TITAC–2 : An asynchronous 32-bit microprocessor based on Scalable-Delay-Insensitive model

Akihiro Takamura† Masashi Kuwako‡ Masashi Imai‡ Taro Fujii†
Motokazu Ozawa† Izumi Fukasaku† Yoichiro Ueno† Takashi Nanya†‡

† Graduate School of Information Science and Engineering
Tokyo Institute of Technology
2–12–1 O-okayama, Meguro-ku, Tokyo 152, Japan

‡ Research Center for Advanced Science and Technology
University of Tokyo
4–6–1 Komaba, Meguro-ku, Tokyo 153, Japan

Abstract

Asynchronous design has a potential of solving many difficulties, such as clock skew and power consumption, which synchronous counterpart suffers with current and future VLSI technologies. This paper proposes a new delay model, the scalable-delay-insensitive (SDI) model, for dependable and high-performance asynchronous VLSI system design. Then, based on the SDI model, the paper presents the design, chip implementation, and evaluation results of a 32-bit asynchronous microprocessor TITAC-2 whose instruction set is based on the MIPS R2000. The measured performance of TITAC-2 is 52.3MIPS using the Dhrystone V2.1 benchmark.

1 Introduction

Projecting forward from today, several reports suggest that, in 10 years, the CMOS technology will reach a point where the switching delay for a single gate is close to 10 picoseconds while a single chip area is nearly 10 square centimeters, with wiring moving into dominance[1]. The dominant wiring delay poses a fundamental limitation on synchronous system design that requires global clock distribution with as little skew as possible. The delay fluctuation may hinder the synchronous system from fully enjoying foreseeable high-speed devices to enhance system performance[10].

A natural solution to this problem is the introduction of asynchronous or self-timed computing. Without a global clock, asynchronous systems can enjoy, 1) average-case performance instead of worst-case performance, 2) low power consumption, 3) ease of modular composition, 4)no clock alignment at the interfaces, 5) timing fault tolerance, etc. Thus, asynchronous circuit/system design has been enjoying a worldwide resurgence of interest for the last decade. Scientific American described this situation as “reviving a challenger to the modern microchip” in its issue of June 1995.

There have been, however, only a few implementations reported of asynchronous general-purpose processors; a 16-bit pipeline CMOS RISC processor[8] and its GaAs version[15] at California Institute of Technology, an asynchronous version of the ARM processor AMULET1[3] and its revised version AMULET2e[2] at University of Manchester, and a simple 8-bit processor TITAC-1[11] at Tokyo Institute of Technology.

Delay models, i.e. assumptions on gate and wire delays, play an essential role in the design of digital systems. If the delay model is too pessimistic (i.e. too cautious when delay variation is unlikely), the resulting circuit can be unacceptably inefficient and expensive, as is the case for the delay-insensitive (DI) or quasi-delay-insensitive (QDI) model[4]. On the other hand, if it is too optimistic (i.e. gambling on fixed or no delays), the design may not guarantee correct circuit operations as is the case for synchronous circuits. Thus, the delay assumption must be carefully examined and validated for the device technology, the fabrication process and the operating environment that may affect the system's delay distribution throughout its life time.

In this paper, we first propose a new delay model, the scalable-delay-insensitive (SDI) model, which provides

with a practical base for dependable and high-performance asynchronous VLSI design. Then, based on the SDI model, we present the design, chip implementation, and evaluation results of a 32-bit asynchronous microprocessor TITAC-2 whose instruction set is based on the MIPS R2000[5].

2 Scalable-Delay-Insensitive Model

2.1 Definition

There are several well-defined delay models which are used as the base for the design of asynchronous circuits. Among these, one extreme model is the synchronous circuits where gate and wire delays are assumed to be fixed or bounded by prescribed minimum and maximum values that determine the clock period. The other extreme is the DI model or QDI model. The DI model imposes the least restriction on delays, i.e. it assumes that the wire delays, as well as gate delays, are finite but unbounded. The QDI model introduces the "isochronic forks[7]" assumption to the DI model.

Our previous processor TITAC-1 as well as the Caltech processors were designed using the QDI model. Design based on the DI or QDI model, however, is required to guarantee correct operations even in the presence of such a delay distribution that is unlikely to occur in practice. As a consequence, the resulting hardware volume tends to be larger, and the speed performance based on the current technology may tend to be lower than an equivalent design on a more optimistic delay model.

A promising design methodology is to divide the entire system into parts of a reasonable size, to design each part based on a more relaxed or optimistic delay model, and to design the interconnection parts or global parts based on the DI model.

This observation leads us to propose a new delay model, the scalable-delay-insensitive (SDI) model. It is very realistic and provides with a nice base for the design of reasonably reliable and high-performance asynchronous processors.

The SDI model is an unbounded delay model, i.e. no upper bound is assumed on the gate and wire delays. Unlike the DI or QDI model, the SDI assumes relative delay ratio between any two components is bounded. The SDI model is defined as follows;

Scalable Delay Insensitive model A component refers to a gate or a interconnection wire between two gates. Let $d1$ and $d2$ be the delay for any two components C1 and C2. The relative delay D of C1 to C2 is expressed as $D = d1/d2$. Let D_e denote the relative delay that is estimated by the designer, and D_a denote an actual relative delay observed through the system's lifetime. The relative variation ratio R is expressed as $R = D_a/D_e$. Then, the Scalable-Delay-Insensitive model assumes that the relative variation ratio R is bounded for any two components, that is, $1/K < R < K$, where K is a constant and will be called the maximum variation ratio.

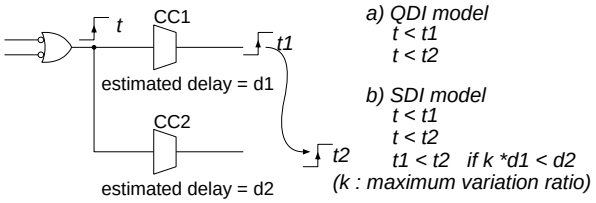

Figure 1. Guaranteed signal order:(a)the QDI model, (b)the SDI model.

Figure 1 illustrates the signal change ordering guaranteed in the QDI model and SDI model. What QDI model guarantees is that signal change t precedes signal changes t1 and t2. On the other hand, in the SDI model, suppose the estimated delays for the datapaths CC1 and CC2 be $d1$ and $d2$, respectively. Then, signal change t1 is guaranteed to precede t2 if $d2 > K \cdot d1$.

2.2 Speed comparison of SDI and QDI circuits

SDI circuits can run faster than the equivalent QDI ones. Consider an example taken from the TITAC-2 design. Figure 2 shows a register file containing 40 general-purpose 32-bit registers. 32-bit write data is written to a register selected by the register select signal and returns ack signal. QDI-ack is based on the QDI model and SDI-ack is based on the SDI model with the maximum variation ratio being 2. For simplicity, let us assume all gates have 2 inputs and a unit-time delay. Hence an N-input gate has a delay of $\lceil log_2 N \rceil$ units. For both models, after the write data and register select signal arrive, it takes 3 units to write write data to a register.

The QDI-ack will require an additional 13 units delay to be issued, as it takes 2 units to produce the Rack from each 1-bit register, 5 units to gather the Rack signals of all 32 1-bit registers and 6 units to produce the final QDI-ack.

On the other hand, the SDI-ack is produced in parallel with the data being written to the register. Its generation can start immediately after the write data and register select arrive. Therefore it takes only 7 units. It is guaranteed that the register will have been written to by this time even if the SDI-ack arrives 2 times earlier than the expected time.

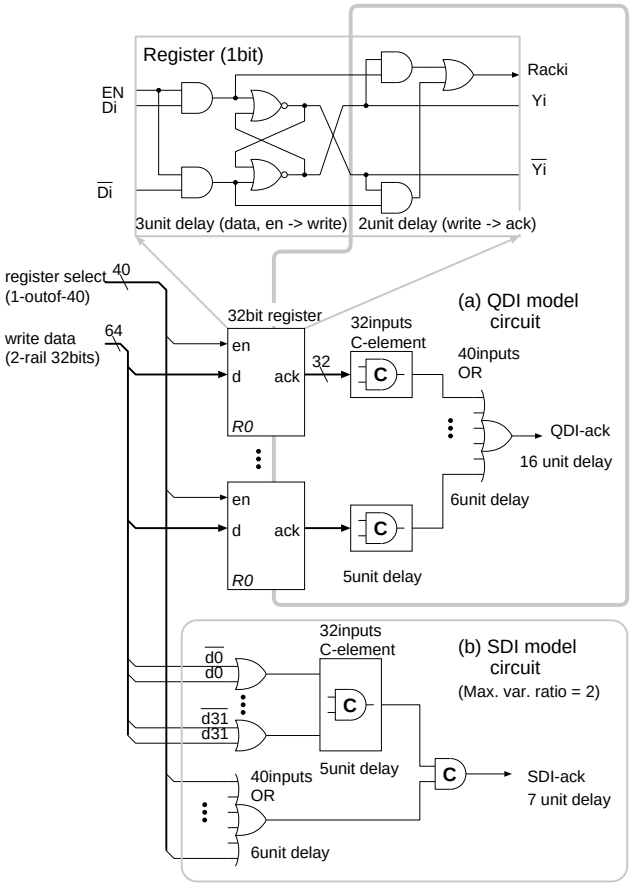

Figure 2. Completion generation for QDI circuit and SDI circuit.

2.3 Design guideline for SDI model

A general guideline for SDI model is given below.

step1 Divide the entire system into a number of functional blocks.

step2 Design each block as well as interconnections based on the QDI model.

step3 Determine maximum variation ratio K by taking into consideration the process technology used and area size required for each block.

step4 Apply SDI implementation as exemplified in Figure 2 to each QDI block whenever possible.

The above guideline impose a set of constraints not only on logic design, but also on technology-mapping and layout in order to validate the SDI model. If wire delays are dominant for possible delay distribution, a circuit in smaller area gives smaller maximum variation ratio K, which in turn makes the logic design easier. Therefore, it is recommended that the entire system is divided into sub-circuits with each sub-circuit being limited within a certain size of area. In the TITAC-2 design which will be described in later sections, we determined the area size in the following ways: we set the maximum variation ratio to 2. Then we assumed the estimated delay for all the gates to be an identical standard delay which is equivalent to the delay of a gate with 3 fanouts and 0.48 mm wire for the output interconnection. Wire delays are considered as a part of gate delays. Then the delay of a 2-input NAND gate, whose fanout equals to 1 and output wire length equals to 3.86mm, becomes 2 times as large as the standard delay for the technology used. Thus, we decided the maximum wire length allowed to be equal to 3.86mm. The maximum area to be used by each sub-circuit is therefore $(3.86/2)^2 mm^2$. There are some sub-circuits, however, that cannot fit inside the maximum area. In this case the maximum length of wires must be restricted in such sub-circuits not to exceed 3.86mm.

3 Processor organization and design

3.1 TITAC-2 architecture

TITAC-2 is a 32-bit asynchronous processor based on the SDI delay model. The architecture of TITAC-2 is very similar to the MIPS R2000 processor[5].

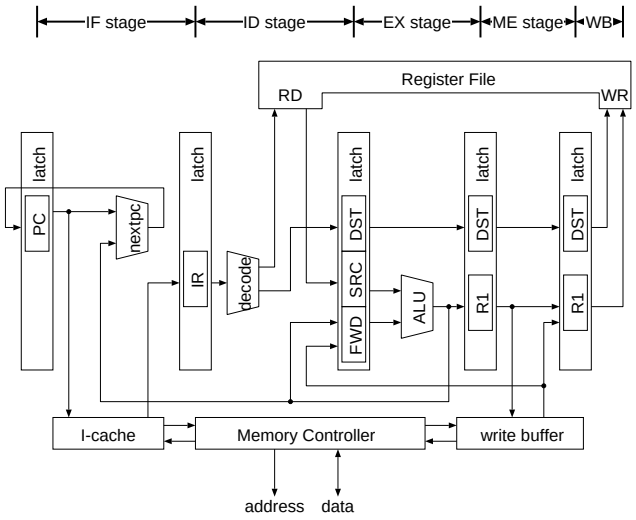

Figure 3. The pipeline structure of TITAC–2.

Instructions are executed through 5 stages, namely, IF (Instruction Fetch), ID (Instruction Decode), EX (EXecution), ME (MEmory Access) and WB (Write Back). The pipeline structure is shown in Figure 3.

The instruction set of the R2000 was modified for TITAC-2. Some R2000 instructions were not implemented, while a small number of R2000 instructions have modified operation. The modified instructions include multi-

ple/divide, privilege instruction and delay slots of branch instructions. In any case, the object code is not compatible because of a different instruction encoding.

3.2 Asynchronous pipeline behavior

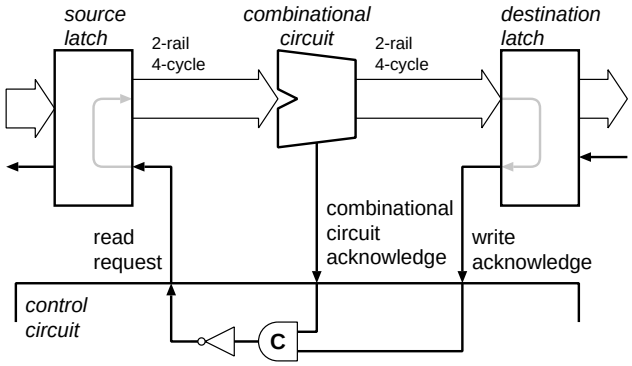

Figure 4. The model of an asynchronous pipeline circuit.

Figure 4 shows a model of pipeline circuits in TITAC-2. For data transfer, two-rail two-phase scheme[11], which may be called four-cycle signaling[12], is used except for the internal cache memory and external bus interface that use bundled-data assumption[14].

An asynchronous pipeline, such as micropipelines[14], manages the flow of data according to the state of the next and previous pipeline stages. In a synchronous pipeline, if a stage is late in completing operation of the combinational circuit, the entire pipeline is delayed by an amount of time equal to the clock period. For an asynchronous pipeline, however, if a stage is late by a duration α, the entire pipeline is delayed just by α. Also in contrast to synchronous pipelines, an asynchronous pipeline controller only judges the state of adjoining stages. Therefore decentralized control is possible.

3.3 Pipeline latch

Figure 5 shows the circuit for a latch in the asynchronous pipeline of TITAC-2. The pipeline latch can be either the source latch or the destination latch for the pipeline stage shown in Figure 4. It consists of 4 primitive latches, including 2 Muller's C-elements[9] and 1 NOR gate, in cascade.

When the number of primitive latches is N, the stage latch works as a FIFO buffer which has N entries. When the number of primitive latches decreases, it is more likely that the entire FIFO buffer is full. Hence, the expected delay in entering the FIFO buffer will get longer. On the other hand, if the number of primitive latches increases, even though

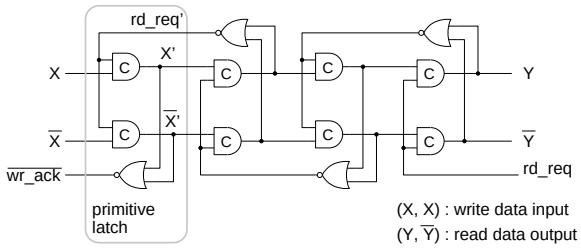

Figure 5. A 2-rail 4-cycle asynchronous latch.

the expected delay in entering the FIFO buffer decreases, the delay between the input and output of the FIFO buffer increases.

For a general pipeline structure, that includes a feedback loop, at least 3 primitive latches are needed[13]. However, it is a difficult problem to find the optimal number of primitive latches for general pipeline structure. We determined the number N of primitive latches by simulating the performance of TITAC-2 for $N = 3, 4, 5$. Cases for $N = 4$ and $N = 5$ show almost the same level of performance, while the case for $N = 3$ is 8% worse than the other two. Therefore, we decided the number of primitive latches to be 4.

3.4 Pipeline controller

In general, an output of a source latch is shared between two or more combinational circuits and the corresponding destination latches. In this case, update of the source latch's output must be carried out after receiving all the write acknowledge signals from combinational circuits and corresponding destination latches which use the output of the source latch.

Therefore, a control circuit is synthesized using the following method : (1) For each combinational circuit with its corresponding destination latch, the write acknowledge signal of a destination latch and the combinational circuit are connected to the inputs of a C-element. (2) In the case when we have multiple pairs of combinational circuits and destination latches the acknowledgments generated by each C-element are combined together using a multi-input C-element. (3) The negated signal produced by (1) and optionally (2), above is the read request signal that is passed to the source latch.

Figure 6 shows an example of a pipeline control circuit which includes shared source latch and separated source latch. The output of the source latch U is shared by V and X. Thus U.rd_req is using C-element and inverter to merge V.wr_ack2 and X.wr_ack2. On the other hand, the output of source latch W is not shared. Thus W.rd_req is generated by simply inverting the signal of X.wr_ack.

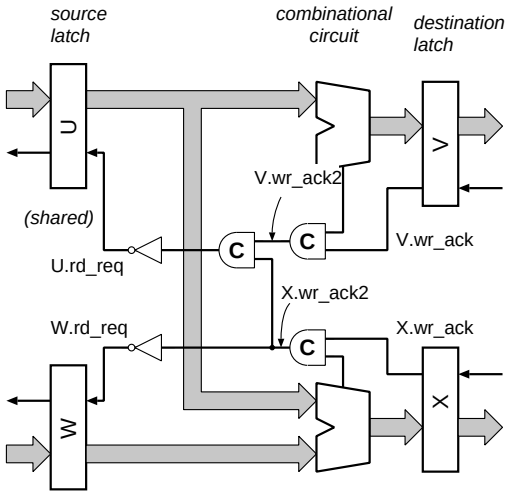

Figure 6. A controller for a shared source latch.

3.5 ALU

The ALU comprises 6 functional unit, ADD, SUB, LOGIC (AND/OR/XOR), SHIFT, MUL and DIV. The SELECTOR chooses a functional unit specified by an instruction, and the MERGER joins the results. Simply, the SELECTOR consists of AND gates and the MERGER consists of OR gates.

The ADD and SUB functional units uses Carry Look Ahead (CLA) logic whose group size is 4. The completion signal of the ADD/SUB unit is made from group's carry signals of first order carry look ahead logic because ADD/SUB unit complete its output within a certain delay after all group's carry signals are generated.

The structure of the MUL functional unit is a Wallace-Tree and generate 32-bit result from two 32-bit inputs. Each half/full adder is implemented by Differential Cascode Voltage Switch Logic, DCVSL[6] synthesized from BDD specification[11]. DCVSL circuit's precharge (evaluation) signal wires and buffers are placed like the H-tree for isochronic signal transition.

The DIV functional unit gets 32-bit dividend and 32-bit divisor and generate 32-bit result using the restoring method. Usually, carry look ahead subtracters calculate sign of the partial remainder earlier than the all of the partial remainder. Therefore, the restoring method is faster than the non-restoring method. In this implementation, using 2 subtracters and 2 multiplexers, 2-bit of partial quotient is obtained in a cycle. The number of cycle depends on divisor and dividend, however, 16 cycles are needed in worst case.

3.6 Instruction Cache

The instruction cache unit on TITAC-2 uses direct-mapped, early-restart with streaming. The cache size is 8KB and the line size is 8 words. It takes 8 cycles to fill the line after a cache miss, however the instruction fetch cycle is not fixed and occasionally stops when the cache fills the missed line while the processor requires another line. To fetch instructions and fill missed line efficiently, it is needed to run cache fill process in parallel with instruction fetch cycle. But previous register transfer model shown in Figure 4 impose that connected circuit blocks with FIFO latches run same cycles. Thus the previous register transfer model is unsuitable for the cache.

The key point is the order of controlling two flags, line fill flag (LF) and 8 exist this word flags (ETW). The line fill process reads 1-word from main memory, then writes to the cache memory and set ETW flag. After repeating this action 8 times, clears the LF flag.

For a start, the instruction fetch process checks if the required address is the same as the previous accessed line. If it is same, data is obtained after setting the ETW flag. Otherwise the cache tag memory is checked whether data is exist in the cache memory or not. If data exist in the cache memory, data is read from the cache memory. Otherwise 1) wait until LF flag is cleared, 2) clear all ETW flags and set LF flag, 3) start line fill process. Requested data will be obtained when the line fill process get data from main memory and set ETW flag.

The cache memory is composed memory macro cells provided by the chip manufacturer. Since the interfaces of the memory macros are not 2-rail, delay element is needed for bundled data encoding. We use presettable delay elements to set an appropriate delay after the chip is manufactured. In this way, we improve the processor's reliability against delay fluctuations without penalizing its performance.

3.7 Register File

An asynchronous register file has to handle write-after-read and read-after-write dependency appropriately. In TITAC-2, WB stage, where data is written writing to register file, and ID stage, where data is read from register file, are synchronized with each other. Thus, write-after-read dependency is always satisfied because the read operation of a register, that is the target of the current write operation, has been completed at least 1 cycle before the write operation.

If the destination register of the WB stage and the source register of the ID stage are identical, the write process must be completed before starting the read process in order to satisfy read-after-write dependency. Usually, source and destination register indices arrive before write data arrives. If

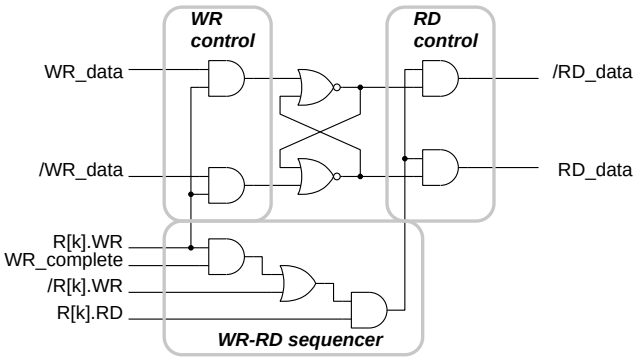

Figure 7. Register file read after write sequencer.

the source register and the destination register are not identical, the reading process and the writing process can run in parallel. Figure 7 shows read-after-write sequencer in the register file. If the source register and the destination register are identical ($R[k].RD = 1$ and $R[k].WR = 1$), read waits $R[k].WR\_complete$. Otherwise, the source register and the destination register are not identical ($R[k].RD = 1$ and $/R[k].WR = 1$) and read is executed in parallel with write.

3.8 Interface

Although, the data encoding is 2-rail in most parts of TITAC-2, bundled data encoding is used for the interface to outside the chip. The reasons for using bundled-data encoding are data transfer speed to external device is slower and easier to estimate delays than internal circuit, it is more easy to connect with off-the-shelf devices, and fewer pin count than 2-rail interface.

Bundled-data circuits can simulate 2-rail data transfer by using 2-rail to bundled data and bundled data to 2-rail converters[11].

4 Chip Evaluation

The TITAC-2 chip was fabricated using 3 layer metal, 0.5 micron rule CMOS standard cell technology: 496,367 MOS transistors as well as 8.6KByte cache and tag memory are integrated in $12.15\text{mm} \times 12.15\text{mm}$. The chip layout is shown in Figure 8.

The first silicon of TITAC-2 runs correctly with its power supply voltage being varied through the range from 1.5 V to 6.0 V and the temperature of its package surface being heated up to about 100°C and cooled down to -196°C with liquid nitrogen.

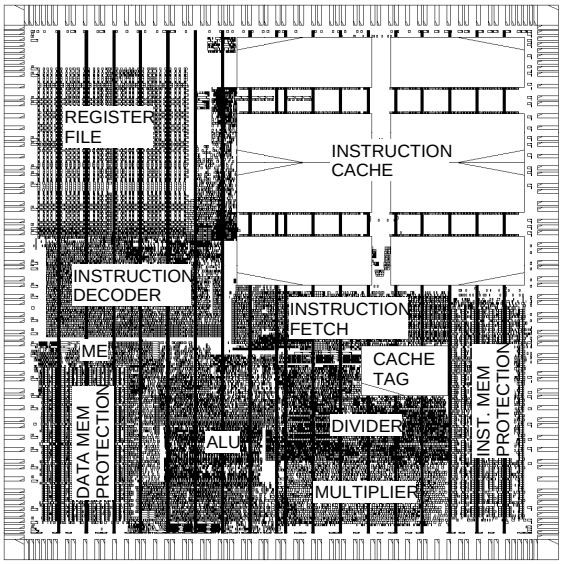

Figure 8. TITAC–2 chip image

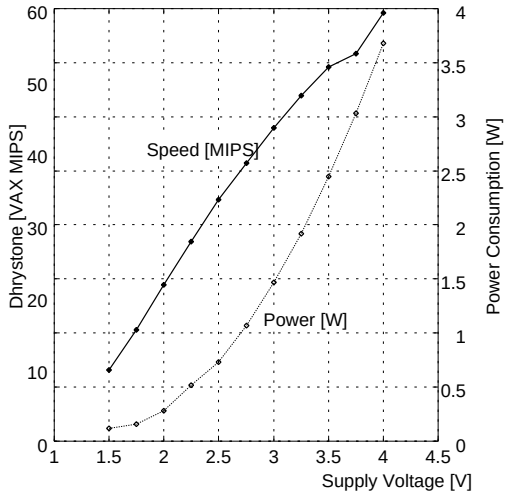

Figure 9. Supply voltage versus speed and power consumption.

Figure 9 shows measured speed and power consumption versus power supply voltage changing from 1.5V to 4.0V at 20°C. TITAC-2 achieves 52.3 VAX MIPS using the Dhrystone V2.1 benchmark and consumes 2.11W at 3.3V, 20°C.

5 Conclusions

Asynchronous design methodology is a promising approach to overcome the limitation of synchronous design due to the wire delay and clock skew problems encountered in modern VLSI system design. Delay models play an es-

sential role in the asynchronous system design. We have proposed a new realistic delay model, called the Scalable-Delay-Insensitive model which enables asynchronous system design to achieve a reasonable level of timing reliability and high-performance operation. Based on the SDI model, we have presented the design and implementation of 32-bit asynchronous microprocessor TITAC-2 whose architecture is almost the same as the MIPS R2000 RISC processor, including five stage pipeline, on-chip instruction cache, exception handling, external interruption, memory protection, etc. The speed of TITAC-2 chip fabricated with 0.5 micron rule CMOS standard cell technology has been measured to be 52.3 VAX MIPS using the Dhrystone V2.1 benchmark. The implementation results are very encouraging for future VLSI system design, although there still remain many challenges, including development of an asynchrony-oriented architecture instead of the existing synchronous RISC architecture, asynchronous verification, test techniques, and asynchronous CAD environment.

Acknowledgments

The authors would like to thank Rafael Morizawa, Makoto Ishikawa, Shinya Nishikawa, Yoshiro Ikeda, Norihito Ishida, Atsushi Yamazaki of Tokyo Institute of Technology for their contribution to the TITAC-2 design, Dr. Masao Fukuma, Dr. Masakatsu Yamashina, Hiroki Fujimoto, Fuyuki Okamoto, Osamu Fujita of NEC Microelectronics Research Lab. for their cooperation in the TITAC-2 chip fabrication. Thanks also go to Dr. Andreas Savva of the University of Tokyo for his careful proofreading of this paper. This work was supported in part by the Ministry of ESSC under Grant-in-aid for Scientific research No.07558036 and by NEDO under the proposal-based Advanced Industrial Technology R&D Program.

References

[1] W. F. Brinkman. The transistor: 50 glorious years and where we are going. In *International Solid State Circuits Conference*, pages 22–27, feb 1997.

[2] S. B. Furber, P. Day, J. D. Garside, N. C. Paver, and S. Temple. AMULET2e. In C. Muller-Schloer, F. Geerinckx, B. Stanford-Smith, and R. van Riet, editors, *Embedded Microprocessor Systems*, Sept. 1996. Proceedings of EMSYS'96 - OMI Sixth Annual Conference.

[3] S. B. Furber, P. Day, J. D. Garside, N. C. Paver, and J. V. Woods. A micropipelined ARM. In T. Yanagawa and P. A. Ivey, editors, *Proceedings of VLSI 93*, pages 5.4.1–5.4.10, Sept. 1993.

[4] S. Hauck. Asynchronous design methodologies: An overview. *Proceedings of the IEEE*, 83(1), Jan. 1995.

[5] G. Kane and J. Heinrich. *MIPS RISC ARCHITECTURE*. PRENTICE HALL, 1992.

[6] W. R. G. Lawrence G. Heller. Cascode voltage switch logic: A differential cmos logic family. *IEEE International Solid–State Circuits Conference*, Digest of Technical Papers:16–17, February 1984.

[7] A. J. Martin. The limitations to delay-insensitivity in asynchronous circuits. In W. J. Dally, editor, *Sixth MIT Conference on Advanced Research in VLSI*, pages 263–278. MIT Press, 1990.

[8] A. J. Martin, S. M. Burns, T. K. Lee, D. Borkovic, and P. J. Hazewindus. The design of an asynchronous microprocessor. In C. L. Seitz, editor, *Advanced Research in VLSI: Proceedings of the Decennial Caltech Conference on VLSI*, pages 351–373. MIT Press, 1989.

[9] R. E. Miller. *Combinational Circuits*, volume 1 of *Switching Theory*. John Wiley & Sons, 1965.

[10] T. Nanya. Challenges to dependable asynchronous processor design. In T. Sasao, editor, *Logic Synthesis and Optimization*, chapter 9, pages 191–213. Kluwer Academic Publishers, 1993.

[11] T. Nanya, Y. Ueno, H. Kagotani, M. Kuwako, and A. Takamura. TITAC: Design of a quasi-delay-insensitive microprocessor. *IEEE Design & Test of Computers*, 11(2):50–63, 1994.

[12] C. L. Seitz. System timing. In C. A. Mead and L. A. Conway, editors, *Introduction to VLSI Systems*, chapter 7. Addison-Wesley, 1980.

[13] J. Sparsø and J. Staunstrup. Design and performance analysis of delay insensitive multi-ring structures. In *Proc. Hawaii International Conf. System Sciences*, volume I, pages 349–358. IEEE Computer Society Press, Jan. 1993.

[14] I. E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, June 1989.

[15] J. A. Tierno, A. J. Martin, D. Borkovic, and T. K. Lee. A 100-MIPS GaAs asynchronous microprocessor. *IEEE Design & Test of Computers*, 11(2):43–49, 1994.