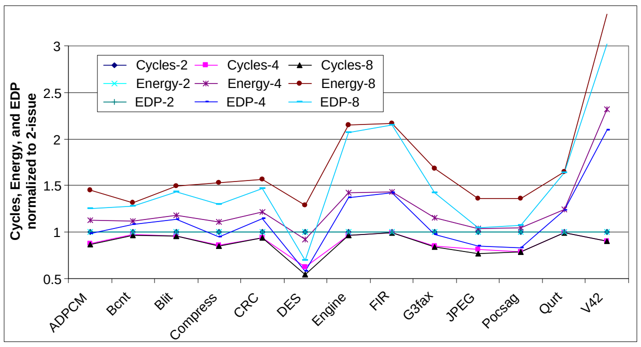

Fig. 9. Execution cycles, energy consumption, and EDP for the 4-issue and 8-issue cores normalized to 2-issue core (all with their best I-caches) for the PowerStone benchmarks.

C. Reconfiguration of both the Issue-width and the I-cache

The ability to reconfigure both the issue-width and the I-cache brings the advantage of both reconfigurations. The system can be reconfigured for the best performance, the least energy consumption, or both. Instead of comparing to a fixed "issue-width + I-cache", we compare the results for the 2-issue, 4-issue, and 8-issue cores with their best I-caches for all of the benchmark applications. In this way, we can optimize the performance, dynamic energy consumption, and the EDP for individual applications. Figures 8 – 10 present the execution cycles, energy consumption, and the EDP for 4-issue and 8-issue cores with their best I-caches normalized to

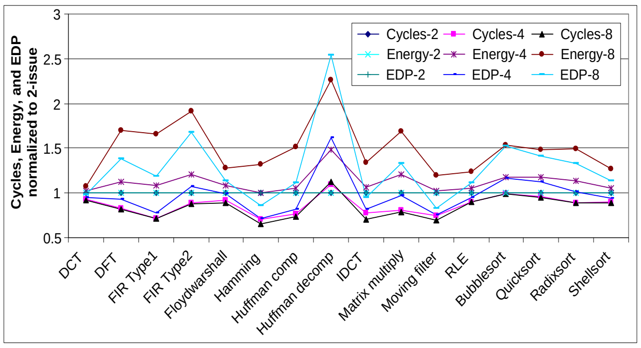

Fig. 10. Execution cycles, energy consumption, and EDP for the 4-issue and 8-issue cores normalized to 2-issue core (all with their best I-caches) for the custom benchmarks.

that of the 2-issue core with its best I-cache for the MiBench, PowerStone, and the custom benchmark suites, respectively. In these figures, the continuous lines are drawn only for clarity purpose; otherwise the values are only at discrete points. In general, we observed that switching from 8-issue core to 4-issue or 2-issue core reduces energy consumption. The main reason is that the 8-issue core reads a longer instruction (256 bits) per access from the cache while the 4-issue and 2-issue cores read smaller instructions (128 bits and 64 bits, respectively) per cache access. Additionally, the 8-issue core utilizes more functional units compared to 4-issue and 2-issue cores. On the other side, switching from 2-issue core to 4-issue or 8-issue core increases the performance as more operations can be executed in parallel. In both of these scenarios, the instruction cache is always tuned to the best configuration for each issue-width.

From Figures 8 – 10, we can observe that there are some applications/kernels such as Bitcount, Tiffmedian, ADPCM decode, DES, DCT, Hamming distance, IDCT, Moving filter, where the EDP for the 8-issue core with its best I-cache is less than or equal to that of the 2-issue core with its best I-cache. For these applications switching from 2-issue to 8-issue core increases the performance more than the energy consumption and hence reduces the EDP. The largest reduction in the EDP is for the DES application, which is about 30%. There are some applications such as Susan smoothing, Rijndael decode, Rijndael encode, SHA, JPEG, Pocsag, FIR Type1, Floydwarshall, Huffman compression, RLE, Shellsort, where, for a small increase in EDP, one can get more performance when the issue-width is changed from 2-issue to 8-issue.

Similarly, considering the 4-issue and 2-issue cores with their best I-caches, we can observe that there are about 29 applications, where the EDP for the 4-issue core is less than or equal to that of the 2-issue core. These applications are; 11 in MiBench: Bitcount, Susan smoothing, JPEG encode, Tiffdither, Tiffmedian, Blowfish decode, Rijndael encode and decode, SHA, ADPCM decode, and GSM encode, 6 in PowerStone: ADPCM, Compress, DES, G3fax, JPEG, and Pocsag, and 12 in custom benchmark suite: DCT, DFT, FIR type1, Floydwarshall, Hamming, Huffman compression, IDCT, matrix multiply, moving filter, RLE, Radixsort, and Shellsort. This means that for these applications switching from 2-issue to 4-issue core increases the performance more than increasing the energy consumption and hence reduces the EDP. Compared to “2-issue + the best I-cache”, “4-issue + best I-cache” reduces the EDP for Tiffmedian, Rijndael decode, and DES by about 30%, 36%, and 41%, respectively. Additionally, there are some applications, where with a small increase in EDP one can get more performance when the issue-width is changed from 2-issue to 4-issue.

Considering the energy consumption with the best I-caches for every issue-width, there is no case in the considered benchmarks, where the 8-issue core consumes less energy than the 2-issue or 4-issue core. There are some applications such as Bitcount, Tiffmedian, Rijndael decode, DES, Hamming distance, where a 4-issue core consumes less energy than that of the 2-issue core, both with their best/optimal I-caches. Tiffmedian and DES consume 11% and 6% less energy, respectively, on a 4-issue core compared to a 2-issue core both with their best caches. There are many applications such as Susan smoothing, JPEG encode, Tiffdither, blowfish decode, Rijndael decode, SHA, ADPCM decode, GSM encode, Compress, G3fax, JPEG, Pocsag, DCT, DFT, FIR Type1, Floydwarshall, Huffman compression, IDCT, Matrix multiply, Moving filter, RLE, Shellsort, where by switching from a 2-issue to a 4-issue core (both with their best I-caches) results in a large performance gain with a small energy increase.

Considering the execution cycles with the best I-caches for every issue-width, all the considered benchmark applications perform better with the 8-issue and 4-issue cores compared to the 2-issue core. Switching from 2-issue to 8-issue core (both with their best caches) reduces the execution cycles for Hamming distance, ADPCM decode, and DES by about 36%, 40%, and 46%, respectively. The largest reduction in execution cycles when switching from a 2-issue to 4-issue core (both with their best caches) is for the Rijndael decode application which is about 37%.

Figures 11 and 12 depict some Pareto points for energy consumption and EDP with same execution cycles at the given I-cache configurations for the Rijndael encode and ADPCM decode applications. Here, the execution cycles decreases with increasing the issue-width, but remains constant for all the considered caches. Considering Fig. 11 for the Rijndael encode application, the 2-issue core consumes less energy at every point compared to the 4-issue and 8-issue cores. As the execution cycles for the 8-issue core are less than that for the 2-issue core, the EDP for the 8-issue is lower than that for the 2-issue core at some points. The 4-issue core behaves somewhere in between the 2-issue and 8-issue cores. For the ADPCM decode application in Fig. 12, the 2-issue core consumes less energy at every point compared to the 4-issue and 8-issue cores. As the execution cycles for the 8-issue core are less than that for the 2-issue core, the EDP

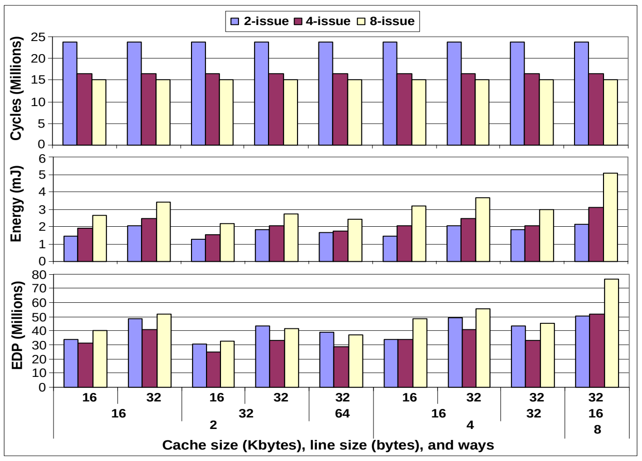

Fig. 11. Pareto points for energy and EDP with same execution cycles at all I-cache configurations for the Rijndael encode.

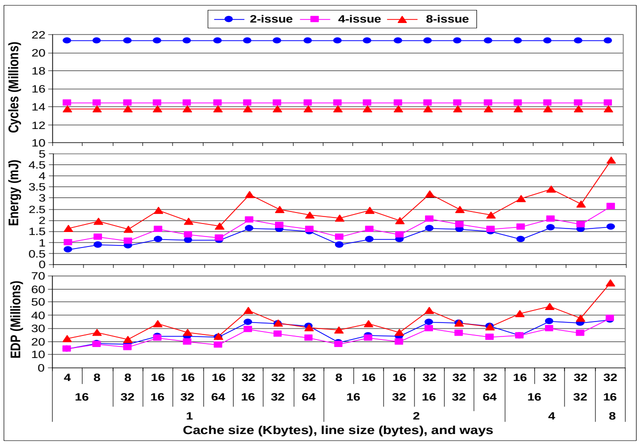
Fig. 12. Pareto points for energy and EDP with same execution cycles at all I-cache configurations for the ADPCM decode.

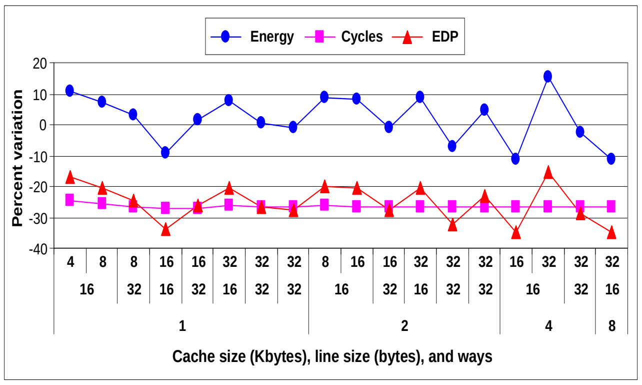
Fig. 14. Percentage variation in energy, cycles, and EDP for 4-issue core compared to 2-issue core with different I-caches for the Tiffmedian.

for the 8-issue core is equal to or less than that for the 2-issue core at some points. The 4-issue core consumes more energy than that for the 2-issue core, but its execution cycles are much lower than that for the 2-issue core and slightly more than that for the 8-issue core, hence, the EDP for the 4-issue core is lower than that for the 2-issue and 8-issue cores. Hence, if low power is the main concern, the 2-issue core can be selected, and if low execution time is the main concern, the 8-issue core can be selected. For lower EDP, the 4-issue core can be selected. By including both the issue-width and I-cache parameters, we increase the Pareto points for energy consumption and execution time and hence have a broader range of selection.

Figures 13 – 15 depict percentage variations in energy consumption, execution cycles, and EDP for the Dijkstra, Tiffmedian, and GSM encode applications when the issue-width is changed from 2-issue to 4-issue with different I-caches. Considering Fig. 13 for the Dijkstra application, when the issue-width is increased from 2-issue to 4-issue, there is a 15% reduction in execution cycles for almost each cache configuration. The energy consumption and the EDP vary from 1% to 30% and from -18% to 0.2%, respectively. The I-cache for which the 4-issue core has the least energy compared to the 2-issue core is selected to be the best “issue-width + I-cache” configuration. This cache has the least execution cycles and the least EDP. Considering Fig. 14 for the Tiffmedian application, when the issue-width is increased from 2-issue to 4-issue, there is a 27% reduction in execution cycles for almost each cache configuration. The energy consumption and the EDP vary from -11% to 15% and from -35% to -15%, respectively. Similarly, considering Fig. 15 for the GSM encode application, when the issue-width is increased from 2-issue to 4-issue, there is a 20% reduction in execution cycles for almost each I-cache configuration. The energy consumption and the EDP vary from -8% to 27% and from -24% to 2%, respectively. Hence, reconfiguring the issue-width for the same I-cache produces multiple options to choose from. If energy is the main concern, the issue-width resulting in lower energy consumption can be selected. If performance is the prime concern, the issue-width with the lower execution cycles can be selected.

Figure 16 depicts percentage variations in energy consumption, execution cycles, and EDP for the Rijndael encode application when the issue-width is changed from 2-issue to 4-issue and 8-issue with different I-caches. When the issue-width is changed from 2-issue to 4-issue, the execution cycles

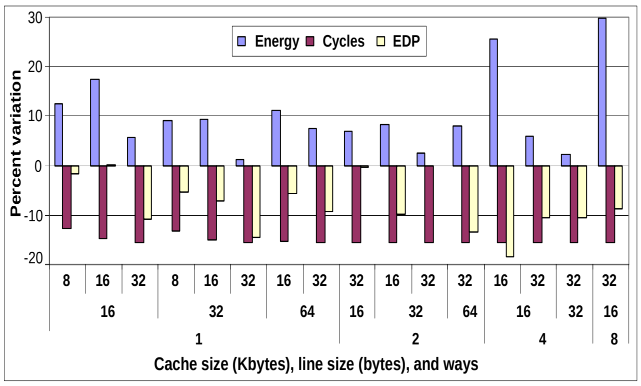
Fig. 13. Percentage variation in energy, cycles, and EDP for 4-issue core compared to 2-issue core with different I-caches for the Dijkstra application.

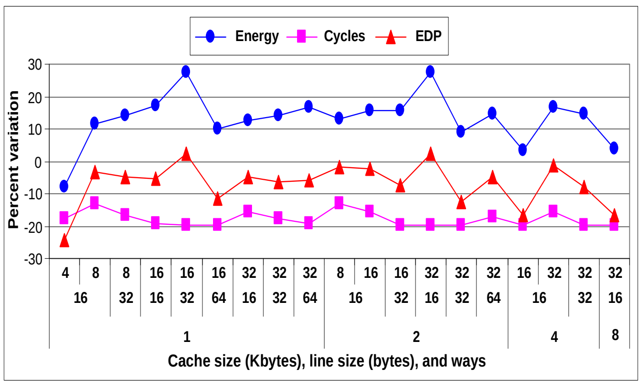
Fig. 15. Percentage variation in energy, cycles, and EDP for 4-issue core compared to 2-issue core with different I-caches for the GSM encode.

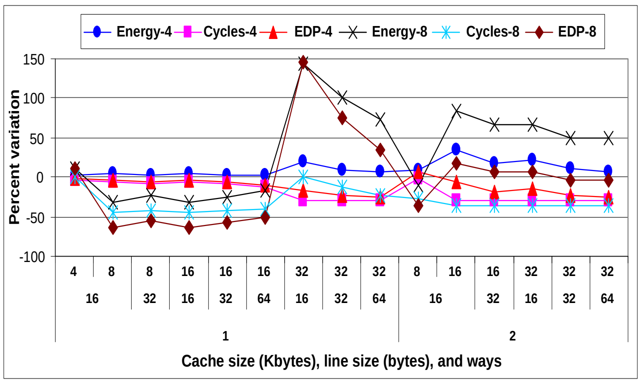

Fig. 16. Percentage variation in energy, cycles, and EDP for 4-issue and 8-issue cores compared to 2-issue core with different I-caches for the Rijndael encode.

vary from -30% to -2%, the energy consumption varies from 2% to 34%, and the EDP varies from -26% to 7%. When the issue-width is increased from 2-issue to 8-issue, the execution cycles vary from -45% to 0.4%, the energy consumption varies from -32% to 144%, and the EDP varies from -63% to 145%. Hence, when the issue-width is included with the I-cache reconfiguration, the Pareto points for the energy consumption, execution cycles, and the EDP are increased and a better configuration of the system can be found.

VI. Conclusion

In this paper, we presented an analysis for simultaneous reconfiguration of issue-width and instruction cache for a VLIW processor. The issue-width of the processor can be adjusted at run-time to be 2-issue, 4-issue or 8-issue and the I-cache can be reconfigured in terms of associativity, cache size, and line size. With different applications, we showed that compared to a “2-issue + the best I-cache”, a “4-issue + the best I-cache” configuration reduces the execution cycles, energy consumption, and EDP by up to 37%, 11%, and 41%, respectively, for different applications. Similarly, compared to a “2-issue + the best I-cache”, an “8-issue + the best I-cache” configuration reduces the execution cycles and EDP by up to 46% and 30%, respectively, for different applications.

References

[1] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, and R.B. Brown, “MiBench: A Free, Commercially Representative Embedded Benchmark Suite”, in *International Workshop on Workload Characterization*, pp. 3–14, 2001.

[2] A. Malik, B. Moyer, and D. Cermak, “A Low Power Unified Cache Architecture Providing Power and Performance Flexibility”, in *International Symposium on Low Power Electronics and Design*, pp. 241–243, 2000.

[3] R. Balasubramonian, D.H. Albonesi, A. Buyuktosunoglu, and S. Dwarkadas, “Memory Hierarchy Reconfiguration for Energy and Performance in General-purpose Processor Architectures”, in *International Symposium on Microarchitecture*, pp. 245–257, 2000.

[4] J. Yang, R. Gupta, and J.F. Martinez, “Energy Efficient Frequent Value Data Cache Design”, in *International Symposium on Microarchitecture*, pp. 197–207, 2002.

[5] D.H. Albonesi, “Selective Cache Ways: On Demand Cache Resource Allocation”, in *International Symposium on Microarchitecture*, pp. 248–259, 1999.

[6] S. Segars, “Low Power Design Techniques for Microprocessors”, in *International Solid-State Circuits Conference - Tutorial*, 2001.

[7] C. Zhang, F. Vahid, and W. Najjar, “A Highly Configurable Cache Architecture for Embedded Systems ”, in *International Symposium on Computer Architecture*, pp. 136–146, 2003.

[8] A.V. Veidenbaum, W. Tang, R. Gupta, A. Nicolau, and X. Ji, “Adapting Cache Line Size to Application Behavior”, in *International Conference on Supercomputing*, pp. 145–154, 1999.

[9] C. Zhang, F. Vahid, and W. Najjar, “Energy Benefits of a Configurable Line Size Cache for Embedded Systems”, in *International Symposium on VLSI*, pp. 87–91, 2003.

[10] K. Inoue, T. Ishihara, and K. Murakami, “Way-Predictive Set-Associative Cache for High Performance and Low Energy Consumption”, in *International Symposium On Low Power Electronics and Design*, pp. 273–275, 1999.

[11] P. Ranganathan, S. Adve, and N.P. Jouppi, “Reconfigurable Caches and Their Application to Media Processing”, in *International Symposium on Computer Architecture*, pp. 214–224, 2000.

[12] Hewlett-Packard Laboratories. VEX Toolchain. [Online]. Available: http://www.hpl.hp.com/downloads/vex/.

[13] CACTI: An Integrated Cache and Memory Access Time, Cycle Time, Area, Leakage, and Dynamic Power Model. *http://www.hpl.hp.com/research/cacti/*.

[14] D. Benitez, J.C. Moure, D. Rexachs, and E. Luque, “A Reconfigurable Cache Memory with Heterogeneous Banks”, in *Design, Automation & Test in Europe Conf. & Exhibition* , pp. 825–830, 2010.

[15] “TMS320C6211 Cache Analysis”, *Texas Instruments Application Report SPRA472*, 1998.

[16] T. Givargis and F. Vahid, “Tuning of Cache Ways and Voltage for Low-Energy Embedded System Platforms”, *Journal of Design Automation for Embedded Systems*, vol. 7, no. 1–2, pp. 35–51, 2002.

[17] M. Powell, A. Agarwal, T.N. Vijaykumar, B. Falsafi, and K. Roy, “Reducing Set-associative Cache Energy via Way-prediction and Selective Direct-mapping”, in *International Symposium on Microarchitecture*, pp. 54–65, 2001.

[18] K.T. Sundararajan, T.M. Jones, and N. Topham, “A Reconfigurable Cache Architecture for Energy Efficiency”, in *International Conference on Computing Frontiers*, 2011.

[19] S.H. Yang, M.D. Powell, B. Falsafi, and T.N. Vijaykumar, “Exploiting Choice in Resizable Cache Design to Optimize Deep-Submicron Processor Energy-Delay”, in *International Conference on High-Performance Computer Architecture*, pp. 151–161, 2002.

[20] C. Zhang, F. Vahid, and R. Lysecky, “A Self-Tuning Cache Architecture for Embedded Systems”, *ACM Transactions on Embedded Computing Systems*, vol. 3, no. 2, pp. 407–425, 2004.

[21] N. Mohamed, N. Botros, and M. Alweh, “Cache Memory Energy Minimization in VLIW Processors”, *Journal of Communication and Computer*, vol. 6, no. 12, pp. 70–74, 2009.

[22] E. Ipek, M. Kirman, N. Kirman, and J.F. Martinez, “Core Fusion: Accommodating Software Diversity in Chip Multiprocessors”, *ACM SIGARCH Computer Architecture News*, vol. 35, issue 2, pp. 186–197, 2007.

[23] K. Sankaralingam, R. Nagarajan, H. Liu, C. Kim, J. Huh, D. Burger, S. W. Keckler, and C. R. Moore, “Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture”, in *International Symposium on Computer Architecture*, pp. 422–433, 2003.

[24] H. Zhong, S. A. Lieberman, and S. A. Mahlke, “Extending Multicore Architectures to Exploit Hybrid Parallelism in Single-thread Applications”, in *International Symposium on High Performance Computer Architecture*, pp. 25–36, 2007.

[25] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz, “Smart Memories: A Modular Reconfigurable Architecture”, in *International Symposium on Computer Architecture*, pp. 161–171, 2000.

[26] F. Anjam, M. Nadeem, and S. Wong, “Targeting Code Diversity with Run-time Adjustable Issue-slots in a Chip Multiprocessor”, in *Design, Automation & Test in Europe Conf. & Exhibition*, pp. 1358–1363, 2011.

[27] S. Wong and F. Anjam, “The Delft Reconfigurable VLIW Processor”, in *International Conference on Advanced Computing and Communications*, pp. 242–251, 2009.

[28] J.A. Fisher, P. Faraboschi, and C. Young, *Embedded Computing: A VLIW Approach to Architecture, Compilers and Tools*. Morgan Kaufmann, 2004.

[29] P. Faraboschi, G. Brown, J.A. Fisher, G. Desoli, and F. Homewood, “Lx: A Technology Platform for Customizable VLIW Embedded Processing”, in *International Symposium on Computer Architecture*, pp. 203–213, 2000.