PALM시스템의 구조와 네트웍 성능

김 석 일[†]

요 약

본 논문에서는 HCH(m, p)에 기반을 둔 PALM 시스템의 구조와 네트웍의 성능을 연구하였다. HCH(m, p)는 하나의 CP를 중심으로 p개의 AP를 연결하여 클러스터를 구성하고, 클러스터를 $m-p$ 차원의 하이퍼큐브로 연결한 소결합 다중프로세서 시스템이다. 본 시스템에서는 AP와 CP 및 CP와 CP간을 DPR로 연결하여 워드단위의 통신이 가능하도록 구성하여 빠르고 안정된 메세지 전송을 가능하게 하였다. PALM 시스템에 사용된 네트웍은 여러가지 HCH 네트웍 중에서 시스템에 포함되는 AP의 갯수가 최대이나 CP 및 링크(또는 DPR)의 합이 최소인 최적 HCH(m, 2) 네트웍이다. 본 논문에서는 HCH(2, 2)인 실험시스템을 구성하고 네트웍의 통신성능을 측정하였으며, PALM시스템의 작은 통신/연산비는 메세지 통신형 다중프로세서 시스템에서도 화인 그레인(fine grain) 병렬성을 다룰 수 있음을 보인다.

The PALM system : Architecture and Network Performance

Sukil KIM[†]

ABSTRACT

This paper introduces the *Parallel Advanced Loosely coupled Multiprocessor* (PALM) architecture, which is based on HCH(m, p), where m is the number of links per a communication processor (CP) and p is the number of application processors (APs) connected to the CP. Communication links between a pair of CPs and/or between a CP and an AP, are made of dual-Port RAMs, which provide fast and reliable *word-parallel* communication between processors. Among the wide spectrum of HCH networks, HCH(m, 2) is also known to be a cost optimal topology, such that HCH(m, 2) consists of the largest number of APs retaining the minimal number of CPs and communication links. We also implement a testbed based on HCH(2, 2). The experiment result shows that the small communication/computation ratio of the PALM system would realize fine-grain parallelism on message-passing MIMD systems.

1. Introduction

Among the various interconnection topologies (i.e., ring, hypercubes, torus, trees, meshes, etc.), hypercubes have been widely used as an interconnection network since their routing scheme is relatively simple and their topological emulation capability is extensive. So far, two types of hypercubes have emerged. One incorporates *packet switching* techniques, as seen in the first generation hypercubes such as the iPSC [1] and the Cosmic Cube [2]. Such systems are known to have the poor communication/computation ratio, typically not better than 20 [18]. The other type facilitates *circuit switching* techniques, as seen in the nCUBE/2 [3] and the iPSC/2 [4], that are known to be the second generation hypercubes. Recently, *wormhole routing* is widely studied and adopted as a basic interprocessor communication strategy in message passing MIMD systems. Such

[†] 정 회 원 : 충북대학교 컴퓨터과학과 교수

논문접수 : 1994년 2월 25인, 심사완료 : 1994년 4월 25일

systems usually incorporate a *dual-processor* node configuration in which each node composes a communication processor (CP) and an application processor (AP). Thus, a dual-node configuration promises concurrent interprocessor communications and computations [5].

The degree of parallelism reflects the extent to which software parallelism matches hardware parallelism. Allocation of closely related parallelism onto a cluster that closely connects a small number of processors through a high performance intra-cluster network would provide better performance than that in the even parallelism distribution [6-9]. Such clustering processors would reduce the communication bandwidth on a global interconnection network, so that a small number of CPs is sufficient to service the communication demands of a large number of APs. A good example can be found in the Connection Machine [10], in which 16 APs are grouped and controlled by one CP. In this paper we design a cluster based hierarchical network, called a *Hierarchical Cluster based Hypercube* (HCH). This network is adopted for the *Parallel Advanced Loosely coupled Multiprocessor* (PALM) system, that is under construction at Chungbuk National University.

In Section 2, a fundamental HCH network is introduced. The HCH network consists of a number of hypercube nodes. Every node configures one CP and a small number of APs connected to the CP. The network features are presented in Section 3. In Section 4, we implement a testbed to evaluate the communication performance of the network. Section 5 represents conclusions and describes further works in progress.

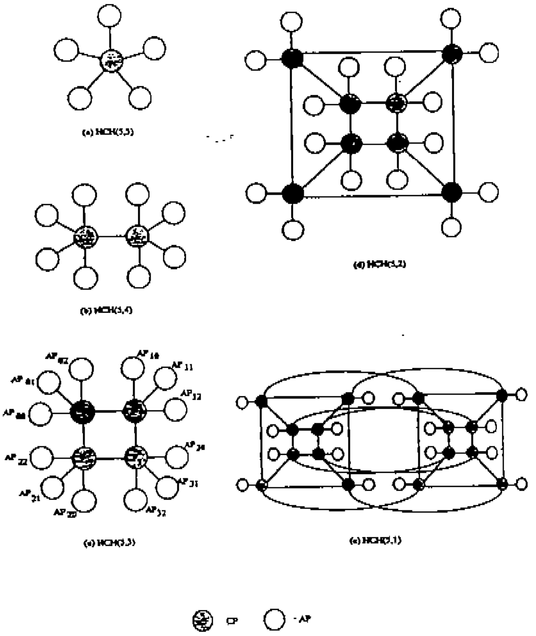
2. Hierarchical Cluster based Hypercube Networks

The basic network topology considered for the PALM system is a hierarchical network, in which p APs are connected to every CP to form a cluster and CPs are interconnected as a hypercube. Such a hierarchical cluster based hypercube is called HCH(m, p), where m is the number of links every CP has and p is the number of links to connect APs. Thus, $m-p$ links shape an $m-p$ dimensional cube network. We summarize the characteristics of HCH(m, p) as follows:

1. Each CP has m links; p links for inter-cluster communication and $m-p$ links for intra-cluster communication.
2. CPs are interconnected as an $m-p$ dimensional cube.
3. The network has 2^{m-p} clusters (or CPs), $p\times 2^{m-p}$ APs and $(m+p)2^{m-p-1}$ links.
4. The diameter of the network is $m-p$.

HCH(m, p) can represent a wide variety of network topologies. For examples, HCH(m, 1) is an (m-1) dimensional second-generation hypercube; HCH(m, m) is a star network in which m APs are connected to one CP. Fig. 1 depicts several types of networks based on HCH(5, p). Fig 1(a) is a star network having 5 APs, while Fig 1(e) is a 4-dimensional cube network.

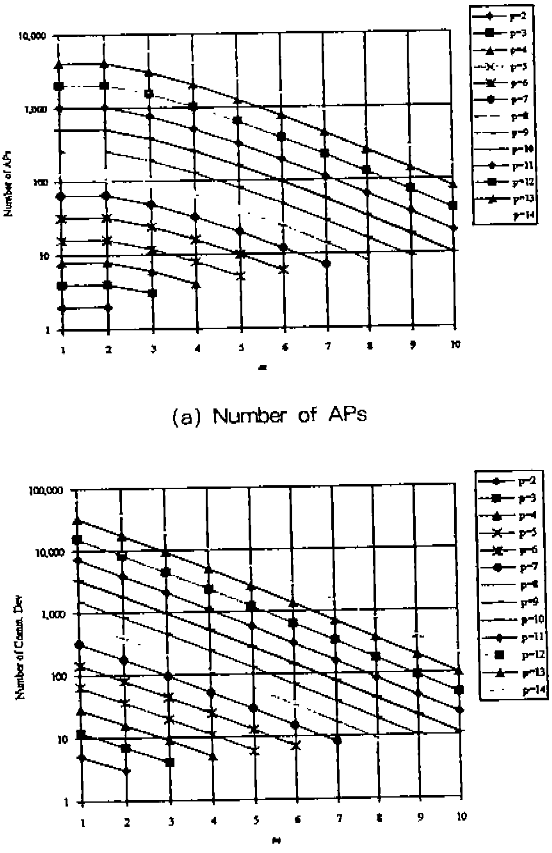
Let the address of an AP be represented by AP_{cq}, where c is the index of a cluster and q is the index of the AP within the cluster c. Communication between a pair of APs whose cluster indices are identical does not incur any inter-cluster communication. How-



(Fig. 1) HCH(5, p) networks.

ever, communication between two APs, that reside in the different clusters, initiates multihop inter-cluster communications along the $m-p$ dimensional cube network. For an example, a message passing between AP_{00} and AP_{32} in Fig. 1(c), requires inter-cluster routing through CP_0, CP_1 and CP_3. Intrinsically, such an inter-cluster communication routing is identical to that in a hypercube, and thus, a hypercube self-routing is adopted for our routing strategy. We will revisit this topic in the next section.

Among several possible combinations of m and p, either HCH(m, 1) or HCH(m, 2) consists of the maximal number of APs [11]. However, HCH(m, 2) requires smaller number of communication devices (total number of CPs and links in our system) than that required in HCH(m, 1). This implies that HCH(m, 2) has the maximal number of APs while it retains a construction cost low. Such a design consideration is crucial, since a word-parallel communication between processors, that is our design goal, would dramatically increase a wiring complexity of the system as the dimension of the inter-cluster communication network increases. Fig. 2 shows the number of APs and the number of communication devices varying m and p. As expected, HCH(m, 2) requires less communication devices than that required in HCH(m, p), such that $p \neq 2$. In this paper, we denote HCH(m, 2) as an optimal HCH network.



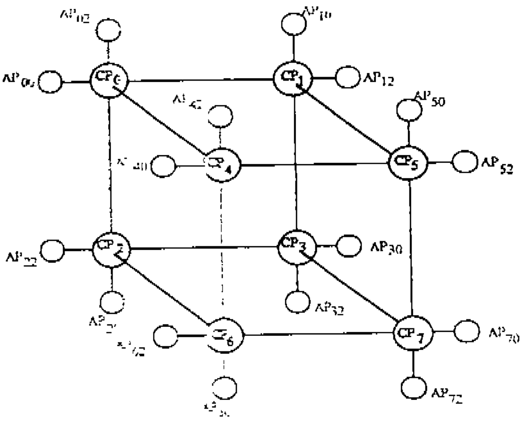
(a) Number of APs

(b) Number of communication devices.

(Fig. 2) The number of communication devices to build HCH(m, p) varying m and p.

3. The PALM System Network

We designed a PALM system based on HCH(m, 2). Each CP is interconnected to two APs, and all CPs are linked as an (m−2) dimensional cube. Fig. 3 shows HCH(5, 2) which is a basic network topology of the PALM system. It has 16 APs and 8 CPs. Every link represents a DPR (dual-port RAM), whose memory space can be accessed directly from both ports. As DPRs facilitate a primitive address-contention prevention logic inside the chip, construction of a reliable word-parallel communication network using DPRs is simple. Furthermore, a word-parallel communication between processors would improve the communication bandwidth of the system compared with a bit-serial communication as seen in most of the loosely coupled MIMD systems [14].



(Fig. 3) The PALM system network based on HCH (5, 2).

Label the network component according to the definition of HCH(m, p) as discussed in section 2. Then, CP_0 is interconnected to CP_1, CP_2 and CP_4 through DPRs, $M_{CP_0-CP_1}$, $M_{CP_0-CP_2}$ and $M_{CP_0-CP_4}$, respectively. AP_{i0} and AP_{i1} are interconnected to CP_i through $M_{AP_{ij}-CP_i}$ such that j=0 or 1. By using the labeling convention, we can prepare an obvious and straightforward routing algorithm for the system. Let AP_{si} and AP_{dj} be addresses of a source AP and a destination AP, respectively. AP_{si} writes a message packet that contains the source and destination AP addresses and necessary information onto the corresponding location of the DPR, $M_{AP_{si}-CP_s}$. Then CP_s fetches the source and destination addresses from the DPR, and determines the inter-cluster routing path from the cluster s to the cluster d.

Let CP_s, CP_k, CP_l, ⋯, CP_n, CP_d, be a routing path between CP_s and CP_d. CP_s reads a message packet from $M_{AP_{si}-CP_s}$ and moves the message packet into $M_{CP_s-CP_k}$, and then, CP_k reads a message packet from $M_{CP_s-CP_k}$ and moves it into $M_{CP_k-CP_l}$, and so on, until the message packet arrives at CP_d. Upon CP_d is

```
Input: Source AP address, ns and destination AP address, nd;
Output: Inter-cluster message passing sequence;

/* Let msg be a message packet from $AP_{ns}$ to $AP_{nd}$, such that ns = [s, i]
   and nd = [d, j]. And let DPR notify CP or AP to signal the message packet arrival. */

while true do
    for every $CP_k$, $0 \le k \le m-1$ do
        if $CP_k$ get msg then
            nd = get_destination_address (msg);
            /* Let r be the location of the first '1' bit in $k \oplus d$ */
            if r = 0 then /* $CP_k$ is the cluster address that contains
                           the destination AP */
                Copy msg to the corresponding location of the r-th DPR;
            else
                Copy msg to the corresponding space of the (r+j)-th DPR;
            end if;
            Remove msg from the message buffer
        end if
    end for
end while
```

(Fig. 4) Message routing algorithm.

notified that a message packet has arrived at $M_{CP_i-CP_d}$, CP_d moves the message packet to $M_{CP_d-AP_{d_j}}$. Then, AP_{d_j} can access the message packet for its own use. We summarize the routing algorithm in Fig. 4. The copy instruction moves a message packet from one DPR to the other DPR. Thus, the instruction requires one block-move operation.

4. Tiny PALM and Performance Experiments

We implement a testbed based on the proposed network. The testbed consists of a single cluster as shown in Fig. 5. Two APs are connected to a CP, that is also connected to the host computer. The host computer downloads application codes onto corresponding APs through the CP, and it receives data from the CP. As the testbed has a pair of APs and a CP, it can be used as a building block to construct more complex systems. For an example, eight building blocks and 4 DPRs are sufficient to construct a 16-AP PALM system. We denote such a building block as the Tiny PALM.

APs and CP of the testbed are a complete computer system. It consists of the iPAX80286 processor running at 18MHz clock frequency, 1Mbyte long private memories, a system ROM, 3-DPRs and an associated logic and a CRT display. The CRT display shows the system status of the processor (either AP or CP). It also reports error messages during communication and computation. The host computer is a PC/386 running under the Minix operating system [12].

We put single-chip 1 Kbyte DPRs (AMD 2130-10DC) between all pairs of neighbor processors. The read and/or write cycle time of the DPR is 100 nsec. When two processors are accessing the same DPR memory space at a time, DPRs generate a busy-wait signal and notify one processor that a memory space is being accessed by the other processor. This signal is fed into a contention-resolution logic that delays several CPU clocks until the other processor finished accessing the memory space. Details can be found in [13].

A) Message passing protocol

The address space of 1 Kbyte DPR is divided into two 512-byte blocks for bidirectional communication. Each block allocates 16 bytes for a message header and 496 bytes for a message buffer. The message header stores a control information for the message block, such as a semaphore, a source AP address, a destination AP address, and others. Fig. 6 shows the contents of two 512-byte blocks. The first byte of each block represents a semaphore that provides a synchronization between processors.

In the PALM system, processors writing a message packet onto a DPR is allowed only when the semaphore is set to 0, and the message packet can be read when the semaphore is set to 1. Thus, processors writing a message packet should wait until the semaphore is 0, and processors that complete writing the message packet must set the semaphore to 1 to notify the other processor that the message is ready to read. Similarly, a processor reading a message packet should wait until the semaphore is set to 1. After reading the message packet, the processor releases the semaphore to be reused in the next round.

This simple semaphore operation provides a correct communication protocol[11].

0	Semaphore/command
1	source AP
2	destination AP
3-4	Message no.
5-6	Message length
7-8	Frame no.
9-a	Process ID
b-f	reserved space
10-1ff	actual message buffer

(a) Send

200	Semaphore/command
201	source AP
202	destination AP
203-204	Message no.
205-206	Message length
207-208	Frame no.
209-20a	Process ID
20b-20f	reserved space
210-3ff	actual message buffer

(b) Receive

(Fig. 6) DPR memory map.

B) System Programs

We implement a library for the testbed operable under the Minix OS [12] and under the MS-DOS environment. Table 1 lists the major system function calls. These are particularly crucial for process synchronization and communication.

(Table 1) System function calls.

System function name	function
creat_list(n)	creates *n* word message buffer message packet pool
send(c_q, msg, msg_no, msg_len, pid)	sends message to AP_{id}
receive(msg, msg_no, msg_len, pid)	receives message and restores them into *msg*
my_node()	returns the current AP identifier
my_cluster()	returns the current cluster identifier
load(file_name, pid)	downloads *file_name* into every AP with process name *pid*

C) Experiments

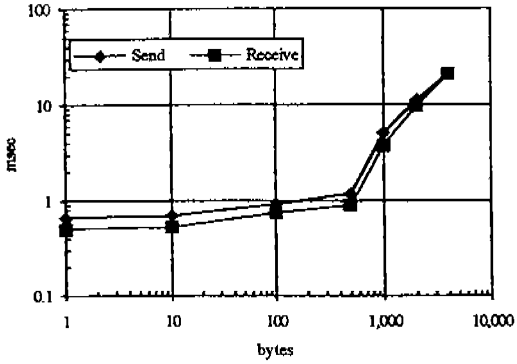
We approximate a communication latency time as a function of the message length and the number of hops as shown in Eq. (1).

$$t(h, m) = t(\alpha + m\beta) \qquad (1)$$

where h is the number of hops for a message packet to be moved, m is the message length in bytes. α is a routing set-up time, and β is a communication time per unit message. In the PALM system, the number of hops for communication is identical to the number of move instructions. By such an observation, experiments on the testbed are considered as a single-hop message transfer. As we use the tiny PALM as a building block to construct the full-scale development, we can also calculate the multihop communication time as Eq. (1).

By varying the message length along with $h=1$, we measure the communication time to transfer messages between two APs. Fig. 7 shows the average communication time with respect to the message length. We can observe that the communication time is proportional to the message length. If a message is smaller than the single frame ($m \leq 496$), the transfer of the single frame would complete the communication. We classify this type of communication as a single-frame message passing. Since the single-frame message passing does not encounter any process for the creation and restoration of frames, it requires no overhead for the process. In Fig. 7, the timing experiment also verifies that the single-frame message passing requires less overhead than that required in longer message transfers. If a message is longer than the single frame ($m>496$), the message has to be partitioned into several frames, so that every frame is transferred consecutively. Thus, the message transfer encounters processes to synchronize and to restore the unordered frames received, so it requires an overhead for the process. We call this as a multiframe message passing. Our test shows that the multiframe message passing requires twice as much time as the single-frame message passing does. We summarize the communication parameter of Eq(1) in Table 2.

We compare the communication time to transfer a 100-byte message with those of three multicomputer generations [17]. The results show that the single-hop communication time to transfer a 100-byte message in the tiny PALM is 0.75 msec and 0.92 msec based on the single-frame and multiframe message passing, respectively. These values are far smaller than 5 msec, that is a typical value on the second generation multi-computers as shown in [17]. This implies that intra-cluster communication would be sufficiently fast for parallel computations.



(Fig. 7) Communication times between a pair of APs

We also calculate the communication/computation ratio. We measure the computation time by performing 1,000 floating-point multiplications on an AP. The result shows 160.5 msec per multiplication on the iPAX80286 processor running at 18MHz clock frequency. The communication/computation ratio is 0.127, which is far lower than 20 that is a typical value of the early-generation message passing MIMD systems [18].

(Table 2) Parameters for the communicationtime estimation based on two message-passing protocols type of protocol

Type of	Sending Messages		Receiving Messages	
protocol	α(μsec)	β(μsec)	α(μsec)	β(μsec)
single frame	670	2.44	510	2.42
multi-frame	670	5.33	510	5.35

With significant improvement in the communication performance, our system will be called a *fine-grain MIMD system*. We believe that our system may increase potential parallelism on message passing MIMD systems as we can see in [15] and [16]. Such a feature would be originated by using DPRs, that provide word-parallel communication and fast point-to-point message passing.

5. Conclusions

We designed a PALM system which has hierarchically clustered hypercube architecture. The system is based on HCH(m, p), where m is the number of links per a communication processor (CP) and p is the number of application processors (APs) connected to the CP. Among several HCH topologies, HCH(m, 2) is an optimal network, such that the optimal network consists of the largest number of APs retaining the minimal number of CPs and communication links (or DPRs).

We implement a testbed based on HCH(2, 2) called a Tiny PALM for an experimental purpose. System function calls for communication and synchronization are also implemented. The test results show that proposed communication algorithms and protocols for the testbed are applicable. It also shows that the communication/computation ratio is very small, so that the system would promise the capability of fine-grain parallel processing. We are currently implementing a PALM system with 16 APs and 8 CPs. Also proposed system function calls are being extended for the PALM system. Several fine-grain applications are soon revealed.

References

[1] Intel Scientific Computers, *iPSC System Overview Manual*.

[2] C. L. Seitz, "The Cosmic cube," *Comm. ACM*, Vol. 28, No. 1, pp. 22-33, Jan. 1985.

[3] J. P. Hayes, T. N. Mudge, Q. F. Stout, S. Colley, and J. Parmer, "Architecture of a hypercube supercomputer," *Proc. 1986 Int. Conf. Parallel Processing*, 1986, pp.653-660.

[4] C. L. Seitz, "iPSC/2: a second generation hypercube," *Proc. 3rd . Hypercube Concurrent Computers and Applications*, Pasadena CA, Jan. 1988.

[5] D. K. Bradley, *First and Second Generation Hypercube Performance*, Tech Rpt. UIUCDCS-R-88-1455, Univ. Illinois at Urbana-Champaign, Sept., 1988.

[6] S. P. Dandamudi, "A performance comparison of routing algorithm for hierarchical hypercube multiprocessor networks," *Proc. 1989 Int. Conf. Parallel Processing*, Vol. 1, pp281-285, Aug. 1990.

[7] S. P. Dandamudi and D. L. Eager, "Hierarchical interconnection networks for multicomputer systems," *IEEE Trans. Computers*, Vol. C-39, No. 6, June 1990.

[8] K. Ghose and K. R. Desai, "The HCH: A versatile interconnection network based cubes," *Proc. 1989 Int. Conf. Supercomputing*, pp. 426-435, 1989.

[9] K. Ghose and K. R. Desai, "The design and evaluation of the hierarchical cubic network," *Proc. 1990 Int. Conf. Parallel Processing*, Vol. 1, pp355-362, Aug. 1990.

[10] W. D. Hillis, *The Connection Machine*, MIT Press, Cambridge MA, 1985.

[11] Y.-S. Lee and Sukil Kim, "PALM system design and it's performance test, "*Proc. Parallel Processing System*, KPPS, Vol. 3, No. 2, pp218-232, Korea, Oct. 1992.

[12] A. S. Tanenbaum, *Operating Systems: Design and Implementation*, Hall Inc., 1987.

[13] N. Jagadish, J. M. Kumar, and L. M. Patnaik, "An efficient scheme for interprocessor communication using dual-port RAMs," *IEEE Macro*, Vol.9, No.10, pp10-19, Oct. 1989.

[14] Y.-S. Lee and Sukil Kim, "The PALM System: A loosely coupled hierarchical multiprocessor system," *Proc. TENCON93 /Beijing*, pp242-245, Oct. 1993.

[15] J. Mauney, D.P. Agrawal, E. Harcourt, Sukil Kim, etc., "Computational Models and Resource Allocation for Supercomputers," *Proceedings of IEEE*, Vol. 77, No. 12, pp1859-1874, Dec. 1989.

[16] Sukil Kim, S.S. Pande, D.P. Agrawal and J. Mauney, "A Message segmentation technique to minimize task completion time," *Proc. 5th Parallel Proc. Symp.*, pp519-524, 1991.

[17] W. C. Athas and C. L. Seitz, " Multicomputers: Message-passing concurrent computers," *IEEE Computer*, Vol. 21, No.8, pp9-24, Aug. 1988, and also

Kai Hwang, *Advanced Computer Architecture: Parallelism, Scalability and Programmability*, p370, McGraw-Hill, N. Y. 1993.

[18] Y. Shin and J. Fier, "Hypercube systems and key applications," *Parallel Processing for Supercomputers and Artificial Intelligence*, Ed. Kai Hwang and D. Degroot, McGraw-Hill, N.Y. 1989.

김 석 일

1975년 서울대학교 전기공학과 졸업 (학사)

1984년 연세대학교 전자계산학 (공학석사)

1989년 North Carolina State University 전기 및 컴퓨터공학 (공학박사)

1975년~1990년 국방과학연구소 연구원, 선임연구원

1990년~현재 충북대학교 자연과학대학 컴퓨터과학과 조교수

관심분야 : 병렬처리컴퓨터 구조, 병렬처리알고리즘, Super-computing, Advanced factory automation 등.