Short Note

Parallel Address Setting Schemes for Complete-binary-tree-connected Machines

HIROSHI UMEO*

The concept of processor address in parallel computer architectures sometimes plays an important role in the design of SIMD-type parallel algorithms. In this paper we consider an address setting problem on complete-binary-tree-connected cellular computers, consisting of many identical processors each with its own address. The problem is to design a parallel algorithm which can assign pre-specified binary address to each processor. Several time-efficient bit-wise address setting schemes with $O(\log_2 n)$ time complexities are developed for the complete-binary-tree-connected machines having $O(n)$ processors.

1. Introduction

Much attention has been paid to the study of tree-structured parallel processors[3], [5], [6], [9]. In this paper we consider an address setting problem on complete-binary-tree-structured cellular computers, consisting of many identical processors each with its own address. The problem is to design a parallel algorithm which can assign pre-specified binary address to each processor[8]. The concept of processor address in the parallel computer architectures sometimes plays an important role in the design of SIMD-type parallel algorithms. The processor address may be used as an attached tag for the complicated data routing and communications between processors. The problem was originally proposed in the study of conversion of parallel algorithms from the SIMD-type to the MISD[1] and several time-optimum parallel binary address setting schemes were established for linear and two-dimensional uniformly-interconnected array processors[8]. This problem will be important for the future general-purpose VLSI-based reconfigurable[9], [2], [4] and/or programmable cellular computers [7] with fault-tolerance in addition to being interesting in its own right from a theoretical point of view.

The organization of this paper is as follows. Section 2 defines the address setting problem for tree machines and in section 3 we develop several time-efficient bit-wise address setting schemes for the complete-binary-tree-connected machines. The conclusion is presented in the last section.

*Dept. of Applied Electronic Engineering, Faculty of Engineering, Osaka Electro-Communication Univ. Hatsu-cho, 18-8, Neyagawa-shi, Osaka, 572, Japan

2. Address Setting Problem on Tree Machines

Consider a complete-binary-tree-connected machine M, shown in Fig. 1, which consists of identical $(n/2-1)$ node processors and $n/2$ leaf processor, where $n=2^m$ for some positive integer m. For convenience we refer to it as depth $[\log_2 n]$ tree, where $[x]$ denotes the ceiling of x. We assume that the level of leaf processors is zero. In one step each node processor can communicate with its father-processor and two son-processors, that is, one right and one left descendant. The root processor has the additional responsibility of being the input/output port of M. The leaf processor can communicate only with its father-processor. Each processor is composed of a finite number of finite registers and an address register Ra.

At time $t=0$ the host computer supplies the root processor with an activating signal. This signal propagates

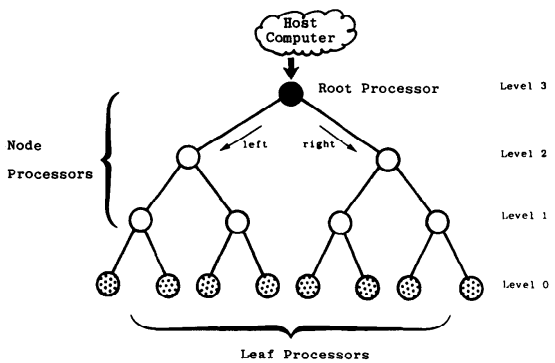


Fig. 1 A complete-binary-tree-connected machine.

down to the binary-tree-structured network at a unit speed, that is, 1-level/1-step. Each processor is in a quiescent state until the activating signal reaches the processor. A local transition function, which is common to all processors, determines the next state of every processor from its local informations.

The address setting problem on M is to design the local transition function of M so that the address register Ra in every processor contains a unique address represented in a binary form according to the prespecified node-ordering system given below. We consider the following tree traversal ordering systems [10]:

(1) Breadth-first order, (2) Preorder, (3) Postorder, and (4) Inorder. See Fig. 2(a), (b), (c), and (d).

Let M be any complete-binary-tree-machine of depth $[\log_2 n]$ and u be any node processor with a right son t and a left son s. We assume that the level of node u of M is j, where $1 \leqq j < [\log_2 n]$. For each ordering, the address of s, t and the root of M are determined by the following equations, where $A(v)$ denotes the address of node v. They are easily obtained from the definitions.

$$\text{Breadth-first-order:} \begin{cases} A(\text{root})=1, \\ A(s)=2A(u), \\ A(t)=2A(u)+1. \end{cases} \quad \cdots(1)$$

$$\text{Preorder:} \begin{cases} A(\text{root})=1, \\ A(s)=A(u)+1, \\ A(t)=A(u)+2^j. \end{cases} \quad \cdots(2)$$

$$\text{Postorder:} \begin{cases} A(\text{root})=n-1, \\ A(s)=A(u)-2^j, \\ A(t)=A(u)-1. \end{cases} \quad \cdots(3)$$

$$\text{Inorder:} \begin{cases} A(\text{root})=n/2, \\ A(s)=A(u)-2^{j-1}, \\ A(t)=A(u)+2^{j-1}. \end{cases} \quad \cdots(4)$$

3. Time-Efficient Address Setting Schemes for Tree Machines

First we will develop a breadth-first address setting scheme.

[Theorem 1] There exists a time-optimum breadth-first address setting scheme which sets up all addresses of depth $[\log_2 n]$ complete-binary-tree-connected processors in exactly $2[\log_2 n]-1$ steps.

(Proof) The addresses are set from the highest digit in order on each node. In Fig. 3 we give a description of the processor and its operation performed by each processor at every cycle. Each processor has three data paths, a_{in}, l_{out}, and r_{out}, each of which is 3-bits wide. We assume that the root processor has also a_{in}. Initially, at time $t=0$, a_{in} of the root processor has 1^* and other a_{in}'s are ϕ.

A processor operates as follows: The data path a_{in} has a value of either ϕ, s, or s^*, where $s \in \{0, 1\}$. While a_{in} is ϕ, the processor does nothing. While a_{in} is s, the processor repeats the Part A in Fig. 3 in each cycle. When $a_{in}=s^*$, after performing the Part A and B once, respectively, the processor will complete the address setting operation. Note that * mark is deleted in Part A. The validity of our setting scheme is obtained by the following two observations.

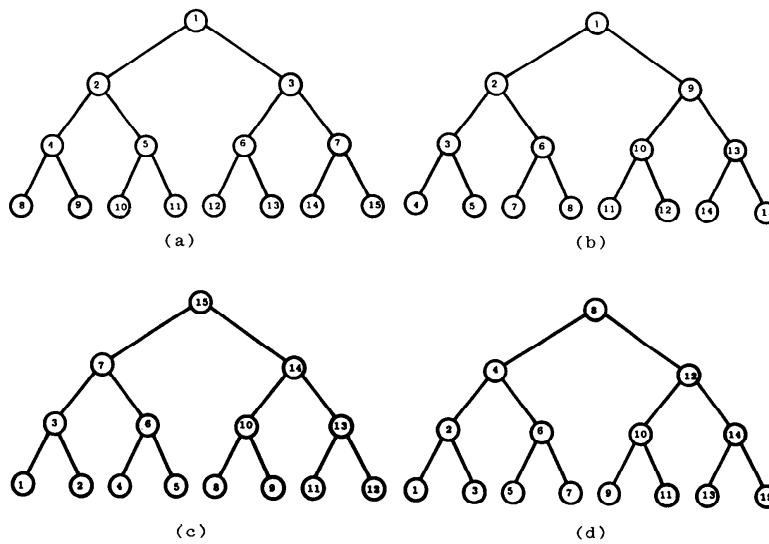
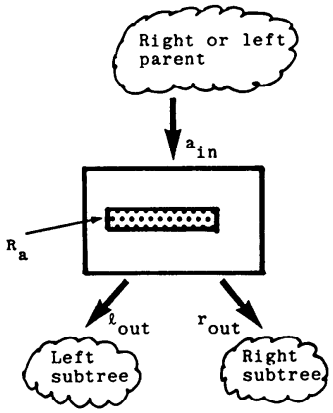


Fig. 2 (1) Breadth-first order, (b) Preorder, (c) Postorder, and (3) Inoreder.



/* a_{in} has either ϕ, s, or s^*, where s is 0 or 1*/
while ($a_{in} \neq \phi$) do {
- The processor sets the next highest digit to s.
- $r_{out}=s$, $l_{out}=s$
- if ($a_{in}=s^*$) break

} Part A

- $l_{out}=0^*$, $r_{out}=1^*$

/* The address setting operations are completed */ } Part B

Fig. 3 Processor description for breadth-first addressing.

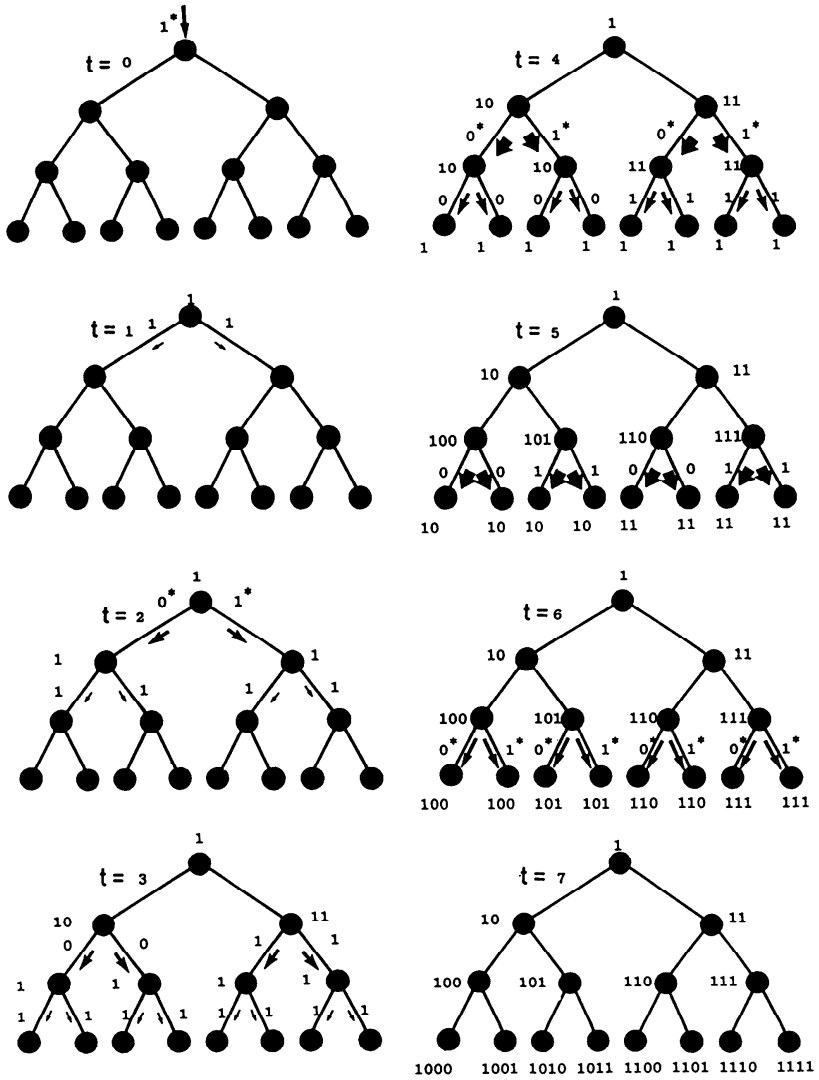


Fig. 4 A snapshot of the breadth-first-addressing scheme from time $t=0$ to 7 (in the case where $n=16$).

(I) Let u be any node which has a left and right descendant s and t, respectively. Then, in the breadth-first addressing, s and t have a binary address $\alpha \cdot 0$ and $\alpha \cdot 1$, based on the eq. (1) given above, respectively, where α is the binary representation of the address of node u and "$\cdot$" denotes a symbol concatenation.

(II) All the nodes of level $[\log_2 n]-j$ have j-digit binary addresses, where $0 \leqq j < [\log_2 n]$.

It is shown that the node processors of level $[\log_2 n]-k$ complete the address setting operations at time $t=2k-1$ where $1 \leqq k \leqq [\log_2 n]$. At time $t=2[\log_2 n]-1$ all addresses will be set. Fig. 4 shows a snapshot of the scheme from time $t=0$ to 7 where $n=16$. ■

[Theorem 2] There exists a preorder, postorder and in-order address setting scheme each of which can set up all addresses of depth $[\log_2 n]$ complete-binary-tree-connected processors in $4[\log_2 n]+O(1)$[†] steps, respectively.

(Proof sketch) We give a scheme for the preorder addressing. A similar technique can be employed for other addressings, so we will omit them. See eq. (2). We refer the terms $+1$, and $+2^j$ as an offset address for s and t, respectively. From the observation below it is easily seen that, within $2[\log_2 n]+O(1)$ steps, every node can set up its offset address, shown in Fig. 5 in the case where $n=16$.

(I) Each node knows whether it is a right or left descendant.

(II) Each node can calculate its height(level) by counting the steps that the down-going wave, generated at the root at time $t=0$, requires to bounce off at the leaf.

We see that any complete-binary-tree of depth $[\log_2 n]$ consists of $n/2$ linear arrays beginning at the root and ending at the leaf. Within the next $2[\log_2 n]+O(1)$ steps, each node can set up its complete address by applying our previous one-way pipeline addressing scheme developed for linear arrays[8]. The scheme[8] says that it requires $(n-1+[\log_2 n])$ steps to assign binary addresses to all n cells. But in our case the length of the array is $[\log_2 n]$, so additional $2[\log_2 n]+O(1)$ steps are sufficient for the complete addressing. Incorporation of the offset address into the final address is easily accomplished by modifying the scheme[8]. The details of them are omitted. ■

[†]To give more generality we add a term O(1).

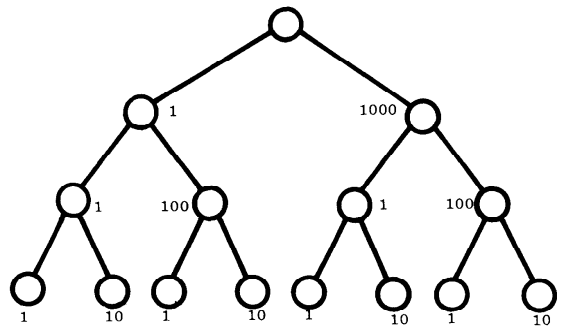


Fig. 5 Offset address for the preorder addressing.

4. Conclusion

We have developed several time-efficient bit-wise parallel address setting schemes for complete-binary-tree-connected machines. Our schemes can be applied to the incomplete-binary-tree machines, but its assigned addresses are not successive integers. An $O(\log_2 n)$-step parallel algorithm for assigning successive address to all nodes in incomplete-binary-tree is left open at present.

Acknowlegement

The author would like to thank the referee who indicated some mistakes in the first version. A part of this work was supported by the grant (No. 61750346) of Ministry of Education, Culture and Science of Japan.

References

1. UMEO, H. A class of SIMD algorithms implemented on systolic VLSI arrays, *IEEE Proc.* of the 1984 *International Conference on Parallel Proc.* (1984), 374-376.
2. ROSENFELD, A. and WU, A. Y. Reconfigurable cellular computers, *Inf. and Contr.*, 50, (1981), 64-84.
3. ULLMAN, J. D. Computational aspects of VLSI, *Computer Science Press*, Maryland (1984).
4. SNYDER, L. Overview of the CHiP computer, in VLSI 81, (1981), 237-246.
5. BHATT, S. N. How to assemble tree machines, *ACM Proc.* of the Foundations of the Computer Science, (1982), 77-84.
6. TAKAHASI, Y., WAKABAYASHI, N. and NOBUTOMO, Y. A binary tree multiprocessor: CORAL, *Journal of Information Processing*, 3, 4, (1981), 230-237.
7. FISHER, A. L., KUNG, H. T., MONIER, L. M., and DOHI, Y. The architecture of a programmable systolic chip, *J. of VLSI* and *Computer Systems*, 1, 2, (1984) 153-169.
8. UMEO, H. Time-optimum parallel binary address setting algorithms for cellular computers, *Trans. IPS*, 25, 1, (1984) 109-115.
9. SNYDER, N. Introduction to the configurable highly parallel computer, *IEEE Computer*, 1, 1, Jan., (1982) 47-56.
10. AHO, A. V., HOPCROFT J. E. and ULLMAN, J. D. The design and analysis of computer algorithms, Addison-Wesley, Reading, Massachusetts, (1974).

(Received September 10, 1986; revised November 25, 1986)