配線遅延を考慮したマルチスレッド方式アーキテクチャ SHIFT Machineの提案

中村 維男[†], 佐藤 幸紀[†]
東北大学大学院情報科学研究科[†]

半導体のスケーリングが進むにつれて、1チップで利用可能なトランジスタ数が10から100億個に達しようとしている。しかし、スケーリングが進むにつれて、トランジスタのゲート遅延よりも配線遅延がマイクロプロセッサの性能向上に対する支配的要因となる。本論文では、このような状況の中で優れたパラダイムとなるシフトアーキテクチャを提案する。このパラダイムは、チップ上でクロックパルスが到達可能な範囲のパイプライン化や、処理速度向上のための更なるパイプライン化を行う。パイプライン化された環境において、それぞれが資源依存の強い命令・データ流からなる大量のスレッドを処理する。このパラダイムは、将来のCMOS技術において非常に有望なアーキテクチャであるといえる。

SHIFT Machine: Multi-Threaded Architecture for Wire-Delayed Chip Environments

Tadao NAKAMURA[†] and Yukinori SATO[†]
[†]Graduate School of Information Sciences, Tohoku University

Abstract—This paper describes a novel paradigm of the SHIFT Architecture on one-ten billion transistor chips that have outstanding wire-delays compared with gate-delays as chip densities increase. The computing paradigm is consisting of welcome to wire-delays that induces pipelining among the logic islands (Pipeline stages) of clock pulses on the chip and further making use of pipelining to increasing the processing speed. We can process multiple threads each of which is one series of instructions to be processed through the pipelined chip environments. It can be said that this paradigm is capable of developing computer architecture in future CMOS technologies.

1 Introduction

In one to ten billion transistors era, the amount of gates is tremendously increased as chip densities increase [1] . From such an environment of hardware, a design concept of computers is changing year by year. It is high time to consider a novel computer architecture suitable for such environment allowing chips to include lots of transistors [4] . Logic islands problem on chips [3] occurs due to wire delays less improved than gate delays. Within the logic island on chips we can design any of gate circuits without thinking of wire delays. However, the island is interconnected in a pipeline fashion on microoperations.

During the above discussion, firstly we must progress in studying of a novel hardware architecture for one-ten billion transistors era. Fortunately we can make use of the fruitful environments of processing resources. If one logic island corresponds to one stage of the pipeline, wire problems can be cancelled. This emphasizes one of the features of pipelining, reasonable space-time uses under the simple control of parallel processing [2] .

On the other hand, software architecture is changed by the special hardware features because the software could be adaptable to changed hardware processing scheme. Also, we can interpret the processing scheme of any expression in its own way . Recent improvement of Compiler technique allows compiler to costomize application code to a target processor [5] .

This paper describes any of paradigms in hardware, software and program interpretation which is novel and useful for designing one-billion transistor chips including wire-delays. Then we firstly show a new interpretation of program to be processed in a novel hardware. This processing mode is very suitable for the processing scheme of the pipeline. We can get some efforts to show that this way is relatively reasonable to design the computer system.

2 Motivation of the SHIFT Architecture

We have a concept the signal drive distance that is a capability for a gate to access the line connected to the gate [3] . The distance is dependent on the line conditions within the gate delay. The optimal is the condition that a gate delay is equal to a wire delay to be propagated from the gate to a line, the property is originally given this. On the silicon chip, $0.6\mu m$ is corresponding to $5mm$. This means that a gate delay is depending on the gate electrical property and a line wire delay is depending on the electrical property of semiconductor and insulator. Fig. 1 shows signal drive distances succesively in CMOS technology generation. The longer the

- sdd 0.6 μm = 5 mm
- sdd 0.35 μm = 5 mm * (0.35/0.6) * $(1.2/1.5)^1$
- sdd 0.25 μm = 5 mm * (0.25/0.6) * $(1.2/1.5)^2$
- sdd 0.18 μm = 5 mm * (0.18/0.6) * $(1.2/1.5)^3$
- sdd 0.13 μm = 5 mm * (0.13/0.6) * $(1.2/1.5)^4$
- sdd 0.1 μm = 5 mm * (0.1/0.6) * $(1.2/1.5)^5$
- sdd 0.08 μm = 5 mm * (0.08/0.6) * $(1.2/1.5)^6$
- sdd 0.06 μm = 5 mm * (0.06/0.6) * $(1.2/1.5)^7$

Fig. 1: Signal drive distances.

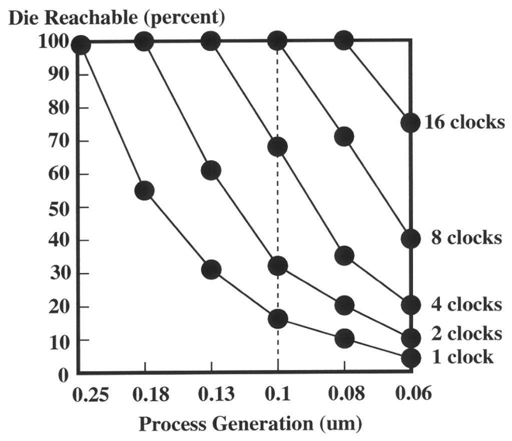

Fig. 2: The clock locality metric.

distance is, the better the accesses service area, especially in the square of services area based on the signal drive region (area) of signal drive distance by signal drive distance. If we cascade the signal drive distance in series, the configuration starts with a pair of gate and line and ends with the end gates. Generally speaking,

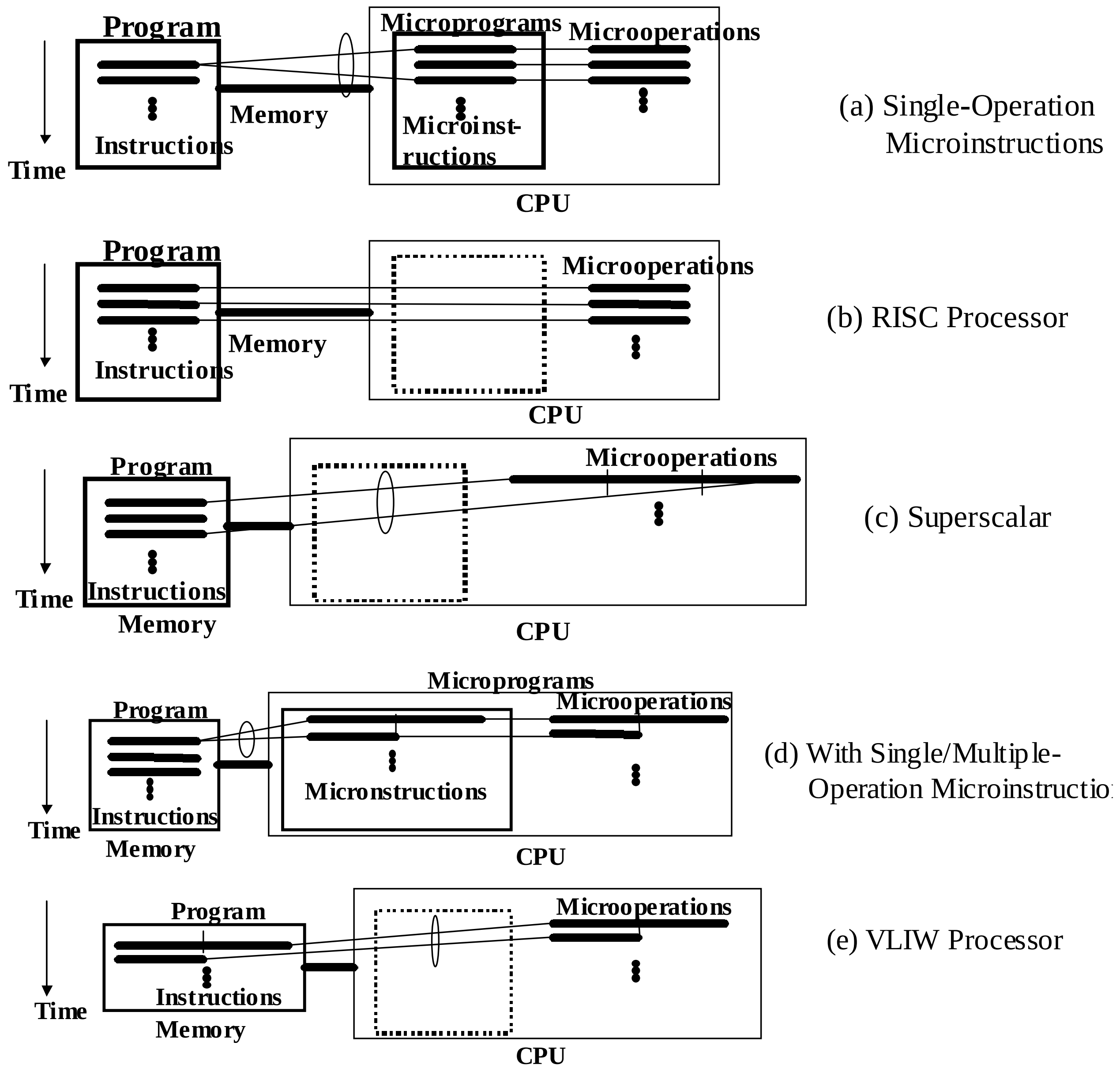

Fig. 3: Microoperations being extended on a Superscalar and VLIW Processor.

as one clock is 25 signal drive distances, the configuration is 13 gates and 12 lines. Using the line connections we can divide the side of the chip by the line we can get how many logic islands there are on the chip. Fig. 2 shows all the cases of such logic islands. $0.25\mu m$ CMOS technology reaches the end of edge of chips during one clock cycle. However, the advanced CMOS technologies, only one clock cycle is not always enough to propagate one signal from edge to edge. This means we have logic islands along the signal propagation on the chip. In our model we make use of these logic islands and through the signal propagation we have a pipeline.

The most important operations in a computer system are microoperations that are register transfer operations. Fig. 3(a) shows the CISC area where microoperations are most fundamental for microinstructions that constitute a microprogram. RISC had changed the relation of microinstructions and microoperations into the relation of machine instructions and microoperations. The reason is obvious in terms of RISC, Reduced Instruction Set Computer. In RISCs, superscalar processors are introduced into computer systems. The principal is Instruction Level Parallelism at RISC level, the microoperartions are at the same time processed. Then, Fig. 3 (c) shows the mode of parallelism. On the other hand, Fig. 3(d) shows the case where one microinstruction has a plural number of microoperations, and the machine instruction has a plural number of microoperations in RISC processors. Especially the microinstruction is a horizontal and have a plural number of microoperations. The microinstructions are graded up to the machine instructions to be VLIW instructions. Fig. 3(e) shows VLIW processor. Each function of the VLIW cells is corresponding to a microoperation of such microinstructions. These examples show that RISC processors, Superrscalar and VLIW, further superpipeline processors stem from microoperations on the datapath.

The most important is to get parallelism on the datapath reasonably. Datapaths are consisting of registers, ALUs and buses. The reasonable status is not to have buses as much as possible even in inner buses excepted von Neumann bottle neck. Von Neumannn bottleneck is the worst case so that we had better avoid to use the neck. To architect and design this concept, let us consider the logic islands due to wire-delayed chip environments. Fig. 4 shows an example of a bus-less and von Neumann bottleneck-less computer system consisting of only both an array of ALUs and shift registers of parallel in - parallel out.

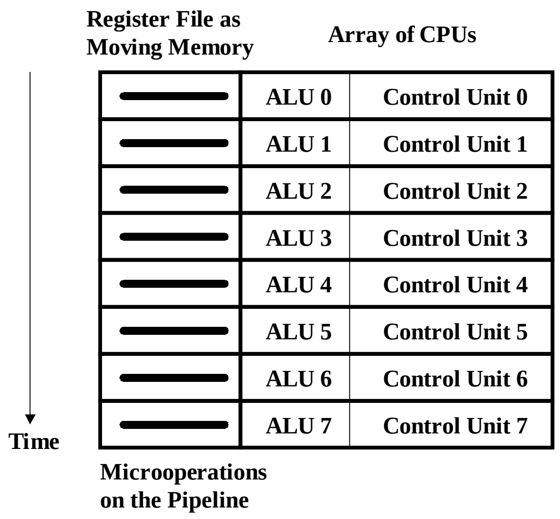

Fig. 4: The SHIFT Architecture.

The feature of this architecture, SHIFT Architecture is firstly no von Neumannn bottleneck that is designed by the moving register memory directly close to ALUs in a pipeline fashion. This configuration gives us the two merits, one of which is to provide no von Neumann bottleneck and the other is pipelining using logic islands.

3 Computation Model

The SHIFT Architecture is a design concept for implementing SHIFT Machine. The data processing scheme is very interestedly interpreted. The scheme is shown as the SHIFT Grammar. The SHIFT Machine is an implemetation of the SHIFT Architecture. The SHIFT Grammar is for a processing scheme of the SHIFT Machine.

The Processing Scheme is defined as follows:

$$P_{i+1} = D_i(P_i) \tag{1}$$

$$D_{i+1} = P_i(D_i) \tag{2}$$

where P_i is either algorithms and data structure or algorithms in the i-th stage. D_i is either data or data structures and data in the i-th stage.

Equations (1)(2) indicate that a program which processes data in a step is changed into the next program to process the data in the next step.

An example scheme of fine grain processing is

$$A = B * C - (D + E)/(E - F)$$

This is represented as follows in Reverse Polish notation:

$$ABC * DE + EF - /- =$$

Its tree representation is in Fig. 5. In here we

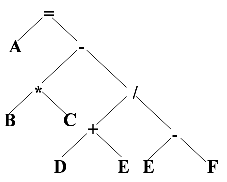

Fig. 5: The tree representation.

have two interpretations on the change of either program or data. One is a case where program is algorithms plus data structures and data is a set of data. The other is a case where a program is an ordered set of operations seemed to be an algorithm and data is equal to a set of data and data structures. The distinction is listed in Fig. 6. Any of the two is interpreted as a functional change of programs and data by the expressions (1) and (2) while processing.

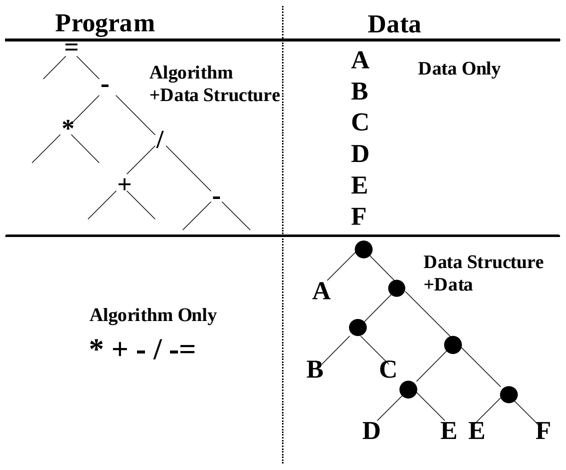

Fig. 6: The relationship between program and data.

4 Evaluation on the SHIFT Machine

Using the SHIFT machine, let us compute the Discrete Cosine transfer

$$F(u,v) = \frac{1}{4} c_u c_v \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cos\frac{(2x+1)u\pi}{16} \cos\frac{(2y+1)v\pi}{16}$$

$$0 < u, v < 7 \quad , \quad 0 < x, y < 7 \qquad (3)$$

On the SHIFT Machine the equation 3 is evaluated in two scheduling policies. Fig. 7 shows not-optimaized and optimized ways. In a general not-optimized way, a order of the threads of sequential instructions performed through the SHIFT Machine is the same as the order of scalar instructions in the program and that way have to perform extra addition threads to make flow of computation in sum of trees. This way does not seem to be optimized within the case of from $u = 0$ and $v = 0$ to $u = terminalNo.$ and $v = terminalNo.$

On the other hand, if we arrange the computation order of a pair of u and v along the outer loops, the thread schedule is optimized. These two ways have an obvious difference between the two in clock cycles for the whole computation of (3). Fig. 8 shows the comparison of

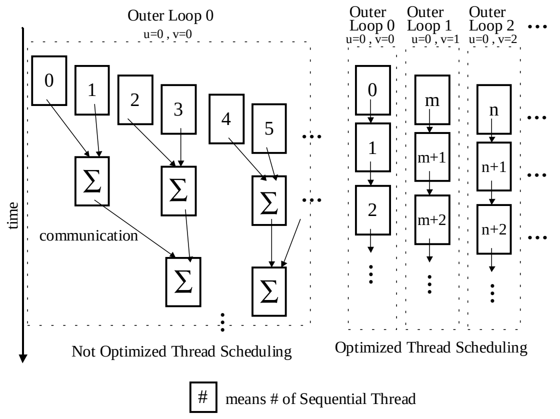

Fig. 7: Thread scheduling policies.

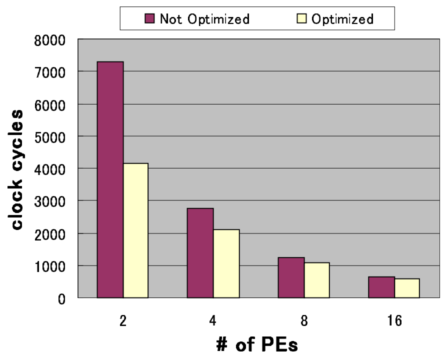

Fig. 8: The performance of the SHIFT Machine driven by Reverse Polish notation.

each of cases in the evaluation of (3). The abscissa is for the number of stages (PEs) of the SHIFT machine and the ordinate is clock cycles to require the whole computation of (3). We obviously understand the optimized effects especially in smaller number of stages (PEs).

5 Conclusion

We have tried to exploit and develop the new architecture, the SHIFT Architecture using the features / characteristics of more than one-billion transistor chips with wire-delayed environments. Logic islands exist on the chip and the cascade of logic islands creates pipelined data processing units. The grain of one logic island is relatively larger compared with usual pipelines such as conventional arithmetic pipelines in vector processors. Therefore, we have realized a general purpose pipeline that can perform thread instructions in parallel on the SHIFT machine as an implementation of the SHIFT Architecture. Then we benchmarked the simple program, Discrete Cosine Transfer on the machine. The results are reasonable to be able to use the machine as multi-threaded parallel processor in MISD fashion.

References

[1] Michael J. Flynn, Patrick Hung, and Kevin W. Rudd. Deep-submicron microprocessor design issues. *IEEE MICRO*, July/August 1999.

[2] Clecio D. Lima, Kentaro Sano, Hiroaki Kobayashi, Tadao Nakamura, and Michael J. Flynn. A technology-scalable multithreaded architecture. *Proc. of the 13th Symp. on Computer Architecture and High Performance Computing*, 2001.

[3] Doug Matzke. Will physical scalability sabotage performance gains? *IEEE Computer*, pp. 37–39, September 1997.

[4] Tadao Nakamura. Introducing cool chips. *IEEE MICRO*, July-August 1999.

[5] Michael Schlansker, Thomas M. Conte, James Dehnert, Kemal Ebcioglu, Jesse Z. Fang, and Carol L.Thompson. Compilers for instruction-level parallelism. *IEEE Computer*, December 1997.