The Performance Potential of Data Dependent Computation on Asynchronous Superscalar Processor

Suk-Jin Kim, Byung-Soo Choi, Chan-Ho Park and Dong-Ik Lee
Department of Information and Communications
Kwangju Institute of Science and Technology
1 Oryoung-dong, Puk-ku, Kwangju 500-712, Korea
Tel: +82-62-970-2248, Fax: +82-62-970-2204
E-mail: {sukjinkim, bschoi, chpark, dilee}@kjist.ac.kr

Abstract: We investigate potential advantages and problems when a superscalar processor is designed and implemented using asynchronous design methods. Conventional techniques of superscalar processing are applied and data dependent adder is considered as an asynchronous component. Intensive simulations on SPEC INT95 benchmark suites are made for the purpose of performance comparison between a synchronous and an asynchronous superscalar processor, respectively. The simulation results show about 5% speedup with asynchronous design methods in the sense of *Issue Rate*.

1. Introduction

Recently, a number of asynchronous microprocessors have been proposed [1]. Since there is no global clock in asynchronous systems, the clock skew problem is eliminated easily. Furthermore, the average case performance can be accomplished due to the data dependent feature of asynchronous systems. In the scalar architecture, however, we cannot fully enjoy the performance advantage of asynchronous systems since there may be situations called starvation or blocking by the slow blocks in pipeline [2]. Figure 1 shows how two functions are performed in a pipelined architecture with different design methods. The arrows represent the time consumed in each block from the instruction stream.

On the other hand, in a superscalar architecture this is not the case that the slow operations in one block can be overtaken by the faster operations in the other block because they have multiple functional blocks. See Figure 2. In addition, the dependencies among instructions can be resolved faster than a synchronous system, because it does not need to wait until the next clock is triggered and most operations have smaller computation time than the worst case delay. Therefore, a superscalar processor designed by asynchronous methods allows to operate as

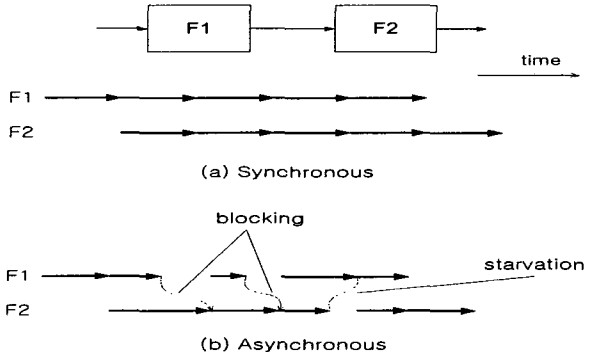

Figure 1. The operation of a scalar processor

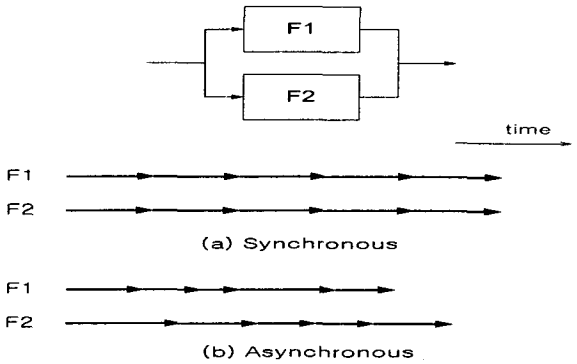

Figure 2. The operation of a superscalar processor

fast as possible, taking advantage of delay variations due to data dependencies. This leads to increase the utilization of subsystems and hence improve the performance in comparison with the synchronous counterpart. In this paper, at the first step of developing high performance asynchronous superscalar microprocessor, we aim at investigating the effectiveness of data dependent computation in superscalar architecture.

The rest of paper is organized as follows. Section 2 discusses data dependent computation in asynchronous systems and derives the statistics of carry chain length from a dynamic instruction trace. Section 3 describes modeled architecture and simulation method. Section 4 reports the results of our simulation. Finally, we draw the conclusion and expected problems of an asynchronous superscalar processor in Section 5.

2. Data Dependent Computation

Each stage of pipeline in synchronous systems has the same completion time since it is bounded by the worst

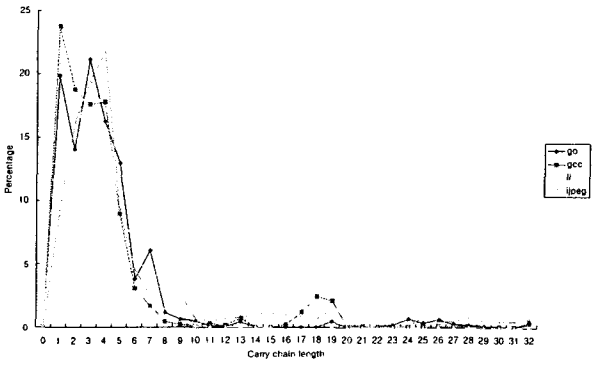

Figure 3. Carry chain length for arithmetic additions

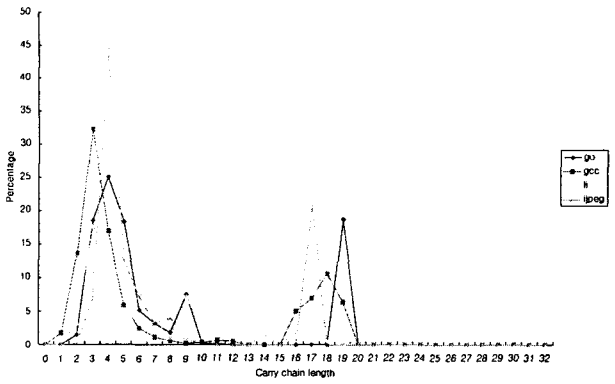

Figure 4. Carry chain length for address calculations

case delay in the slowest stage. In asynchronous systems, however, the time consumed in each stage is not constrained to a fixed cycle but may depend on both types of operations and their input data. Therefore, asynchronous systems show average case performance instead of the worst case performance.

To find out this feature in real processing, we investigate the 32-bit addition for SPEC INT 95 benchmark program. Figure 3 and 4 show the distribution of maximum carry propagation chain length for the arithmetic addition and memory address calculation of each benchmark program, respectively. The results indicate that the typical arithmetic addition has carry propagation chains approximately 5 bits long whereas those for address calculations are about 7 bits. For address calculation, some significant extra peak lies between 17 and 19 bits. The overall average has propagation chain approximately 6 bits long. This feature is very important to make asynchronous systems increase the performance.

3. Modeled architecture and simulation

The architecture we are considering models an out-of-order instruction issue mechanism and speculative execution to exploit the best instruction level parallelism. Furthermore, we assume neither branch prediction miss nor cache miss. Thus, only data and memory

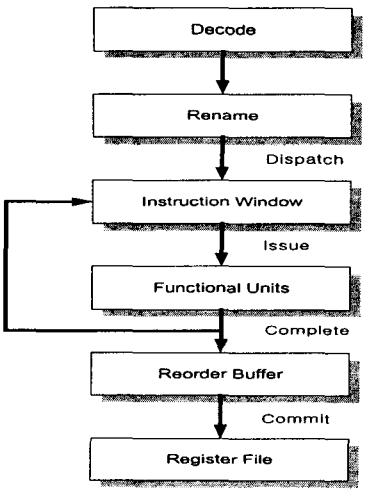

Figure 5. Block diagram of modeled architecture

Table 1. Details of the modeled architecture

Instruction Fetch	4 instruction fetch per cycle, no branch miss
Renaming	Mapping Table and Reorder buffer
Instruction Window (Wakeup, Selection)	20 entry central window (Tomasulo's algorithm[3], Oldest first mechanism)
Functional Units	2 ALUs, 1 shifter, 1 multiplier, 1 divider
FU latency	ALU 1, shifter 1, multiplier 3, divider 6, load/store 1, branch 1
Memory	32 entry load/store queue. Load can be from matching store ahead in queue. Store execute only after all the previous instruction have completed
Commit	256 entry Reorder Buffer, Infinite commit size

Dependencies are taken into account. Figure 5 shows the block diagram of modeled architecture and their details are listed in Table 1. The configuration and latencies of functional units are the same as those of SuperSPARC™ [4]. To investigate data dependent computation of asynchronous systems, we adopt a ripple carry adder and design using dual rail coding scheme[5], while a carry skip adder is used for a synchronous processor. The designed adders are synthesized by SYNOPSYS using 0.6μm IDEC C-631 library[6]. When input data has random distribution, the mean latency of the asynchronous adder is 0.6 times as long as that of the synchronous carry skip adder. All the configuration and latencies of processors are same except for the adder.

The cycle-base simulation based on the fixed computation time is difficult to evaluate the performance variation of data dependency in an asynchronous processor. New simulation method based on Parallel Virtual Machine(PVM)[7] is proposed. PVM consists of tasks and the message passing primitives; the communication mechanism between tasks in PVM is much similar to that between blocks in asynchronous systems. Thus, each stage in an asynchronous processor is mapped to a task of PVM and the simulation is undertaken as followings :

- receive the data and time information from the previous task,
- process his own operation,
- calculate computation time of current stage and add to the received time,
- send results and the time to the next task.

To ensure the correctness of proposed simulation method, it was applied to synchronous superscalar processor with fixed latencies of all blocks. Finally we had same results to the cycle-base simulation.

4. Simulation Result

SPEC 95 benchmark suits was used to produce instruction traces using SHADE[8] based on SPARC v.8 ISA. All benchmarks were removed in the initialization

Table 2. Simulation Results

Benchmark Programs	Sync. Superscalar [Issue Rate]	Async. Superscalar [Issue Rate]	Carry chain length [bit]	Latency of Async. Adder [Sync. Adder = 1]
go	1.728	1.790	4.04	0.522
gcc	2.172	2.265	4.21	0.534
lisp	2.238	2.361	4.12	0.527
ijpeg	1.949	2.021	8.62	0.994
H-mean	2.001	2.085	4.74	0.598

phase, then we executed about 52 million instructions. Table 2 shows the simulation results. We investigated issue rate without NOP instructions to evaluate the performance. Also, each benchmark was examined the longest carry chain in average and the latency for an asynchronous adder. The latency of asynchronous arithmetic addition is normalized by the latency of the synchronous counterpart. For all benchmark programs an asynchronous superscalar processor has better performance than synchronous one. In case of ijpeg, although the average latency is almost same as synchronous, the performance is better than synchronous. This is because instructions whose computational time is comparatively small lead to the succeeding dependent instructions issue quickly. The average performance gain is approximately 5%.

5. Conclusion

In this paper, we investigated the performance improvement of an asynchronous superscalar processor due to the data dependent computation. The simulation results have shown that an asynchronous superscalar processor yields about 5% higher performance than a synchronous one in the sense of issue rate. The improvement is caused by: 1) data dependent addition and hence 2) the fast result forwarding. Therefore, the more data dependent computation blocks are designed and implemented in an asynchronous superscalar processor, the better performance we can obtain. In addition, a new simulation method using PVM was investigated for an asynchronous processor. This simulation method is useful for the performance evaluation in the early phase of asynchronous system design.

However, some problems that influence the operation of an asynchronous superscalar processor have been identified. Tomasulo's algorithm known as a fast result forwarding mechanisms is hard to be implemented efficiently due to the difficulty of broadcasting. Furthermore, the micropipeline[9] style buffer is not suitable to implement the instruction window, because the position of data is unknown to the block after a block places data on a micropipeline stage. Also, the data in the micropipeline may be moving as a forwarding operation is performed. These issues are left for future research.

References

[1] T. Werner and V. Akella, "Asynchronous Processor Survey", IEEE Computer, pp. 67-76, vol. 30, 1997

[2] P. B. Endecott, "Superscalar instruction issue in an asynchronous microprocessor", IEE Proceedings on Computers and Digital Techniques, pp. 266-272, vol. 143, no. 5, 1996

[3] R. M. Tomasulo, "An Efficient Algorithm for Exploiting Multiple Arithmetic Units", IBM Journal, pp. 25-33, vol. 11, 1967

[4] G. Blanck and S. Kruger, "The SuperSPARC™ Microprocessor", IEEE Compcon Proceedings, 1992

[5] T. Nanya, Y. Ueno, H. Kagotani, M. Kuwako, and A. Takamura, "TITAC: Design of a Quasi-Delay-Insensitive Microprocessor", IEEE Design and Test of Computers, pp. 50-53, 1994

[6] IDEC Cell Library Data Book Release 9804, 1998

[7] A. Geist, A. Beguelin, J. Dongarram, W. Jiang, and R. Mancheck, "PVM: Parallel Virtual Machine A users' Guide and Tutorial for Networked Parallel Computing", The MIT Press, Cambridge, Massachusetts, 1994

[8] "Introduction to Shade", Sun Microsystem Laboratories, Inc. TR 415-960-1300, Revision A of 1/Apr/92

[9] Ivan Sutherland, "Mciropipelines", Communications of the ACM, pp. 720-738, vol. 6, no. 6, 1989