Middle-East Journal of Scientific Research 22 (11): 1612-1617, 2014
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2014.22.11.21523

Performance Evaluation of VLIW and Superscalar Processors on DSP and Multimedia Workloads

Mutaz Al-Tarawneh

Computer Engineering Department, Faculty of Engineering,
Mu'tah University, P.O.Box (7), Mu'tah 61710, Jordan

Abstract: This paper provides a quantitative evaluation of the performance of Very Long Instruction Word (VLIW) and Superscalar (SS) processors on Digital Signal Processing (DSP) and Multimedia applications. The VLIW configuration has been evaluated using Trimaran, an integrated compilation and performance monitoring infrastructure, while the Superscalar architecture has been evaluated using Simplescalar, a suite that provides both detailed and high- performance simulation environment of modern microprocessors, on a set of DSP and Multimedia benchmarks. Our benchmark suite includes both kernels and applications. Performance monitoring results on Trimaran and Simplescalar have been used to compare the Instruction-Level Parallelism (ILP) achieved by running the chosen benchmarks on the candidate processor architectures. In comparison to superscalar processors, VLIW exhibits a speedup range of 1.18 to 2.44 on Kernels. On the other hand, Superscalar processors show a high performance on some control-intensive applications with a speedup range of 1.15 to 1.33 compared to VLIW processors. The benchmarks are seen to contain a large amount of parallelism. Out-of-order execution and branch prediction have been observed to be extremely important to exploit such parallelism in multimedia applications.

Key words: DSP · Multimedia · VLIW · Superscalar · Performance · ILP

INTRODUCTION

Digital signal processors have been widely used to realize real-time signal processing systems using hardware architectures and software instruction sets that are optimized for such applications. However, their lack of complier support has introduced several backward compatibility problems from programmer's perspective [1]. Moreover, general purpose microprocessors have grown in capability to the point they can serve as an alternative platforms for digital signal processing (DSP) and multimedia applications [2]. General purpose microprocessors employ a number of architecture/micro-architecture performance enhancement features such as compiler support, fast clock rates, cache memories, pipelining and instruction level parallelism (ILP). In order to achieve high performance in the DSP and Multimedia applications, the architecture of choice should be able to exploit the high level of parallelism available in these applications. VLIW and Superscalar processors are two candidates that can be used to run DSP and Multimedia applications because of their ability to execute multiple operations simultaneously [3, 4]. Superscalar processor implements a form of parallelism by executing more than one instruction simultaneously based on some hardware-based scheduling techniques. On the other hand, VLIW executes operations in parallel based on a fixed schedule determined during the compile time. Since determining the order in which instructions will be executed and which operations can execute simultaneously is handled by the compiler, the processor does not need the scheduling hardware that the superscalar requires. Therefore, relying on compiler-based instruction scheduling techniques leads to a simple hardware implementation of VLIW processors. This paper evaluates the performance of VLIW and superscalar processors on DSP and multimedia applications. Its contributions are twofold. First, it provides a quantitative evaluation of the speedup that VLIW and superscalar can achieve on DSP and multimedia applications. Second, it compares hardware-based and compiler-based performance enhancement features and Figure out which one is more beneficial for DSP and multimedia workloads.

Corresponding Author: Mutaz Al-Tarawneh, Computer Engineering Department, Faculty of Engineering, Mu'tah University, P.O.Box (7), Mu'tah 61710, Jordan. E-mail: mutaz.altarawneh@gmail.com.

Related Work: DSP and multimedia applications have already become a ubiquitous computing workload in almost all kinds of computer markets. Their usage spans a wide range of applications such as digital filtering, video and audio compression/decompression, radar signal processing, etc. moreover, they are usually employed in application domains where high performance is of great importance from user's perspective. Consequently, evaluating the performance potentials and figuring out the best hardware platforms for such applications has attracted designer's attention in the last few years. In [5], an investigation of the multimedia-aware enhancement of the general purpose processors has been presented. They have presented the enhancements made to both processor's architecture and instruction set architecture (ISA). In [6], an evaluation of vector, superscalar and VLIW architectures for embedded applications has been provided. On the other hand, an evaluation of signal processing and multimedia applications on SIMD, VLIW and superscalar architectures has been given in [7]. In [8], a customized VLIW processor design has been presented and its performance on multimedia applications has been evaluated. The energy exploitation of cache hierarchy in VLIW processors has been analyzed in [9]. This paper represents a performance-centric study that differs from the previous work in three main aspects. First, our work is based on a fully parameterizable simulation environment in which VLIW and superscalar processors can be unified as much as possible in terms of their cache hierarchy, number of functional units and their latencies. This parameterization leads to a reasonable performance evaluation and allows processor designers to get an accurate estimate of the impact of instruction scheduling techniques on the performance of the processor. Second, previous studies did not take into account the impact of out-of-order execution and branch prediction accuracy on the performance of superscalar processors. Third, they did not take into account the impact of instruction scheduling granularity on the performance of VLIW processors.

Processor Architecture: This part gives a high level description of the VLIW and superscalar processor models that have been used in this work. In order to be able compare the two different design paradigms, on DSP and multimedia applications, we have set their configurations such that all common hardware structures have the same characteristics. Therefore, any performance differences will be due to the instruction flow and scheduling techniques that uniquely characterize each processor design. Table 1 shows the unified configurations of the two processors. As shown in table 1, both of the processors employ a two-level cache hierarchy with a diversified instruction pipeline that is capable of handling 4 simultaneous instruction every clock cycle.

Table 1: Unified Processor Configurations.

Parameter	Specifications
Datapath	4 Integer ALU, 4 FP ALU, 2 Ld/St units, 1 Branch unit,4-issue
L1 I-Cache	32K, 1-way, 32 byte block, 1 cycle latency
L1 D-Cache	64K, 4-way, 32 byte block, 1 cycle latency
L2 cache	1M, 8-way, 64 byte block, 10 cycle latency

TOOLS: This section briefly describes the simulation tools that have been used to obtain the performance of VLIW and superscalar processors on our benchmarks.

Trimaran 3.7 [10]: The Trimaran system is an integrated compilation and performance monitoring infrastructure. It is comprised of the following components: a) a machine description facility (mdes) for describing ILP architectures. b) A parameterized ILP Architecture called HPL-PD. c) A compiler front-end (IMPACT) for C. It performs parsing, type checking, and a large suite of high-level (i.e. machine independent) optimizations. d) A compiler back-end (Elcor) parameterized by a machine description, performing instruction scheduling, register allocation, and machine-dependent optimizations. e) An extensible IR (intermediate program representation) which has both an internal and textual representation, with conversion routines between the two. The textual language is called Rebel. This IR supports modern compiler techniques by representing control flow, data and control dependence, and many other attributes. It is easy to use in its internal representation and its textual representation. f) A cycle-level simulator of the HPL-PD architecture which is configurable by a machine description and provides run-time information on execution time, branch frequencies, and resource utilization. g) An integrated Graphical User Interface (GUI) for configuring and running the Trimaran system. Included in the GUI are tools for the graphical visualization of the program intermediate representation and of the results. In this work, three main parts of Trimaran have been used as follows. First, Dinero has been used to change the cache memory configuration in order to unify the VLIW and Superscalar processor configurations. Second, the machine description facility (mdes) has been used to parameterize the target VLIW processor and match its configuration with the Superscalar processor. Third, the front-end and back-end compilers have been used to change the optimization

levels as well as scheduling algorithms to study their impact on the ILP.

Simplescalar 3.0 [11]: The Simplescalar tool set is a simulation infrastructure used to model and analyze the performance of superscalar processors. Using the Simplescalar tool, users can simulate real programs on a range of modern processors and systems. The Simplescalar tool set is a system software infrastructure used to build modeling applications for program performance analysis detailed micro-architectural modeling and hardware-software co-verification. Using the Simplescalar tools, users can build modeling applications that simulate real programs running on a range of modern processors and systems. The tool set includes sample simulators ranging from a fast functional simulator to a detailed, dynamically scheduled processor model that supports non-blocking caches, speculative execution and state-of-the-art branch prediction. During this project we have used sim-outorder simulator in order to simulate the target superscalar processor; as this simulator supports the out of order issue and execution. Simplescalar provides the ability to change the configuration of the target superscalar processor e.g., number of functional units, cache memory configuration and the branch prediction type. In order to run our chosen benchmark programs on simplescalar, it was necessary to install the simplescalar cross complier so that we are able to compile those benchmarks and run it on our target architecture.

Becnhmraks: For all benchmarks, the same source code has been compiled on the two processors and was executed using the same parameters and input files in order to avoid any input-dependent performance variations. Therefore, any performance differences between the simulated processor configurations are due to the inherence micro-architectural and scheduling techniques employed within each configuration.

MATERIALS AND METHODS

In this work, the Instruction Level Parallelism (ILP) has been used as the primary performance measure. For the VLIW processor, different instruction scheduling techniques have been used. Basic-Block (BB), Hyper-Block (HB) and Super-Block (SB) based scheduling algorithms have been evaluated in order to quantify the impact of increasing the instruction scheduling scope on processor's performance [15, 16, 17]. On the other hand, we have examined the performance of in-order and out-of-order superscalar execution models in order to measure the extent to which instruction execution order can impact processor's performance. Moreover, two different branch prediction techniques have been simulated in order to evaluate the effect of branch prediction accuracy and the branching behavior of the application itself on the performance achieved by superscalar processors.

RESULTS AND ANALYSIS

This section illustrates the performance of VLIW and superscalar processors with the configurations shown in Table 1 on the DSP/Multimedia benchmarks shown in Table 2. Table 3 shows the simulation results for the VLIW processor with Basic-Block, Hyper-Block and Super-Block scheduling and the Superscalar processor with in-order and out-of-order execution. The analysis of these results has gone through three main steps as shown below.

Step 1: The baseline of our analysis was to compare the VLIW paradigm (with Basic-Block scheduling algorithm) against an out-of-order superscalar processor as shown in Fig. 1.

Table 2: Benchmarks Description.

Benchmark	Category	Description
Cjpeg	Application	JPEG image compression
Djpeg	Application	JPEG image decompression
G721decode	Application	G.721 decoder
G721encode	Application	G.721 encoder
rawcaudio	Application	ADPCM encoder
rawdaudio	Application	ADPCM decoder.
fir	Kernel	Filtering benchmark
Dot product	Kernel	Dot product benchmark
Matrix	Kernel	Matrix multiplication
Complex multiply	Kernel	Filter benchmark
convolution	Kernel	Filter benchmark

Table 3: Simulation Results in terms of ILP.

	VLIW			SS	
	BB	HB	SB	I-O	O.o.O
Cjpeg	1.39	1.6	1.58	0.77	1.84
Djpeg	1.5	1.2	1.23	0.74	1.94
G721decode	1.25	1.75	2.04	0.91	1.99
G721encode	1.25	1.72	2.06	0.89	1.95
rawcaudio	1.16	2.36	1.95	0.60	1.50
rawdaudio	1.25	1.81	2.13	0.63	1.68
fir	1.59	1.59	1.59	0.41	0.65
Dot product	0.89	0.91	0.91	0.40	0.62
Matrix	1.57	1.89	1.89	0.65	1.33
Complex multiply	0.84	0.75	0.75	0.40	0.62
convolution	1.55	1.56	1.56	0.41	0.64

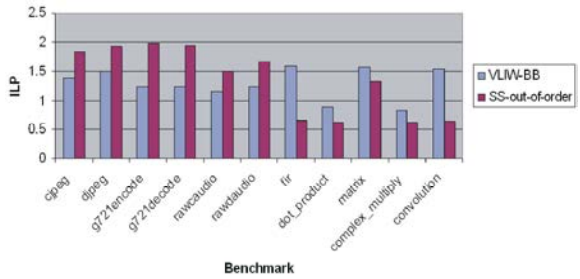

Fig. 1: VLIW Basic-Block vs. Superscalar Out-of-Order.

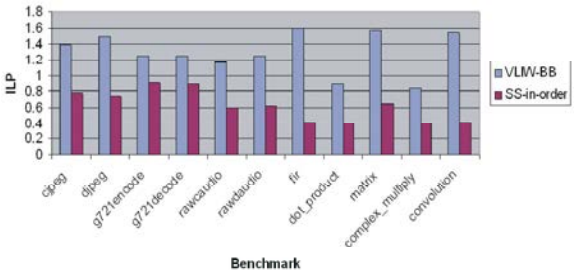

Fig. 2: VLIW-Basic-Block vs. Superscalar In-Order.

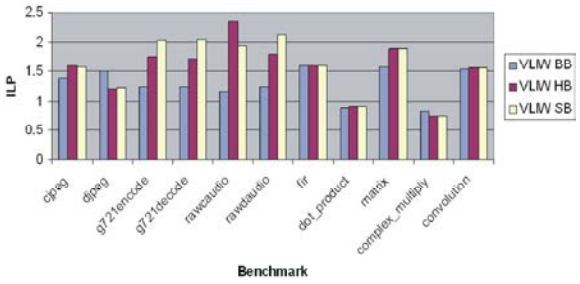

Fig. 3: Performance of VLIW with Different Scheduling Algorithms.

Based on Fig. 1, it can be observed that in the application benchmarks the Superscalar processor with out-of-order execution achieves a higher performance while in the kernel benchmarks the VLIW has better performance. In order to figure out the reason why the superscalar processor outperforms the VLIW processor in the application benchmarks, the same test has been repeated but using a superscalar processor with in-order execution as shown in Fig. 2.

Fig. 2 indicates that changing the execution model of the superscalar processor from out-of-order to in-order has resulted in a significant performance drop in the application benchmarks. The percentage of performance degradation ranges between 54% and 62%. In addition, there was a performance drop in the kernel benchmarks between 35% and 51%. This observation shows that out-of-order execution has a great impact on superscalar processor performance.

Table 4: Branch prediction accuracy.

Benchmark	No. of Branches	Misprediction Rate
Cjpeg	3484044	4.88%
Djpeg	352232	5.67%
Rawcaudio	2138236	26.22%
Rawdaudio	1770826	16.21%
g721encode	63072090	8.80%
g721decode	58985168	9.26%

Step 2: This step aims at comparing the performance of VLIW processor using three different instruction heduling algorithms (Basic-Block, Hyper-Block and Super-Block). Simulation results are shown in Fig. 3.

Fig. 3 shows that for the majority of the benchmarks it is possible to enhance the ILP level by using Hyper-Block or Super-Block scheduling algorithms. Using Hyper-block or super-Block scheduling extends the instruction scheduling granularity beyond a single Basic-Block giving the compiler the chance to find more instructions that can be executed in parallel. However, this does not hold for all applications since some benchmarks perform better using Basic-Block scheduling. In other words, the performance enhancement that can be achieved by any of these scheduling algorithms is application-dependent; there is no guarantee that a given scheduling algorithm will be better than the others for all application types.

Step 3: This step compares the VLIW processor (with Hyper-Block or Super-Block scheduling algorithms) against an Out-of-Order Superscalar processor as shown in Fig. 4. The main goal of this step is to compare the best performance achieved by the VLIW processor with the best performance an Out-of-Order Superscalar processor can attain. Fig. 4 shows an alternating behavior in which the Superscalar processor outperforms the VLIW processor on some benchmarks while the latter achieves a higher performance on another set of benchmarks. In order to explain this behavior, we have studied the impact of branch prediction accuracy on superscalar performance. Branch prediction is a well-known instruction flow technique that allows the superscalar processor to achieve high ILP levels by extending its scheduling granularity to span multiple Basic-Blocks. The role of branch prediction in superscalar processors resembles the role of Hyper-Block Super-block scheduling in VLIW processor. Table 4 gives the number of branches and the branch misprediction rate for each of the application benchmarks. Kernel benchmarks are a small code segments and are usually invoked as subroutines with application benchmarks. In general, Multimedia

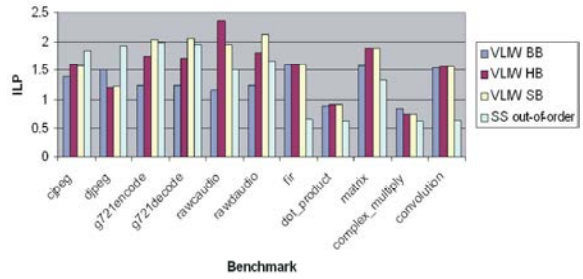

Fig. 4: VLIW vs. Superscalar Performance Comparison.

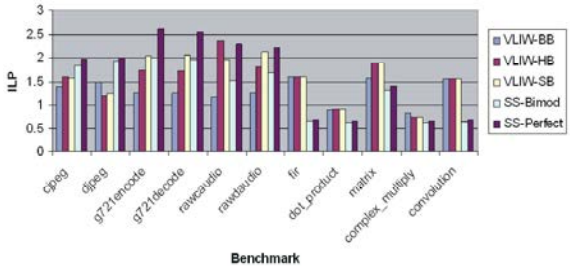

Fig. 5: The Effect of Branch Prediction Accuracy on Superscalar Performance.

applications have low branch predictability due to data-dependent branch instructions in this kind of workloads. Therefore an accurate branch predictor should be used in order to capture the runtime behavior of such kind of branch instructions [18].

By analyzing the results shown in Fig. 4 and Table 4, a threefold observations can be made. First, for control- intensive applications with high branch prediction accuracy (such as cjpeg and djpeg) (Table 4), the superscalar performance is better than VLIW performance. Second, for control-intensive applications with low branch prediction accuracy (such as rawcaudio and rawdaudio) (Table 4), the superscalar performance is worse than the VLIW performance. Third, for kernel benchmarks, the VLIW has better performance than that of superscalar. This is due to the fact that VLIW compiler-based scheduling gives more advantages over the dynamic scheduling provided by the superscalar processors. In order to measure extent to which branch prediction accuracy can influence the performance of the superscalar processor, we have repeated the previous step using out-of-order superscalar processor with Prefect branch prediction instead of the Bimod branch prediction used in the previous steps. A perfect branch prediction is a hypothetical branch predictor whose accuracy is 100%. Fig. 5 depicts a comparison between the VLIW processor with Hyper-Block / Super-Block scheduling and the superscalar processor with perfect branch prediction. Based on Fig. 5, it can be noticed that branch prediction accuracy has a great impact on the performance achieved by superscalar processors in the application benchmarks; superscalar processor with perfect branch prediction outperform the best VLIW configuration on all benchmarks. On the other hand, as compared to a superscalar processor with even a perfect branch prediction; VLIW is still more capable of exploiting the ILP available in Kernel benchmarks.

CONCLUSION

In this paper, the performance of general-purpose VLIW and Superscalar processors, on DSP and Multimedia applications, has been evaluated and compared. Several interesting observations have been made; First, Instruction level parallelism available in DSP application can be exploited by software techniques as in VLIW or by hardware techniques as in superscalar. Second, VLIW processor achieves higher performance on kernels and outperforms the superscalar even with perfect prediction. Third, branch prediction and out-of-order execution in superscalar processors play a major role in enhancing the performance and achieving high ILP levels. Fourth, application structure affects the performance enhancements that can be attained by VLIW or superscalar processors, e.g. VLIW performs better on applications with high loop inter-iteration parallelism while superscalar performs better on control-intensive with high branch predictability.

REFERENCE

1. Berkeley Design Technology, Inc. (BDTI), http://www.bdti.com.
2. Barkdull, J.N. and S.C. Douglas, 1996. General-purpose microprocessor performance for DSP applications. In the Proceedings of the Thirtieth Asilomar Conference on Signals, Systems and Computers, pp: 912-916.
3. Patterson, D.A. and J. L. Hennessy, 2006. Computer Architecture: a Quantitative Approach. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
4. Shen, J.P. and M.H. Lipasti, 2005. Modern Processor Design: Fundamentals of Superscalar Processors. McGraw-Hill.
5. Assaf, M. and A. Rajesh, 2007. General Architecture and Instruction Set Enhancements for Multimedia Applications. Journal of Systemics, Cybernetics and Informatics, 5(6): 64-72.

6. Kozyrakis, C. and D. Patterson, 2002. Vector vs. Superscalar vs. VLIW Architectures for Embedded Multimedia Benchmarks. In the Proceedings of the 35th International Symposium on Microarchitecture, pp: 283-293.
7. Talla, D., L.K. John, V. Lapinskii and B.L. Evans, 2000. Evaluating Signal Processing and Multimedia Applications on SIMD, VLIW and Superscalar Architectures. In the Proceedings of the International Conference on Computer Design, pp: 163-172.
8. Behera, R.K., D.K. Kumar and K.S. Pandey, 2013. Concept, Design and Performance Evaluation of VLVIW Processor. International Journal of Scientific Engineering and Technology, 2(7): 719-723.
9. Mohamed, N., N. Botros and M. Alweh, 2010. Cache Memory Energy Exploitation in VLIW architectures, Advanced Techniques in Computing Sciences and Software Engineering, pp: 351-355.
10. Chakrapani, L.N., J. Gyllenhaal, W.W. Hwu, S.A. Mahlke, K.V. Palem and Rodric M. Rabbah, 2005. Trimaran: An Infrastructure for Research in Instruction-Level Parallelism. Languages and Compilers for High Performance Computing, Lecture Notes in Computer Science, 3602(1): 32-41.
11. Bourger, D. and T. Austin, 1997. The Simplescalar Toolset Version 2.0. Computer Architecture News, pp: 13-24.
12. Zivojnovic, V., J. Martinez, C. Schlager and H. Meyr, 1994. Dspstone a DSP-oriented Benchmarking Methodology. In the Proceedings of International Conference on Signal Processing: Applications and Technology' pp: 94.
13. Lee, C., M. Potkonjak and W. H. Mangione-Smith, 1997. MediaBench: A Tool for Evaluating and Synthesizing Multimedia and Communications Systems. In the Proceedings of the Thirtieth Annual International Symposium on Microarchitecture, pp: 330-335.
14. Bishop, B., T.P. Kelliher and M.J. Irwin, 1999. A detailed Analysis of Media Bench. In the IEEE Workshop on Signal Processing Systems, pp: 448-455.
15. Malik, A.M., J. Mclnnes and P.V. Peek, 2006. Optimal Basic Block Instruction Scheduling for Multiple-Issue Processors using Constraint Programming. In the Proceedings of the 18th IEEE International Conference on Tools and Artificial Intelligence, pp: 279-287.
16. Mahlke, S., D. Lin, W. Chen, R. Hank and R. Bringman, 1992. Effective Compiler Support for Predicated Execution Using the Hyperblock. In the Proceedings of the 25th Annual International Symposium on Microarchitecture, pp: 45-54.
17. Shobaki, G. and K. Wilken, 2004. Optimal Superblock Scheduling Using Enumeration, In the Proceedings of the 37th International Symposium on Microarchitecture, pp: 283-293.
18. Fritts, J., W.H. Wolf and B. Liu, 1999. Understanding Multimedia Application Characteristics for Designing Programmable Media Processors. In SPIE Proceedings, 3655(1): 2-13.