**Evaluation of Static and Dynamic Scheduling for Media Processors**

Jason Fritts[1] and Wayne Wolf[2]

[1]Dept. of Computer Science, Washington University, St. Louis, MO
[2]Dept. of Electrical Engineering, Princeton University, Princeton, NJ
jefritts@cs.wustl.edu, wolf@ee.princeton.edu

## Abstract

*This paper presents the results of an architecture style evaluation that compares the performance of static scheduling and dynamic scheduling for media processors. Existing programmable media processors have predominantly used statically-scheduled architectures. As future media processors progress to higher frequencies and higher degrees of parallelism, the dynamic aspects of processing become more pronounced and dynamic hardware support may be needed to achieve high performance. This paper explores many of the dynamic aspects of media processing by evaluating various fundamental architecture styles and the frequency effects on those architecture models. The results indicate that dynamic out-of-order scheduling methods enable significantly higher degrees of parallelism and are less susceptible to high frequency effects. Consequently, dynamic scheduling support may be necessary for maximizing performance in future media processors.*

## 1. Introduction

This paper compares the performance of static and dynamic scheduling for media processors. Existing programmable media processors have predominantly used statically-scheduled VLIW or DSP architectures. However, the choice between static and dynamic scheduling has heretofore been based upon the simpler hardware design of static architectures, which enable lower cost and power and shorter design time. With the continuing growth of the multimedia industry, the increasing multimedia market will make more advanced media processor architectures, with such features as out-of-order scheduling and dynamic memory speculation, commercially viable. Consequently, it becomes necessary to evaluate the effectiveness of advanced dynamic architectures on current and future multimedia applications. Since we expect future media processors will have significantly higher frequencies, this study compares the performance of static and dynamic scheduling, and examines the effects of increasing frequency on each scheduling model.

This paper continues in Section 2, discussing the future of media processing, and the potential need for dynamic hardware support. Section 3 examines related research. Sections 4 and 5 proceed to describe the evaluation environment and base architecture model for this study. Section 6 compares static and dynamic scheduling performance and the impact of various architecture parameters. Section 7 then examines the impact of increasing frequency on static and dynamic scheduling. Finally, Sections 8 summarizes our conclusions.

## 2. Future Media Processing

Currently, there are three primary methods of support for multimedia: application-specific hardware, general-purpose processors with multimedia extensions, and programmable media processors. We focus our studies on the third option, *programmable media processors*, as we expect processors designed specifically for multimedia will become the dominant method of support. While the other two options are both viable, they are better suited to specific needs or markets. Application-specific hardware is best suited to single-application embedded systems, since it provides excellent performance at low cost, but has very limited flexibility for supporting evolving and future applications. General-purpose processors with multimedia extensions (e.g. Intel's MMX and SSE) provide significant flexibility, and provide some specialized instructions and hardware for multimedia and streaming memory, but they are designed primarily for general-purpose applications.

*Avenues to High-Performance*

Media processors have proven themselves capable of meeting the processing demands and flexibility requirements for supporting a wide range of media applications. Unfortunately, existing media processors achieve only a fraction of their potential performance. High-performance processing can be achieved through a variety of mechanisms. Media processors are currently only using subword parallelism, instruction level parallelism (ILP), and specialized instructions and coprocessors (e.g. DCT, motion estimation) for achieving high performance. Two other avenues for significant performance gains include high frequency and improved ILP.

Subword parallelism and specialized hardware are obvious areas for efficient performance, as they have minimal area and power costs for realizing potentially significant performance gains. ILP provides more flexibility, returning performance gains when specialized hardware and subword parallelism fail to provide much benefit. ILP typically requires greater resources to achieve performance gains, so current media processors, such as TI's C6x [1] and TriMedia's TM-1300 [2], rely on the simpler hardware of VLIW processors to minimize the cost and power of ILP.

The two largely unexplored avenues for greater performance in media processing include higher frequency and improved ILP. Significant potential exists from both mechanisms. With respect to high frequencies, all current media processors operate at no more than 300 MHz, but general-purpose processor demonstrate that frequencies in excess of 1 GHz are possible. Likewise, studies on dynamic scheduling have shown that it can improve ILP performance by as much as 100% [3].

*Cost – The Diminishing Barrier*

The primary reason media processors do not yet support high frequency and dynamic scheduling is because of the associated costs. Increasing frequency and adding dynamic scheduling hardware both require increased power, area, and processor design time. Comparing the most advanced media processor, the TI C62x [1], and the current popular general-purpose processor, the Intel Pentium III [4], we see there are some significant differences between the two. The C62x is an 8-issue VLIW processor that has a maximum frequency of 300 MHz, and costs about $250 (in quantities of 1000+). It was designed with a .15µm VLSI process and consumes an average of 1-2 W of power. Conversely, the Intel Pentium III is an out-of-order superscalar processor that is available at frequencies in excess of 1 GHz. It has approximately half the issue capability, but consumes between 13 W (at 500 MHz with 256 KB L2 cache) and 26 W (at 1 GHz with 256 KB L2 cache). It uses a .15µm VLSI process, and typically has much lower cost (except at very high frequencies). While the difference between these two processors seems substantial, it should also be noted that the Pentium III uses significantly less power per MIPS (million instructions per second) than previous Intel processors.

Because media processors currently have a much smaller market than general-purpose processors, computer users are not currently willing to pay the extra cost for these performance gains. However, as these costs begin to decrease and the market for multimedia continues to grow, we can expect media processors with these features will begin to appear. TI's recently announced C64x [1] provides an indication that the market is already moving in that direction. The C64x is a media/embedded processor targeting frequencies up to 1.1 GHz. Initial chips will soon be available at 600-800 MHz.

## 3. Related Work

There has not been a significant amount of academic or commercial work in designing or evaluating dynamic scheduling for media processor architectures. There are a number of commercially available media processors, but they mainly use static architectures [1][2]. Many research-based media and video signal processors have also been proposed [5][6], but performance analysis on these architectures typically just evaluates kernels. Only a select few have examined media processors using full applications on static architectures [7][8], and only the latter evaluates multiple-issue processors. In that paper, they examined performance using various issue widths, cache sizes, and numbers of branch units on a static architecture. They found minimal ILP gains from more than 3-4 issue slots (as we similarly conclude in Section 6.3 below).

There have been a number of studies evaluating static and dynamic architectures for general-purpose processing, but most have focused on only one of the two architecture models. We are only aware of studies by the IMPACT group that directly compare static and dynamic architectures [3]. These studies compared three architecture models: static VLIW, in-order superscalar, and out-of-order superscalar. The VLIW and in-order superscalar were found to perform comparably, while out-of-order scheduling typically performed 50-60% better.

## 4. Evaluation Environment

Accurate comparison of statically- and dynamically-scheduled architectures for media processing requires a benchmark suite representative of the multimedia industry, and an aggressive compilation and simulation environment. This architecture style evaluation uses an augmented version of the MediaBench benchmark suite defined by Lee, et al. [7] at UCLA. This benchmark provides applications covering six major areas of media processing: video, graphics, audio, images, speech, and security. We also augmented MediaBench with additional video applications, MPEG-4 and H.263. MediaBench already contains MPEG-2, but H.263 and MPEG-4 are distinct in that H.263 targets very low bit-rate video, and MPEG-4 uses an object-based representation. These applications are believed to make MediaBench more representative of the current and future multimedia industry.

The compilation and simulation tools for this architecture style evaluation were provided by the IMPACT compiler, produced by Wen-mei Hwu's group at UIUC [9]. The IMPACT environment includes a trace-driven simulator and an ILP compiler. The simulator enables both statistical and cycle-accurate trace-driven simulation of a variety of parameterizable architecture models, including both VLIW and in-order superscalar datapaths. We recently expanded the simulator to also support an out-of-order superscalar datapath.

The IMPACT compiler supports many aggressive compiler optimizations including procedure inlining, loop unrolling, speculation, and predication. IMPACT organizes its optimizations into three levels: *Classical* – performs only traditional local optimizations; *Superscalar* – adds loop unrolling and speculation; and *Hyperblock* – adds predication (conditional execution). The dependence of the scheduling models on these optimizations is measured below.

## 5. Base Architecture Model

Systematic evaluation of different architectures requires a base processor model for performance comparisons. We defined an 8-issue base media processor targeting the frequency range from 500 MHz to 1 GHz, with instruction latencies modeled after the Alpha 21264, (see Table 1 below). The base processor model may use any of three datapath models: (a) VLIW, (b) in-order superscalar, and (c) out-of-order superscalar. The datapath models do not currently support subword parallelism. Support for subword parallelism is currently being added and will be reported with future results. Additional details on the datapath models are discussed in Section 6.1.

The memory hierarchy of the base processor provides separate L1 instruction and data caches, a 256 KB unified on-chip L2 cache, and an external memory bus that operates at 1/6 the processor frequency. The L1 instruction cache is a 16 KB direct-mapped cache with 256-byte lines and a 20 cycle miss latency. The L1 data cache is a 32 KB direct-mapped cache with 64-byte lines and a 15 cycle miss latency. It is non-blocking with an 8-entry miss buffer, and uses a write-allocate/write-back policy with an 8-entry write buffer. The L2 data cache is a 256 KB 4-way set associate cache with 64-byte lines and a 50 cycle miss latency. It is non-blocking with an 8-entry miss buffer, and uses a write-allocate/write-back policy with an 8-entry write buffer. Further details are available in [10].

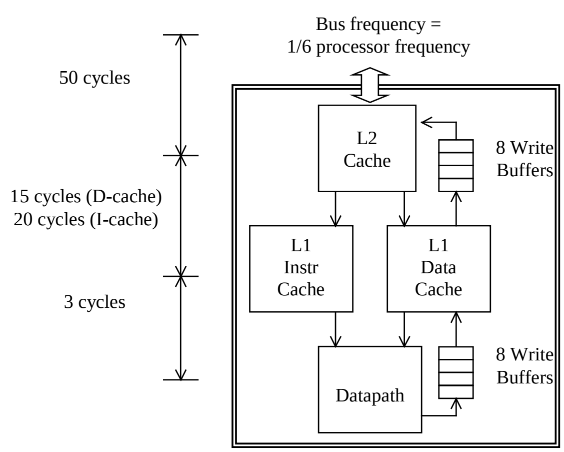


Figure 5.1 – Base processor model.

## 6. Architecture Style

This section evaluates three fundamental architecture types across the range of static and dynamic scheduling. Three experiments are performed that examine these architecture models and the effects of various compiler and architecture variations on them.

### 6.1. Basic architecture style

The first architecture experiment examines static and dynamic scheduling on the three basic architectures: out-of-order superscalar, in-order superscalar, and VLIW processor. The out-of-order superscalar processor enables full dynamic scheduling with out-of-order instruction issue using a 32-entry issue-reorder buffer. It provides memory disambiguation for dynamic data speculation, and allows early evaluation of branches. The in-order superscalar processor allows partial dynamic scheduling with register scoreboarding, which enables non-dependent instructions to continue issuing in-order during a memory stall. The VLIW architecture defines an entirely static architecture that issues long-instruction-words with up to 8 parallel operations[1]. Since the VLIW processor has no dynamic support, memory misses stall the entire pipeline. All architecture models use a 1024-entry 2-bit counter branch predictor.

Results comparing the average IPCs for the three architectures are shown in Figure

[1] VLIW instructions use a compressed instruction format.

6.1. These results indicate two important conclusions. First, static scheduling performs nearly as well as dynamic in-order scheduling for media processing. With average IPCs of 1.32 and 1.38 for the VLIW and in-order superscalar, respectively, there is only 5% difference in performance. We were expecting a much higher differential since dynamic in-order scheduling enables instructions to continue issuing on non-dependent memory stalls. Consequently, we believe the minimal difference in performance indicates good compiler efficiency. Because of its simpler hardware, between the two we, expect the VLIW architecture is the more power and cost effective solution for media processing (when binary compatibility is unnecessary).

Secondly, it is apparent that dynamic out-of-order scheduling, with an average IPC of 2.17, provides much better performance than static scheduling. The out-of-order superscalar processor enables 64% better performance on average than a VLIW processor over all MediaBench applications.

Detailed analysis of run-time performance between the out-of-order superscalar and both the VLIW and in-order superscalar processors revealed that there are two major advantages of out-of-order scheduling. As expected, one significant advantage is that out-of-order scheduling enables a significant degree of dynamic control speculation. Compile-time control speculation depends upon static branch/path prediction for making control speculation decisions. Static branch prediction is typically much less accurate than dynamic branch prediction (in media processing static branch prediction has about twice the branch misprediction rate), so control speculation at compile-time is less accurate than control speculation at run-time. Register scoreboarding as provided by the in-order superscalar only enables a small degree of dynamic control speculation. Out-of-order scheduling provides much larger degrees of control speculation in media processing.

The other major benefit of out-of-order scheduling is early branch evaluation. In an out-of-order superscalar processor, branch operations can be issued and executed without having to wait on earlier non-dependent memory stalls. Consequently, the processor is able to determine the exact path of execution much earlier than typically feasible in VLIW and in-order superscalar processors. Consequently, the processor can

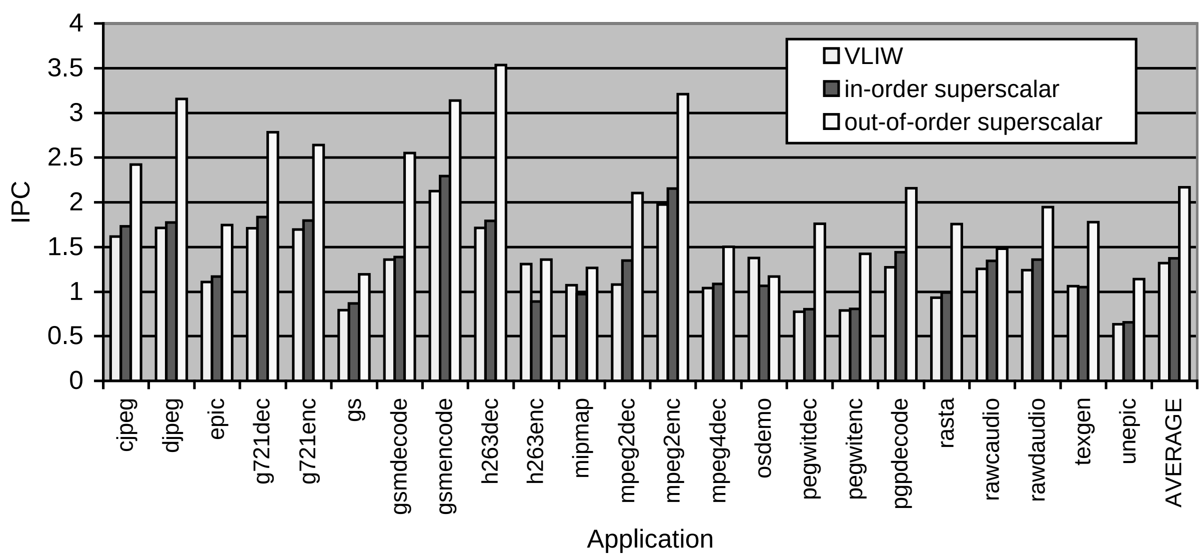


Figure 6.1 – Comparison of performance for MediaBench applications.

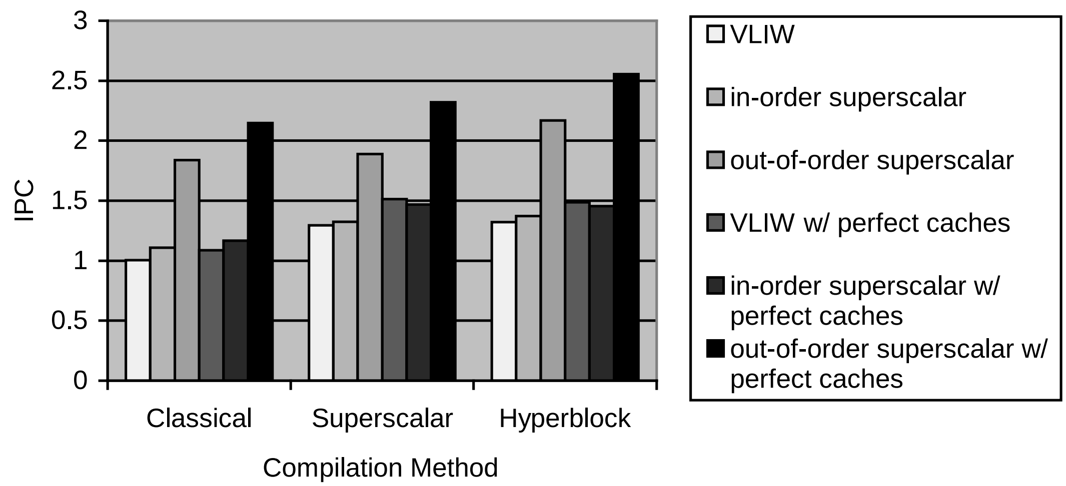


Figure 6.2 – Comparison of three processor models, simulated with and without perfect caches.

begin executing along the correct path, instead of speculating along the expected path.

In VLIW and in-order superscalar processors, branch operations must typically wait in the event of a memory stall. In VLIW architectures, static scheduling requires strict in-order issue, which precludes any early branch evaluation. In in-order superscalar processors, register score-boarding allows non-dependent instructions to execute in-order beyond memory stall. But since basic blocks often have the property that the result a load operation earlier in the block is required by one or more operations later in the block, the branch at the end of the block must typically wait on either a stalled load operation or on a stalled dependent instruction. Consequently, only out-of-order issue effectively enables early branch evaluation.

Because of improved control speculation and early branch evaluation, as well as other factors such as dynamic data speculation, the out-of-order superscalar is able to perform significantly better than either the in-order superscalar processor or VLIW processor. Consequently, dynamic out-of-order scheduling is expected to be necessary for achieving maximum ILP in future media processors.

## 6.2. Compiler optimizations

Because the performance of static scheduling is primarily dependent upon the compiler, evaluation with different compiler methods is necessary to thoroughly gauge its effectiveness. Figure 6.2 displays the results of an experiment that compared the performance between the three compiler optimization levels introduced in Section 4. From the results we can see that the hyperblock is the most effective method, providing 6% better performance than the superscalar method, and 26% better performance than classical optimization. The hyperblock's effectiveness varies considerably, however, with 12% greater performance over superscalar optimization on the out-of-order superscalar processor, but only 2% for the other architectures. And because predication requires an additional source operand for specifying the predicate condition, it is questionable whether the minimal improvement justifies the cost of an extra register port or separate register file, and increased instruction size.

### 6.3. Issue-width

Because the previous experiments produced ILP results on average of less than 3 instructions per cycle (IPC), we also experimented with smaller processor issue widths, since even with the most aggressive processor and compilation methods. Keeping the same ratios of functional units (rounding up as necessary), we examined processor widths of 1, 2, 4, and 8 issue slots. As shown in Figure 6.3, the results provide two important results. First, performance levels off around 4 issue slots for all three architecture models. Based on this result, we expect the ideal issue width to be 3-4 issue slots for ILP in media processors compiled with optimizations similar to IMPACT's. Secondly, the out-of-order superscalar processor performs significantly better than the VLIW and in-order superscalar processor over all processor issue widths. Furthermore, the performance of the out-of-order superscalar is such that on average the 2-issue out-of-order superscalar outperforms both 8-issue VLIW and 8-issue in-order superscalar processors.

## 7. Frequency effects

As the frequency of a processor increases, fewer levels of logic are available in each clock cycle, and wire delay becomes a more dominant factor in logic evaluation time. Fewer logic levels creates deeper pipelines and longer instruction latencies, while extra wire delay increases communication delays between the processor and memory as well as between separate functional units in the datapath. To understand the degree to which increasing frequency affects IPC and various architecture and compiler methods, this section performs two experiments. The first experiment examines the performance of the three architecture models on various processor frequency models. The second experiment examines the impact of delayed bypassing as caused by extra inter-functional-unit communication costs at high frequencies.

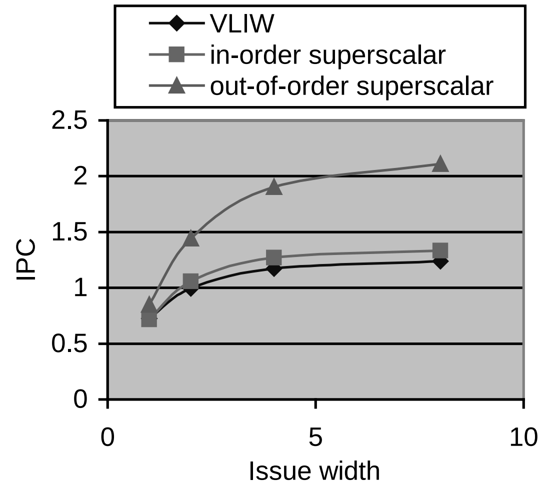


Figure 7.1 – Performance of various issue widths for VLIW and superscalar processors.

| *Instruction* | *Model 1* | *Model 2 (Base)* | *Model 3* |
|---|---|---|---|
| Frequency Range | 250-500 MHz | 500 MHz – 1 GHz | 1-2 GHz |
| Processor-Bus Freq. Ratio | 4:1 | 6:1 | 8:1 |
| ALU | 1 | 1 | 1 |
| Branches | 1 | 1 | 1 |
| Store | 1 | 2 | 3 |
| Load | 2 | 3 | 4 |
| Floating-Point | 3 | 4 | 5 |
| Multiply | 3 | 5 | 7 |
| Divide | 10 | 20 | 30 |

Table 1 - Instruction latencies for three models processor.

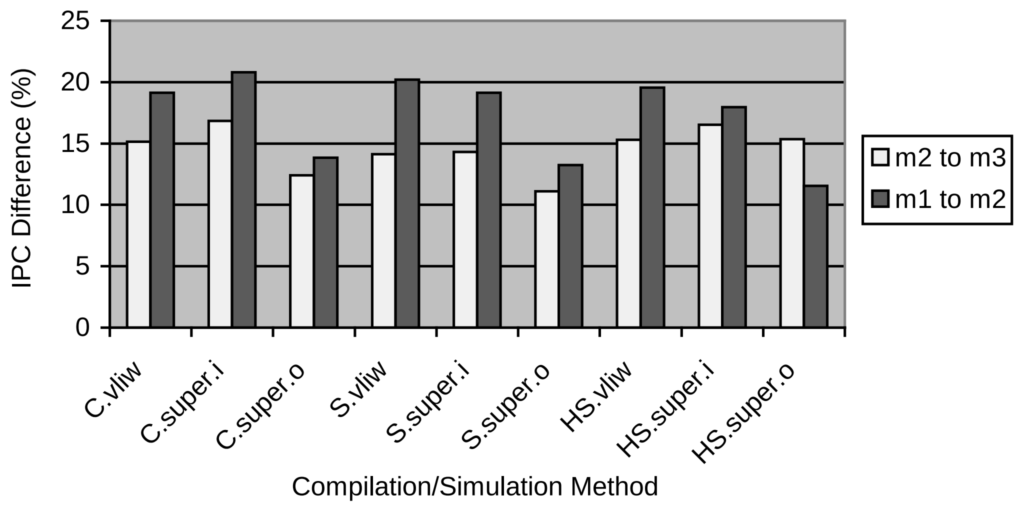


Figure 7.2 – Performance difference between three processor frequency models; *super.i* and *super.o* represent in-order and out-of-order superscalar processors, respectively.

## 7.1. Frequency Models

Increasing processor frequency leads to deeper pipelines, longer instruction latencies, and increased memory latency, all of which reduce IPC. From a scheduling point of view, longer instruction latencies increase the length of the critical path in each scheduling region. Because the same number of instructions is available per region, but the critical path is now longer, parallelism is reduced. Likewise, increased memory latency acts to reduce IPC by increasing the stall penalties from cache misses.

To examine the IPC degradation from increasing instruction and memory latencies, we evaluate three different processor models over the frequency range from 250 MHz to 2 GHz, each with the instruction and memory latencies scaled appropriately, as defined in Table 1. These processor models are designed such that the base processor model is processor model 2, while processor models 1 and 3 target processor frequencies of half and twice model 2, the base processor model, respectively. For accurate evaluation the code is re-compiled for each new processor model.

Figure 7.2 presents the difference in performance between models 1 and 2, and between models 2 and 3, for nine different combinations of compiler optimization levels and architecture types. The average reduction from doubling processor frequency (as defined by these models) degrades IPC by approximately 16%. The longer instruction latencies were found to account for 2/3 of this IPC reduction, while increased memory latency accounts for the remaining 1/3. The IPC degradation varies by architecture type and compiler optimization level, with out-of-order superscalar realizing the least degradation. Because scheduling is variable at run-time in out-of-order superscalar architectures, the schedule can be adjusted to lessen IPC degradation.

Among the compiler optimization levels, the superscalar method is the least influenced by longer instruction latencies. It employs aggressive speculation, which helps minimize the effects of longer instruction and memory latencies. The hyperblock optimization level also employs speculation, but compilation there is optimized for predication, not speculation.

## 7.2. Bypassing models

Another aspect of increasing wire delay is greater inter-functional-unit communication costs, which impacts the ability of the architecture to quickly bypass results to all other functional units. As opposed to waiting for a result to be written to the register file, bypassing makes the results immediately available to all other functional units. Dependent operations may immediately use the result, avoiding one or two cycles of extra delay in waiting for the result from the register file.

With increasing wire delay, the time to move a result between functional units increases, so bypassing requires more time. Bypassing therefore requires either lengthening cycle time to allow the necessary communication time, or assuming a one-cycle delay for delayed bypassing or no bypassing. An experiment was conducted to compare the performance between bypassing without any delay and bypassing with a one-cycle delay. As above, the code was re-compiled to accommodate the extra one-cycle delay in operation latencies.

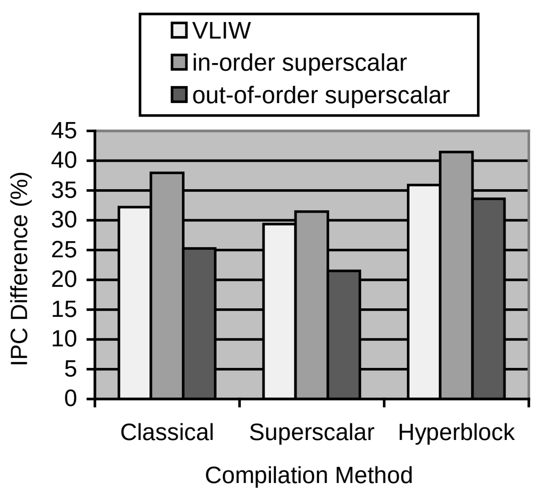


Figure 7.3 – Difference between immediate bypassing and bypassing with a one-cycle delay.

Figure 7.3 displays the difference in performance for immediate bypassing and delayed bypassing. The extra one-cycle delay in bypassing creates significant IPC degradation, with an average decrease of 32%. Like the previous experiment, the out-of-order superscalar processor and the superscalar compilation method are again the least susceptible to decreased performance. Out-of-order scheduling and speculation will be important architecture and compiler features for mitigating the effects of increasing frequency in media processors.

## 8. Conclusions

This paper examined the range of static and dynamic scheduling and explored the effects of increasing frequency on those architecture models in order to determine the most appropriate scheduling model for media processing. It was found that statically-scheduled VLIW architectures and in-order superscalar processors perform comparably, while dynamic out-of-order scheduling performs significantly better, with a 64% average improvement over VLIW performance. An evaluation of compiler techniques indicated that hyperblock optimization has the best performance, but its minimal performance improvement over superscalar optimization probably does not warrant the cost of additional hardware conditional execution. Finally, a variety of processor issue widths were evaluated for the three architecture models. Only minimal performance gain was found when using more than 4 issue slots. Furthermore, the 2-issue out-of-order superscalar demonstrated better average performance than both the 8-issue VLIW and 8-issue in-order superscalar.

This study also examined the impact of increasing frequency on all architecture models to determine the most suitable scheduling model for future high frequency media processors. The effect of increasing frequency was analyzed via two experiments, the first of which examined scheduling performance on three processor frequency models with varying instruction and memory latencies, while the second examined the performance variation between immediate and delayed bypassing. The experiments showed doubling processor frequency degrades parallelism by an average of 16% (for a total execution speedup of 72%), while adding a one-cycle delay to bypassing causes a significant IPC degradation of 32%. Among the various architecture and compiler optimization alternatives, dynamic out-of-order scheduling and superscalar optimization are about 30% less susceptible to IPC degradation than the other architectures and compiler methods.

Overall we have shown that some dynamic hardware support may be desirable in future media processors. Dynamic out-of-order scheduling demonstrated significantly better performance than other architectures, plus it is less susceptible to high frequency effects of longer instruction and memory latencies, and delayed bypassing. The cost of out-of-order scheduling and high frequency design in media processors is currently prohibitive, but as these costs diminish, we expect high frequency and out-of-order scheduling will be used to maximize performance in future media processors.

## Bibliography


[1] Texas Instruments C6x: http://www.ti.com/sc/docs/dsps/products/c6x/index.htm

[2] TriMedia processor: http://www.trimedia.philips.com/

[3] Pohua P. Chang, William Y. Chen, Scott A. Mahlke, and Wen-mei W. Hwu, “Comparing Static and Dynamic Code Scheduling for Multiple-Instruction-Issue Processors,” *Proceedings of the 24th International Symposium on Microarchitecture*, December 1991.

[4] Intel Pentium III: http://www.intel.com/

[5] P. Pirsch, H. J. Stolberg, Y. K. Chen, and S. Y. Kung, “On Implementation of Media Processors,” *IEEE Signal Processing Magazine*, vol. 14, no. 4, July 1997, pp. 48-51.

[6] Scott Rixner, William J. Dally, Ujval J. Kapasi, Bruce Khailany, Abelardo Lopez-Lagunas, Peter R. Mattson, and John D. Owens, "Media Processors Using Streams," *SPIE Photonics West: Media Processors '99*, San Jose, CA, January 28-29, 1999, pp. 122-134.

[7] Chunho Lee, Miodrag Potkonjak, and William H. Mangione-Smith, “MediaBench: A Tool for Evaluating and Synthesizing Multimedia Communications Systems,” *Proceedings of the 30th Annual International Symposium on Microarchitecture*, December 1997.

[8] C. Lee, J. Kin, M. Potkonjak, and W. H. Mangione-Smith, “Media Architecture: General Purpose vs. Multiple Application-Specific Programmable Processor,” *Proceedings of the 35th Design Automation Conference*, San Francisco, CA, June 1998.

[9] IMPACT compiler group: http://www.crhc.uiuc.edu/Impact

[10] J. Fritts, “Architecture and Compiler Design Issues in Programmable Media Processors,” Ph.D. Thesis, Dept. of Electrical Engr., Princeton Univ., 2000.