

## Multicycle Broadcast Bypass: Too Readily Overlooked

Peter G. Sassone     D. Scott Wills

Microelectronics Research Center  
School of Electrical and Computer Engineering  
Georgia Institute of Technology  
Atlanta, GA 30332, USA  
{sassone, scott.wills}@ece.gatech.edu

### Abstract

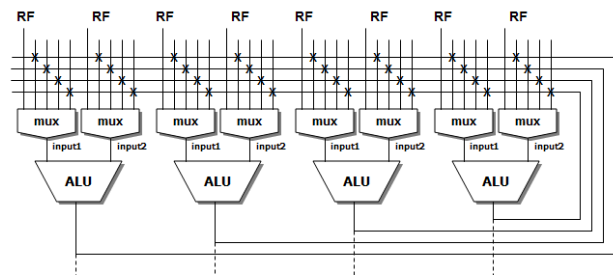
The bypass path, also called the forwarding path, allows processors to broadcast operands from one functional unit to another more quickly than through the register file. In modern superscalar out-of-order CPUs bypass is part of the execution pipeline stage, allowing dependant instructions to issue on subsequent cycles. In these modern machines, however, the bypass network complexity is becoming a limiting factor in frequency scaling. Traditionally, architects have been unwilling to separate the execute and bypass into different stages for fear of huge IPC losses. Through cycle-time calculations and cycle-accurate simulation with Spec2000int and Mediabench, though, we show that multicycle broadcast bypass is a simple and beneficial design choice. By allowing bypassed values multiple cycles to reach their destination, processor frequency can be increased more than IPC decreases. This solution involves no repeaters, no instruction steering, and no complex control logic. At 90nm, instruction throughput increases by 26% by separating the bypass into a separate stage on a four-wide machine, and throughput increases by 62% by using three bypass stages on an eight-wide machine.

### 1. Introduction

In a single-issue in-order processor, the addition of a forwarding path is likely a beneficial design choice. In exchange for control logic and backwards-directed data wires, a significant number of pipeline *bubbles* (no-ops) can be avoided, raising the instructions per second (IPC) of the pipeline. This is usually implemented as zero-cycle bypass; that is, the bypass delay is a part of the execution stage. This allows dependant instructions to execute on back-to-back cycles, removing all operand forwarding bubbles.

In a modern superscalar out-of-order processor, however, the complexity and delay of this enhancement is substantial. These factors grow geometrically as the pipeline is widened, reducing the practicality of large bypass networks. Figure 1, for instance, shows a diagram for a broadcast bypass on a four-wide machine. Long result buses and multiplexers dominate the design, producing complexity and delay. There have been many

academic and commercial endeavors to minimize the penalties for bypass, though most focus on removing its broadcast nature or enhancing its electrical characteristics. The basic suggestion of separating the arithmetic-logic unit (ALU) and bypass into separate stages is generally avoided however. Reinserting the bubbles that the forwarding path was intended to remove seems counterproductive and would decrease the IPC of the pipeline. However, our research shows that holding onto single-cycle bypass sacrifices too much cycle time for too little IPC, especially as the superscalar factor increases. Though optimizations such as electrical repeaters and non-broadcast bypass exist to alleviate the cycle time penalty, they come at the expense of power and complexity. Simple pipelining of the broadcast bypass network, though, is still faster than writing and reading to the register file and can produce higher execution rates than zero-cycle bypass.



**Figure 1. Bypass network on a four-wide superscalar processor. Long result buses and large multiplexers dominate the design.**

This paper is outlined as follows: in Section 2 we discuss related work in bypass delay estimation and alleviation. By combining these prior publications on bypass delays with other work on ALU delay and clocking overhead, Section 3 presents calculations for potential cycle times with various levels of bypass pipelining. Next, Section 4 gives the methodology and results for simulating IPC under each of these bypass conditions. Section 5 then combines the cycle time calculations with the IPC results to form instruction

throughput results. Finally, Section 6 concludes with caveats of these estimates.

## 2. Related Work

Interconnect scaling has been a well studied topic in architecture since submicron technologies emerged. Designers are often faced with sacrificing parallelism in order to decrease latency on these wires. Agerwal et al. point out in [1] that this is from an inherent conflict between instructions per cycle (IPC) and clock rate in modern processors. This conflict stands only to escalate since wires do not scale as effectively as logic [4], leaving designers with fast operand computation but slow operand transportation.

The traditional bypass network is a quintessential example of interconnect scaling complexity. Consisting of numerous buses and multiplexers (see Figure 1), this IPC enhancement becomes geometrically more complex as superscalar factor increases [2]. Palacharla et al., using Spice and a simple distributed RC model, derive formulas and delay estimates for bypass at various technology points [16]. Their results illustrate the geometrically increasing cost of this architectural feature, and suggest more effective use of it. Though repeaters can electrically accelerate these wires, their inclusion in interconnect design is non-trivial. Ho et al. state that repeaters can be quite large relative to the wires themselves, widening the buses' pitch and changing the floorplan significantly [12].

A common architectural method of alleviating bypass delay is clustering—dividing a processor's resources into logical groups. In this scheme, bypassing within a group is quick and efficient, but moving values between clusters incurs an additional delay. This heterogeneous bypass requires intelligent steering of instructions into clusters to minimize global communication. This technique is implemented commercially on the Alpha 21264 processor which has two identical pipelines with distinct register files and bypass networks [11]. More advanced implementations can be found in academia, such as Multicluster [8] and CTCP [3]. Parcerisa et al. study various techniques concluding that performance is very dependant on cluster interconnection and steering logic. They conclude that a point-to-point topology with latency-aware assignment is the most effective option [17].

Similarly, other work into explicit bypass removes the all-to-all broadcast nature of the network. Bunchua et al. in [5] propose local register files for each functional unit (FU) in out-of-order processors, a technique commonly reserved for VLIW machines. In this scheme, values are only bypassed between FUs on-demand, allowing free local bypass at the cost of longer inter-ALU delay. Transport-triggered architectures (TTAs) [7] are another mechanism for explicit data forwarding, exposing the bypass and reservation stations to the programmer for

more intelligent operand transport. Finally, grid-based processors such as TRIPS [15] allow forwarding only to nearest-neighbor FUs, moving the burden of bypass from physical layout to compiler analysis.

## 3. Cycle Time Estimation

To determine the overall impact of broadcast bypass on processor performance, we need both the instructions per cycle (IPC) and the cycle-time. To determine the latter, we make the assumption that execute is the longest pipeline stage and the rest of the chip can be pipelined at least this far. Though instruction scheduling, register file access, rename delay, and other effects also limit clock scaling, we feel that bypass will become dominant in future generations as bypass does not scale with technology [4]. Thus, by this assumption, the processor cycle time is only a function of the ALU delay, the bypass delay, and the latch overhead.

Palacharla et al. in [16] consider the delay of the bypass path as distributed RC lines modeled with Spice. Their equation governing the delay of a bypass network is given as:

$$T_{bypass} = \frac{1}{2} \cdot R_{metal} \cdot C_{metal} \cdot L^2 \quad (1)$$

Though the input capacitance of the multiplexers is not included in their model, they claim this component diminishes as feature size is reduced. The wires themselves do not scale, however, so they claim the estimate shown in below should hold at modern technology levels. The wire length,  $L$ , is a function of the number of functional units being bypassed while the resistance ( $R_{metal}$ ) and capacitance ( $C_{metal}$ ) per unit length remain constant.

Plugging in their parameter values and wire length estimations into this equation produces delays for various bypass widths, shown in Table 1. Using their assumption of non-scalability, we use this as the bypass delay at 180nm and 90nm. From these numbers, it is evident that the length term is dominant as the delay grows exponentially with more bypassed units.

**Table 1. Calculated bypass delays for various processor widths.**

Width	Delay (ps.)
2 wide	13
4 wide	185
6 wide	524
8 wide	1057

Next, to determine the delay of the ALU, we consider the 180nm Intel Itanium 1 and Itanium 2 microprocessors. It is claimed in [10] and [9] respectively that these CPUs spend half of their execute cycle on ALU execution and the other half traversing the six-way full bypass network. Thus using the model in (1), the delay of the Itanium ALU should be approximately equal to the six-wide

bypass delay of 524ps. To verify this estimate, this would make their cycle time twice that number, or 1048ps. This corresponds to a frequency of 954 MHz, which is close to the 800MHz frequency of the 180nm Itanium 1 and 1GHz of the 180nm Itanium 2 [14]. As logic scales well with technology, we make the assumption that the same ALU would have a delay of 300ps at 90nm technology.

Finally, to determine the clock overhead (latching, clock skew, and jitter) we use the determinations of Hrishikesh et al. [13]. In this work, they estimate the total overhead as approximately 125ps at 180nm and 66ps at 100nm. According to their assumption that these numbers scale with technology, we extrapolate an overhead of 60ps at 90nm.

With bypass delay, ALU delay, and clocking overhead delay, we can calculate cycle times based on these estimates. Figure 2 shows various bypass pipeline design choices which we will analyze. First is zero-cycle bypass, where the execute stage is comprised of the ALU execution and the bypass. This enables back-to-back issue of dependent instructions and should provide the highest IPC. Next is one-cycle bypass, where the bypass is a separate cycle from the ALU. This inserts a one cycle delay after each instruction's execution where the value cannot be read by another functional unit. The other two are two- and three-cycle bypass, which divides the bypass delay into two or three cycles respectively. Figure 2 is not to scale, however, as bypass delay is unlikely a multiple of ALU delay.

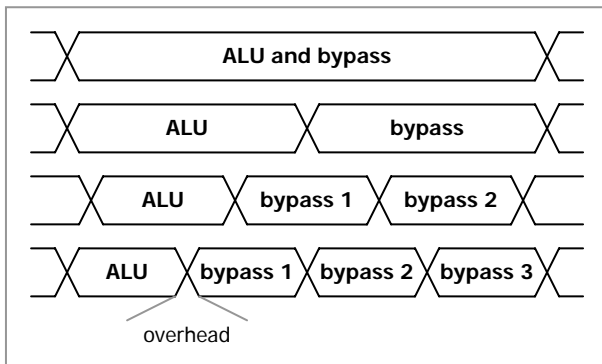


Figure 2. From top to bottom, clock diagrams for zero-, one-, two-, and three-cycle bypass.

For these designs, the pipeline latches could be inserted at various places in the bypass network, but evenly spaced between ALUs is a reasonable choice. Technically, this would allow for the value to arrive at some ALUs earlier than others. However, to not complicate the issue logic, each unit would assume the value takes the maximum number of cycles to arrive. Any early-arriving operands would simply be buffered at the inputs of the ALU. There are certainly opportunities for optimization here, but all come at the cost of additional

complexity within the issue logic, which is already a bottleneck [16].

The resultant cycle times for each bypass configuration are shown in Figure 3. For single-cycle bypass machines, the cycle time is the sum of the ALU delay, bypass delay, and clocking overhead. For one-cycle bypass, it is the maximum of ALU+overhead and bypass+overhead. For two-cycle bypass, it is the maximum of the ALU+overhead and half the bypass delay+overhead. Finally, for three-cycle bypass, it is the maximum of the ALU+bypass and a third of the bypass delay+overhead.

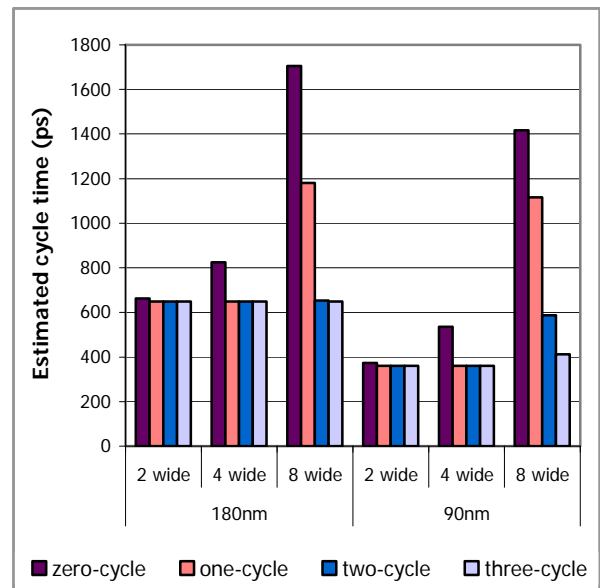


Figure 3. Estimated processor cycle times assuming execute is the critical stage.

#### 4. IPC Simulation

To determine the effect bypass delays have on application cycle count, we modified the SimpleScalar 3.0 cycle accurate simulator [6] to include bypass delays. All simulations were done with 32KB of L1 I-cache, 32KB of L1 D-cache, 256KB of L2, a 64-entry reorder buffer, and a 32-entry load-store queue. Superscalar width and bypass delay were the only parameters varied.

Table 2. Benchmarks used for IPC simulation.

Spec2000	Bzip Gcc Gzip Mcf	Parser Vortex Vpr
MediaBench	adpcm decode adpcm encode jpeg decode jpeg encode epic decode epic encode	g721 decode g721 encode mpeg2 decode mpeg2 encode pegwit decode pegwit encode

Table 2 shows the Spec2000int and MediaBench applications used for our simulations. Benchmarks from these suites which are not included did not compile cleanly using gcc 2.95.3 with O2 optimizations. For each run, we simulated 500 million instructions after skipping the first 100 million. Spec2000int inputs are taken from the *test* input set, and the default MediaBench inputs have been expanded to lengthen the benchmarks' execution.

Average IPC results for each combination of pipeline width and bypass pipelining are shown in Figure 4. As would be expected, IPC increases with wider pipelines and decreases as more bypass cycles are added. These rapidly diminishing IPC numbers are the reason that architects hold fast to single-cycle bypass.

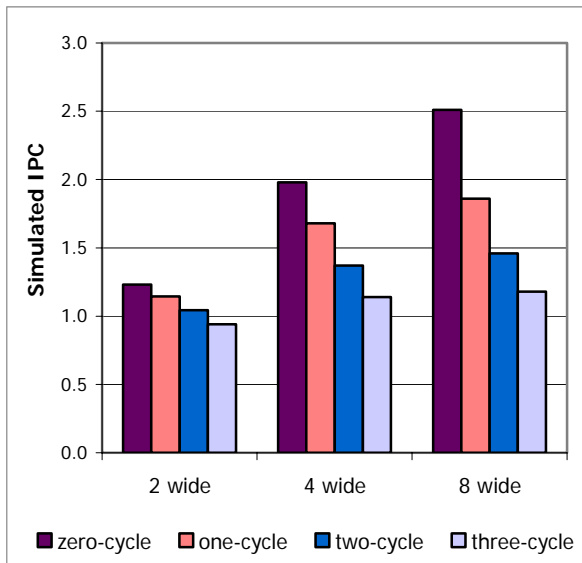


Figure 4. Average SimpleScalar IPC results across Spec2000int and MediaBench.

### 5. Instruction Throughput Results

Instruction throughput, measured in instructions per second, measures the total execution rate of a processor. This final calculation, a simple division of our simulated IPC by our calculated cycle time, is shown in Figure 5.

Though two-wide and four-wide pipelines perform predictably at 180nm, the figure shows an eight-wide machine has the highest instruction throughput at two cycles of bypass. This is surprising as architects hold firm to zero-cycle bypass despite increasing pipeline widths and decreasing feature size. It is also evident from Figure 5 that the eight-wide system is slower than the four-wide and even slower than the two-wide. Though it is intuitive that adding resources might make a processor less *efficient*, it is less so that the processor would become *slower*.

At 90nm, a modern wire-dominated technology point, these effects are more pronounced. The fastest four-

wide machine and optimal design point is with one-cycle bypass. A full three stages of bypass produces the highest execution rate on the eight-wide machine, though it cannot take advantage of logic scaling as the other pipelines do. The fastest eight-wide is now 60% as fast as the four-wide and 40% as fast as the two-wide. It is important to note that these curves are difficult to predict for arbitrary architectures where cycle-time and IPC curves will vary significantly. Each machine requires a separate analysis to determine the optimal number of bypass stages.

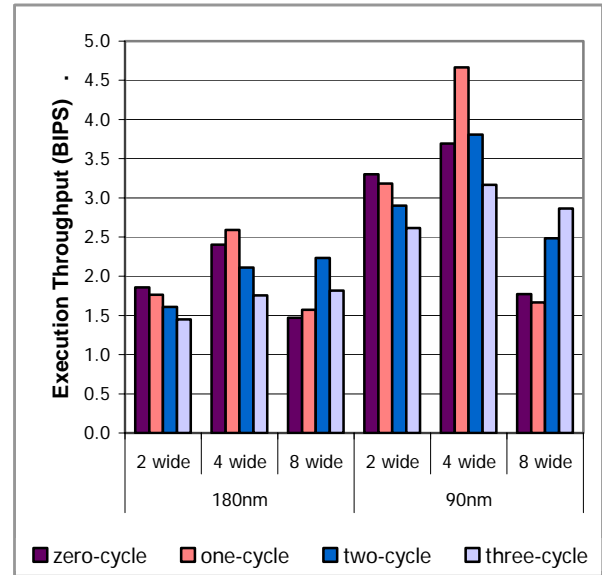


Figure 5. Instruction execution throughput, measured in billions of instructions per second.

### 6. Conclusion

For high-performance architectures, we propose that multicycle bypass is too readily overlooked. The cycle-time penalty for single-cycle bypass is growing on two dimensions in current designs. Bypass delay increases geometrically as more functional units are added, and relatively increases as ALU logic delay decreases with technology. These effects precipitate a loss in performance, not just efficiency, with a superscalar factor greater than four on traditional integer code.

It is imperative not to overlook the assumption that the ALU is the longest atomic delay in the pipeline. While this may not be accurate for all processor designs, the potential for speedup with multicycle bypass remains even at lower frequencies. A related assumption is that these cycle times are desirable—important issues such as heat and power may make such high frequencies impractical.

For most processor designs, however, the relative delay of bypass is growing rapidly with each technology shrink. As feature size shrinks below 90nm, bypass delays

will become even more evident. Architectural and electrical optimizations can slow the trends, but consume power, enlarge the floorplan significantly, or add to the intricacies of the issue logic. Once bypass has similar magnitude to that of the ALU delay, designers should consider the value of multicycle broadcast bypass. This complexity-effective design choice produces a beneficial cycle-time to IPC tradeoff without additional logic or power costs.

## References

- [1] V. Agerwal et al., "Clock Rate Versus IPC: The End of the Road for Conventional Microarchitectures," in *Proceedings of the 27<sup>th</sup> Annual International Symposium on Computer Architecture (ISCA-27)*, June 2000.
- [2] P. Ahuja et al., "The Performance Impact of Incomplete Bypassing in Processor Pipelines," in *Proceedings of the 28<sup>th</sup> International Symposium on Microarchitecture (MICRO-28)*, Nov. 1995.
- [3] R. Bhargava and L. John, "Improving Dynamic Cluster Assignment for Clustered Trace Cache Processors," in *Proceedings of the 30<sup>th</sup> International Symposium on Computer Architecture (ISCA-30)*, June 2003.
- [4] M. Bohr, "Interconnect Scaling – The Real Limiter to High Performance ULSI," in *1995 International Electron Devices Meeting Technical Digest*, pg 241-244, 1995.
- [5] S. Bunchua et al., "Reducing Operand Transport Complexity of Superscalar Processors Using Distributed Register Files," in *Proceedings of the 21<sup>st</sup> International Conference on Computer Design (ICCD-21)*, Oct. 2003.
- [6] D. Burger and T. Austin, "The SimpleScalar Tool Set, Version 2.0," Technical Report #1342, University of Wisconsin-Madison Computer Science, 1997.
- [7] H. Corporaal, "TTAs: Missing the ILP Complexity Wall," *Journal of Systems Architecture*, vol. 45, no. 12/13, June 1999.
- [8] K. Farkas, et al, "The Multicluster Architecture: Reducing Cycle Time through Partitioning," in *Proceedings of the 30th International Symposium on Microarchitecture (MICRO-30)*, Dec. 1997.
- [9] E. Fetzer, et al., "A Fully Bypassed 6-Issue Integer Datapath and Register File on an Itanium-2 Microprocessor," in *Proceedings of the International Solid State Circuits Conference (ISSCC)*, Vol. 37, Nov. 2002.
- [10] E. Fetzer, J. Orton, "A Fully Bypassed 6-Issue Integer Datapath and Register File on an Itanium Microprocessor," in *Proceedings of the International Solid State Circuits Conference (ISSCC)*, Vol. 1, Feb. 2002.
- [11] L. Gwennap, "Digital 21264 Sets New Standard," *Microprocessor Report*, pg 11-16, Oct. 28, 1996.
- [12] R. Ho et al., "The Future of Wires," *Proceedings of the IEEE*, vol. 89, no. 4, Apr. 2001.
- [13] M. Hrishikesh, et al., "The Optimal Logic Depth per Pipeline Stage is 6 to 8 FO4 Inverter Delays," in *Proceedings of the 29<sup>th</sup> International Symposium on Computer Architecture (ISCA-29)*, May 2002.
- [14] "Intel Itanium Processor Product Information," [www.intel.com](http://www.intel.com), last accessed: Mar. 2004.
- [15] R. Nagarajan et al., "A Design Space Evaluation of Grid Processor Architectures," in *Proceedings of the 34th International Symposium on Microarchitecture (MICRO-34)*, Dec. 2001.
- [16] S. Palacharla, et al., "Complexity-Effective Superscalar Processors," in *Proceedings of the 24th International Symposium on Computer Architecture (ISCA-24)*, 1997.
- [17] J. Parcerisa et al., "Efficient Interconnects for Clustered Microarchitectures," in *Proceedings of the International Conference on Parallel Architectures and Compilation Techniques (PACT)*, Sep. 2002.