the partial synchronizing sequence is transformed into constraints for the state assignment problem. After the encoding is obtained, the state variables that need a reset signal can be easily identified.

VII. Conclusion

We have studied the initializability problem of finite state machines. Many machines synthesized by current automatic synthesis systems cannot be initialized by gate-level analysis tools, even when a synchronizing sequence exists. State assignment and logic minimization play important roles in fixing this discrepancy. We have derived the conditions for state assignment for initializability and described an algorithm to incorporate them in an automatic state assignment program. We also suggest that a single-output minimization may have better initializability than multioutput minimization. Experimental results indicate that, in general, initializability can be achieved without any appreciable overhead.

If the machine does not have a synchronizing sequence or the synchronizing sequence is too long, a partial reset technique is suggested that will be less expensive than the full reset (i.e., reset every state variable). Partial reset methods require further investigation.

References

[1] G. De Micheli *et al.*, "Optimal state assignment for finite state machines," *IEEE Trans. Comput.-Aided Design*, vol. CAD-4, pp. 269–285, July 1985.
[2] S. Devadas *et al.*, "MUSTANG: State assignment of finite state machines targeting multi-level logic implementations," *IEEE Trans. Comput.-Aided Design*, vol. CAD-7, pp. 1290–1300, Dec. 1988.
[3] R. Brayton *et al.*, *Logic Minimization Algorithms for VLSI Synthesis*. Boston, MA: Kluwer Academic, 1984.
[4] R. Brayton *et al.*, "MIS: A multiple level logic optimization system," *IEEE Trans. Comput.-Aided Design*, vol. CAD-6, pp. 1062–1081, Nov. 1987.
[5] K. Keutzer, "DAGON: Technology binding and local optimization by DAG matching," in *Proc. 24th Design Automat. Conf.*, June 1987, pp. 341–347.
[6] V. D. Agrawal, K. T. Cheng, and P. Agrawal, "A directed search method for test generation using a concurrent fault simulator," *IEEE Trans. Comput.-Aided Design*, vol. CAD-8, pp. 131–138, Feb. 1989.
[7] S. Mallela and S. Wu, "A sequential circuit test generation system," in *Proc. Int. Test Conf.*, Philadelphia, PA, Nov. 1985, pp. 57–61.
[8] K. T. Cheng and V. D. Agrawal, "Concurrent test generation and design for testability," in *Proc. ISCAS*, Portland, OR, May 1989, pp. 1935–1938.
[9] G. Hachtel, R. Jacoby, K. Keutzer, and C. Morrison, "On properties of algebraic transformations and the multifault testability of multilevel logic," in *Proc. Int. Conf. Comput.-Aided Design*, Santa Clara, CA, Nov. 1989, pp. 422–425.
[10] K. Keutzer, private communication.
[11] Z. Kohavi, *Switching and Finite Automata Theory*, second edition. New York: McGraw-Hill, 1978.
[12] A. Miczo, *Digital Logic Testing and Simulation*. New York: Harper and Row, 1986.
[13] T. Villa and A. L. Sangiovanni-Vincentelli, "NOVA: State assignment of finite state machines for optimal two-level logic implementations," in *Proc. 26th Design Automat. Conf.*, June 1989, pp. 327–332.
[14] R. Lisanke, "Finite-state machine benchmark set," *Preliminary Benchmark Collection*, Sept. 1987.

A Microprocessor-Based Office Image Processing System—An Extension of Work

M. Atiquzzaman and W. H. Shehadah

***Abstract*—This paper is an extension of the work done by Ni *et al.* [1] on microprocessor-based office image processing system. In [1] the authors did not distinguish between compute-bound and I/O-bound state of the machine in the case where image input time is greater than the output time. We have clearly shown the above distinction and derived the correct expressions for such cases.**

***Index Terms*—Image processing, multiprocessors, processor scheduling policies, shared-bus systems.**

I. Introduction

A multimicroprocessor system, consisting of m processors and a single common bus, for processing of images has been described in [1]. Two processor scheduling policies have been discussed, and a deterministic method of analysis of performance of the system has been carried out in terms of the scheduling policies. The state of the machine can be described as either 1) I/O-bound when the bus is always busy and the processor is idling, 2) compute-bound when the processors are always busy and the bus is idling, and 3) near optimal when the bus and the processors are always busy.

For the three-step overlapping policy the authors have distinguished between compute-bound and I/O-bound *only when the image input time is less than or equal to the output time* ((3) in [1]). *But when input time is greater than the output time, the authors have not considered compute and I/O-bound cases resulting in an incorrect expression for execution time.* In this correspondence, we have considered the above two different cases, and subsequently derived the correct expressions along with relevant state-time diagrams of the multiprocessor system.

II. Notation

We use the same notations used in [1].

t_i = time to input a segment

t_o = time to output a segment

t_p = time to process a segment

s = number of segments of an image

$T_3(s, m)$ = time to process s segments of an image using m processors and a three-step scheduling policy

III. Execution Time when $t_i > t_o$

As given in [1]

$$T_3(s, m) = \lfloor s/m \rfloor T(m) + T(s \bmod m) \tag{1}$$

where $T(n)$ is defined by

$$T(n) = t_i + t_p + nt_o \quad \text{if } t_i \le t_o \quad \text{and} \quad t_p \ge (n-1)t_i \tag{2}$$

$$= n(t_i + t_o) \quad \text{if } t_i \le t_o \quad \text{and} \quad t_p < (n-1)t_i \tag{3}$$

$$= nt_i + t_p + t_o \quad \text{if } t_i > t_o. \tag{4}$$

Manuscript received June 1, 1989; revised June 1, 1990. This work was supported by King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia.

M. Atiquzzaman is with the Department of Computer Science and Computer Engineering, La Trobe University, Bundoora, Victoria, 3083 Australia.

W. H. Shehadah is with the Department of Electrical Engineering, King Fahd University of Petroleum and Minerals, Dhahran 31261, Saudi Arabia.

IEEE Log Number 9102633.

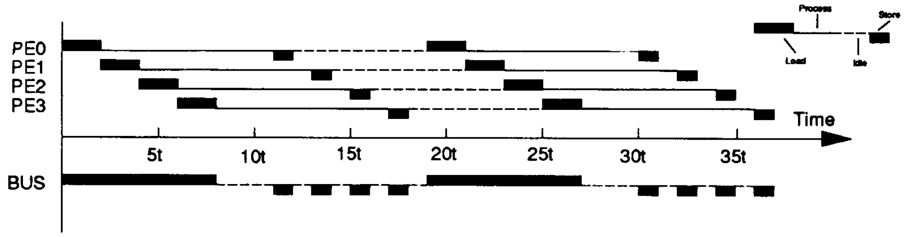

Fig. 1. Compute-bound: $t_p = 9t$, $t_i = 2t$, $t_o = t$.

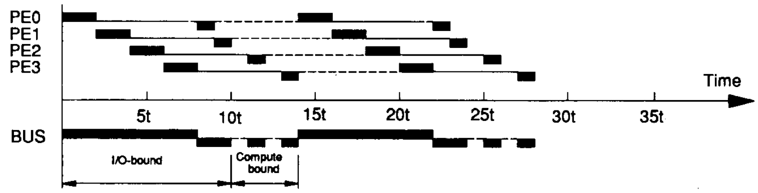

Fig. 2. Semi I/O-bound: $t_p = 5t$, $t_i = 2t$, $t_o = t$.

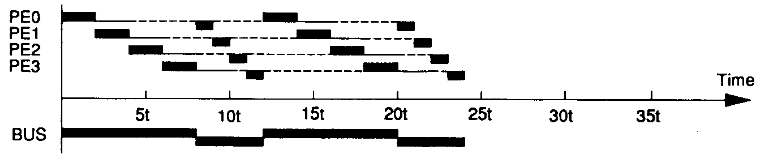

Fig. 3. I/O-bound: $t_p = 2t$, $t_i = 2t$, $t_o = t$.

We agree to (1), (2), and (3) but disagree with (4). Fig. 3 shows a situation where $t_i > t_o$. Clearly $T(4) = 12t$, whereas according to (3) it should be $4.2t + 2t + t = 11t$. When $t_i > t_o$ we propose three different states of the machine, and also determine the correct expression for $T(n)$ as follows.

Case 1: Compute bound: $t_i > t_o$ and $t_p \geq (n-1)t_i$.

Fig. 1 shows the state diagram where $t_p = 9t$, $t_i = 2t$, and $t_o = t$. As is evident from the diagram the processors are always busy, and the bus is seen to idle during certain times. Therefore,

$$T(n) = nt_i + t_p + t_o \quad \text{if } t_i > t_o \quad \text{and} \quad t_p \geq (n-1)t_i.$$

Case 2: Semi I/O Bound: $t_i > t_o$, $t_p < (n-1)t_i$, and $t_p > (n-1)t_o$.

Fig. 2 shows the semi I/O-bound state of the machine where $t_p = 5t$, $t_i = 2t$, and $t_o = t$. PE_0 idles for time t upto which point the system is I/O-bound. But processors PE_1, PE_2, and PE_3 do not idle making the system compute-bound. The system switches from I/O-bound to compute-bound state as shown in Fig. 2. Therefore,

$$T(n) = nt_i + t_p + t_o \quad \text{if } t_i > t_o, t_p < (n-1)t_i \quad \text{and} \quad t_p > (n-1)t_o.$$

Case 3: Totally I/O-bound: $t_i > t_o$, $t_p < (n-1)t_i$, and $t_p \leq (n-1)t_o$.

Fig. 3 depicts the totally I/O-bound state of the machine, where $t_p = 2t$, $t_i = 2t$, and $t_o = t$. All the processors are idle for a certain period, but the bus is always busy. Therefore,

$$T(n) = nt_i + nt_o \quad \text{if } t_i > t_o, t_p < (n-1)t_i, \quad \text{and} \quad t_p \leq (n-1)t_o.$$

IV. Switchover Point and Modified Expressions

As has been observed for $t_i > t_o$, when $t_p \geq (n-1)t_i$ the system is always compute-bound. When $t_p < (n-1)t_i$ the system can be totally I/O-bound or semi I/O-bound. The switchover point from totally I/O-bound to semi I/O-bound is $t_p = (n-1)t_o$ at which point the bus is always busy, and the last processor is also completely busy. Increasing t_p results in semi-I/O-bound and decreasing t_p results in a totally I/O-bound state. The resulting modified execution times for a three-step overlapping policy can be summarized as follows.

$$T_3(s, m) = \lfloor s/m \rfloor T(m) + T(s \bmod m)$$

where $T(n)$ is given by

$$\begin{aligned} T(n) &= t_i + t_p + nt_o \quad \text{if } (t_i \leq t_o) \wedge (t_p \geq (n-1)t_i) \\ &= n(t_i + t_o) \quad \text{if } ((t_i \leq t_o) \wedge (t_p < (n-1)t_i)) \\ &\quad \vee ((t_i > t_o) \wedge (t_p < (n-1)t_i) \wedge (t_p \leq (n-1)t_o)) \\ &= nt_i + t_p + t_o \quad \text{if } ((t_i > t_o) \wedge (t_p \geq (n-1)t_i)) \\ &\quad \vee ((t_i > t_o) \wedge (t_p < (n-1)t_i) \wedge (t_p > (n-1)t_o)). \end{aligned}$$

V. Conclusions

We have shown that when $t_i > t_o$ the total execution time also depends on t_p. The expressions derived in [1] do not show this dependence on t_p. Furthermore, we have identified three possible conditions of the machine. Appropriate expressions for execution times have been derived for all three conditions, and modified equations have been presented.

REFERENCES

[1] L. M. Ni, K. Y. Wong, D. T. Lee, and R. K. Poon, "A microprocessor-based office image processing system," *IEEE Trans. Comput.*, vol. C-31, pp. 1017–22, Oct. 1982.

An Algorithm for Optimal Static Load Balancing in Distributed Computer Systems

Chonggun Kim and Hisao Kameda

Abstract—**This paper proposes a load balancing algorithm that determines the optimal load for each host so as to minimize the overall mean job response time in a distributed computer system that consists of heterogeneous hosts. The algorithm is a simplified and easily understandable version of the single-point algorithm originally presented by Tantawi and Towsley.**

Index Terms—**Distributed computer systems, local area networks, optimal load, optimal static load balancing, single-point algorithm, star network configurations.**

I. INTRODUCTION

Tantawi and Towsley studied a model of a distributed computer system that consists of a set of heterogeneous host computers connected by a communications network [5]. They considered an optimal static load balancing strategy which determines the optimal load at each host so as to minimize the mean job response time. A key assumption of theirs was that the communication delay does not depend on the source–destination pair. This assumption may apply to single channel networks such as satellite networks and some LAN's. Given this assumption, they determined the requirement that the optimal load at each host satisfies, and derived an algorithm that determines the optimal load at each host for given system parameters. It is this algorithm that they call a single-point algorithm.

The Tantawi and Towsley single-point algorithm [5] is surprising in the sense that it does not calculate the load at each node iteratively. Note that previous algorithms on related models such as flow-deviation type algorithms (see, e.g., Fratta, Gerla, and Kleinrock [2]) and Gauss–Seidel type algorithms (see, e.g., Dafermos and Sparrow [1] and Magnanti [3]) require iterative calculation of loads. However, the algorithm appears to be complicated and rather difficult to understand.

In this paper, we consider the same model as Tantawi and Towsley [5] under the same assumptions concerning the communication delay. Additionally, we derive some properties that the optimal solution satisfies. On the basis of these properties, we offer another single-point algorithm that is more easily understandable and more straightforward than that of Tantawi and Towsley [5]. Furthermore, we identify properties relating to the convergence of our algorithm and demonstrate its performance.

Manuscript received May 9, 1989; revised August 26, 1989.
C. Kim is with the Department of Computer Engineering, College of Engineering, Yeungnam University, Gyongsan, 712-749 Korea.
H. Kameda is with the Department of Computer Science, and Information Mathematics, The University of Electro-Communications, Chofu-shi, Tokyo 182, Japan.
IEEE Log Number 9102593.

II. NOTATION AND ASSUMPTIONS

We assume the same model as that of Tantawi and Towsley [5]. That is, the system consists of n nodes (hosts) connected by a communications network. For reference, we repeat a portion of the notation and assumptions contained in [5] here (see Appendix C of [5]).

- β_i Job processing rate (load) at node i.
- $\boldsymbol{\beta}$ $[\beta_1, \beta_2, \cdots, \beta_n]$.
- ϕ_i External job arrival rate to node i.
- Φ Total external arrival rate ($\Phi = \sum_{i=1}^{n} \phi_i$).
- λ Network traffic.
- F_i Mean node delay of a job processed at node i—an increasing positive function.
- G Source-destination-independent mean communication delay–a nondecreasing positive function.

III. PROPERTIES OF THE OPTIMAL SOLUTION AND AN OPTIMAL LOAD BALANCING ALGORITHM

The problem of minimizing the mean response time of a job is expressed in the following formulations, as stated by Tantawi and Towsley [5].

$$\text{minimize} \quad D(\boldsymbol{\beta}) = \frac{1}{\Phi}\left[\sum_{i=1}^{n} \beta_i F_i(\beta_i) + \lambda G(\lambda)\right], \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} \beta_i = \Phi,$$
$$\beta_i \geq 0, \qquad i = 1, 2, \cdots, n$$

where the network traffic λ may be expressed in terms of the variable β_i as

$$\lambda = \frac{1}{2} \sum_{i=1}^{n} |\phi_i - \beta_i|. \tag{2}$$

Define the following two functions.

$$f_i(\beta_i) = \frac{\partial}{\partial \beta_i}(\beta_i F_i(\beta_i)),$$
$$g(\lambda) = \frac{\partial}{\partial \lambda}(\lambda G(\lambda)).$$

Tantawi and Towsley [5] derived the following theorem by using the Kuhn–Tucker theorem (see Theorem 2 of [5]):

[Tantawi-Towsley Theorem]: The optimal solution to problem (1) satisfies the relations

$$\begin{array}{lll} f_i(\beta_i) \geq \alpha + g(\lambda), & \beta_i = 0 & (i \in R_d), \\ f_i(\beta_i) = \alpha + g(\lambda), & 0 < \beta_i < \phi_i & (i \in R_a), \\ \alpha \leq f_i(\beta_i) \leq \alpha + g(\lambda), & \beta_i = \phi_i & (i \in N), \\ \alpha = f_i(\beta_i), & \beta_i > \phi_i & (i \in S), \end{array} \tag{3}$$

subject to the total flow constraint

$$\sum_{i \in R_a} f_i^{-1}(\alpha + g(\lambda)) + \sum_{i \in N} \phi_i + \sum_{i \in S} f_i^{-1}(\alpha) = \Phi \tag{4}$$

where α is the Lagrange multiplier.

Tantawi and Towsley's single-point algorithm [5] first determines the node partitions (see steps 2–5 [5]). Then it solves (4) for α, and