The SKB: A Semi-Completely-Connected Bus for On-Chip Systems

Masaru Takesue

Dept. Electronics and Information Engr., Hosei University, Tokyo 184-8584 Japan
takesue@ami.ei.hosei.ac.jp

Abstract. This paper proposes a semi-completely-connected bus, called *SKB*, to alleviate the long-wire and pin-neck problems against on-chip systems through a small diameter and dynamic clustering. Dynamic clustering allows to reduce the traffic to the per-cluster units such as the global interconnect interface, as compared with the static clustering fixed in hardware. We derive a 2^n-node *semi-complete (SK) graph* from a simple node-partitioning. An SKB is produced from the SK graph when we replace the links incident to a node by a single bus for the node. The diameter of SKB equals 1 (bus step), though the bus length is rather long, $\mathbf{O}(\sqrt{2^n})$. Simulation results show that relative to the hypercube with the link delay of 1 clock, the SKB's bandwidth is about 0.97 and 0.14 assuming the bus delay of 1 and 8 clocks, respectively, that increases to about 4.57 and 0.71 with the dynamic clustering.

1 Introduction

With future LSI technologies, we will be able to put a large portion of a system in a single chip. However, the technologies have three problems above all: First, the signal delay due to long wires will dominate the clock cycle time of the system if the feature size of wires will scale with the same pace as for transistors [1].

Second, the design of a system on a chip (SoC) may be restricted by the power consumable in the chip. In the current high-performance microprocessors, interconnect power is over 50% of the total dynamic power, and its about 50% is consumed by global wires [2]. So a large-scale network on a chip (NoC) may limit the performance and/or power consumption against high-performance SoCs.

Third, on-chip multiprocessors (CMPs) in the near future will have on-chip caches and off-chip memory. Then a CMP design may be restricted by the number of available I/O pins. Although the number of pins per chip will increase, the number of transistors per chip will increase at a much higher rate [3]. So with the limited number of pins, we have to maintain a high traffic rate required between the on-chip caches and probably off-chip memory.

Static clusters fixed in hardware are effective to reduce the traffic to memory [12,13], and hence, to alleviated the pin-neck problem. However, the traffic concentrates on the per-cluster units such as the global interconnect interface and the cache coherence directory when the request rate is high in each cluster, leading to a long delay for the requests crossing the chip boundary.

K. Li et al. (Eds.): NPC 2007, LNCS 4672, pp. 404–414, 2007.

Good news with VLSI technologies is that a small node degree (i.e., the number of links incident to a node) is not necessarily a measure of good networks since almost all links of a network can be accommodated in the chip. Notice that a larger node degree generally leads to a smaller diameter but longer wires.

This paper proposes a *semi-completely-connected bus (SKB)* to cope with the long-wire and pin-neck problems mentioned above. We approach to the first problem by a small diameter, while to the second one by one set of dynamic clusters produced for *each target*, such as a memory block, of requests.

The small diameter with a large node degree may aggravate the long-wire problem, though it reduces the number of hops required in communications. So we assume a technology for very fast global wires, such as the transmission lines with a delay determined by an LC time constant [4], instead of an RC constant. Alternatively, we can assume a much larger number of metal planes than the current (about 10) to allow very fat and very sparse wires.

We derive a *semi-complete (SK)* graph by a simple partitioning of nodes. The SKB has the topology of a modified SK graph so that the links incident to a node are replaced by a single bus owned by the node. The diameter of the SKB is equal to 1 bus step. The layout pattern for the buses is very regular, but the maximum bus length of the 2^n-node SKB is rather long, i.e., $\mathbf{O}(\sqrt{2^n})$.

In the rest of paper, Section 2 introduces the simple partitioning and derives SK graph and its recursive version. Section 3 presents the structure, layout, routing, and clustering of the SKB. Section 4 shows the evaluation results. Section 5 describes related work, and Section 6 concludes the paper.

2 Semi-Complete Graphs

This section introduces the simple partitioning and defines the SK graph and its recursive version.

Let $\langle s_{p_1}, \ell_{k_1} \rangle$ denote an n-bit node address, where s_{p_1} and ℓ_{k_1} are the values of the upper p_1-bit and the lower k_1-bit ($p_1+k_1 = n$) parts of the address. Moreover, let $P_{\langle s,\ell \rangle}$ be the partition represented by leader (i.e., the representative node) $\langle s, \ell \rangle$. Then the simple partitioning is defined as follows:

Definition 1. *The suit $S_{s_{p_1}}$ consists of 2^{k_1} leaders $\langle s_{p_1}, *_{k_1} \rangle$, and the partition $P_{\langle s_{p_1}, \ell_{k_1} \rangle}$ has 2^{p_1} members $\langle *_{p_1}, \ell_{k_1} \rangle$, where $*_b$ stands for b-bit don't care.*

With Definition 1, we can partition the n-bit address space in two ways: First we can produce 2^{p_1} suits $\mathcal{S} = \{S_{s_{p_1}} \mid 0 \leq s_{p_1} \leq 2^{p_1} - 1\}$ each of 2^{k_1} leaders. Second, with each suit $S_{s_{p_1}}$, we can obtain a set of 2^{k_1} partitions $\mathcal{P}_{s_{p_1}} = \{P_{\langle s_{p_1}, \ell_{k_1} \rangle} \mid 0 \leq \ell_{k_1} \leq 2^{k_1} - 1\}$ each of 2^{p_1} nodes and represented by leader $P_{\langle s_{p_1}, \ell_{k_1} \rangle}$, leading to a total of 2^{p_1} sets $\mathcal{P} = \{\mathcal{P}_{s_{p_1}} \mid 0 \leq s_{p_1} \leq 2^{p_1} - 1\}$.

This partitioning has the following interesting property: Members $\langle *_{p_1}, \ell_{k_1} \rangle$ of partition $P_{\langle s_{p_1}, \ell_{k_1} \rangle}$ are distributed to the all suits, and a member $\langle s'_{p_1}, \ell_{k_1} \rangle \in P_{\langle s_{p_1}, \ell_{k_1} \rangle}$ ($s'_{p_1} \neq s_{p_1}$) is the leader of partition $P_{\langle s'_{p_1}, \ell_{k_1} \rangle}$ when partitioned with suit $\langle s'_{p_1}, *_{k_1} \rangle$. Note that partition $P_{\langle s'_{p_1}, \ell_{k_1} \rangle}$ has also the members $\langle *_{p_1}, \ell_{k_1} \rangle$. We

graphically represent this property by a *partition graph* $PG_n(p_1, k_1)$ that has a link between each member and the leader in every partition.

Property 1. The $\mathrm{PG}_n(p_1, k_1)$ graph has 2^{k_1} complete graphs, K_{p_1}'s, each consisting of the 2^{p_1} nodes $\langle *_{p_1}, \ell_{k_1} \rangle$.

Example 1. The $\mathrm{PG}_3(2, 1)$ graph ($p_1 = 2$ and $k_1 = 1$) is shown in Fig. 1(a), where the node address $\langle s_{p_1}, \ell_{k_1} \rangle$ is denoted by $s_{p_1}\ell_{k_1}$. In this case, we can produce 4 ($= 2^{p_1}$) suits, S_0, S_1, S_2, and S_3, each of 2 ($= 2^{k_1}$) leaders. With any of the suits, we can obtain 2 ($= 2^{k_1}$) partitions each of 4 ($= 2^{p_1}$) members. For instance, with suit S_1, we can obtain the two partitions $P_{\langle 1,0 \rangle} = \{\langle 1, 0\rangle, \langle 0, 0\rangle, \langle 2, 0\rangle, \langle 3, 0\rangle\}$ and $P_{\langle 1,1 \rangle} = \{\langle 1, 1\rangle, \langle 0, 1\rangle, \langle 2, 1\rangle, \langle 3, 1\rangle\}$ with the leaders $\langle 1, 0\rangle$ and $\langle 1, 1\rangle$ in the suit. The members of each partition are scattered into the all suits and organizes a K_2. So the $\mathrm{PG}_3(2, 1)$ has 2 K_2 graphs each of 4 nodes (Property 1).

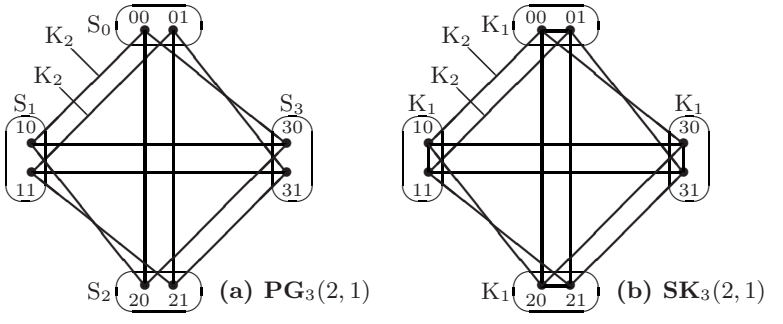


Fig. 1. (a) The $\mathrm{PG}_3(2, 1)$ graph for the partitioning when $p_1 = 2$ and $k_1 = 1$, and (b) the $\mathrm{SK}_3(2, 1)$ produced from the PG

We derive a *semi-complete (SK)* graph from a PG graph as described below. Then the obtained SK graph has the property presented next.

Definition 2. *An $SK_n(p_1, k_1)$ graph is obtained from the $PG_n(p_1, k_1)$ graph if we configure the 2^{k_1} leaders in each suit into the K_{k_1} graph, where $n = p_1 + k_1$.*

Property 2. The $\mathrm{SK}_n(p_1, k_1)$ has 2^{p_1} K_{k_1} graphs (Definition 2), that are connected to each other by 2^{k_1} K_{p_1} graphs (Property 1).

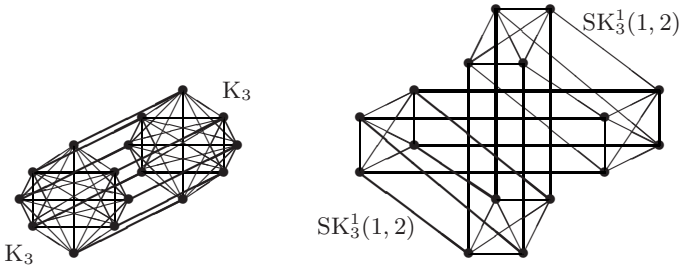
Example 2. The $\mathrm{SK}_3(2, 1)$ shown in Fig. 1(b) is obtained from the $\mathrm{PG}_3(2, 1)$ (shown in Fig. 1(a)) by connecting the nodes in every suit with each other to produce, in this case, a K_1. So the $\mathrm{SK}_3(2, 1)$ has 4 K_1 (i.e., 2^{p_1} K_{k_1}) graphs connected with each other by 2 K_2 (i.e., 2^{k_1} K_{p_1}) graphs.

When n is large, the K_{k_1} and K_{p_1} may be too large to achieve short wires in organizing an SKB form the SK. To cope with this problem, we exploit a recursive SK graph. Let $\mathrm{SK}^0_n(p_0, k_0) = \mathrm{K}_n$ ($p_0 = 0$, $k_0 = n$) and $p_i + k_i = n_i = k_{i-1}$

$(1 \leq i \leq r)$, where $\sum_{i=1}^{r} p_i + k_r = n$. Then we produce an r-level recursive SK graph, $\mathrm{SK}_n^r(p_1, \ldots, p_r, k_r)$, from the K_n by the recursive partitioning of each k_{i-1}-bit space to obtain an $\mathrm{SK}_{n_i}^1(p_i, k_i)$ (i.e., $\mathrm{SK}_{n_i}(p_i, k_i)$) as follows:

Definition 3. *An $SK_n^r(p_1, \ldots, p_r, k_r)$ is produced if we transform each $K_{k_{i-1}}$ in the $SK_n^{i-1}(p_1, \ldots, p_{i-1}, k_{i-1})$ to an $SK_{n_i}^1(p_i, k_i)$ to obtain the $SK_n^i(p_1, \ldots, p_i, k_i)$, for i from 1 to r.*

Example 3. We transform the initial K_4 into $\mathrm{SK}_n^1(p_1, k_1) = \mathrm{SK}_4^1(1, 3)$ (see Fig. 2(a)), and next obtain $\mathrm{SK}_n^2(p_1, p_2, k_1) = \mathrm{SK}_4^2(1, 1, 2)$ (see Fig. 2(b)) by converting each $\mathrm{K}_{k_1} = \mathrm{K}_3$ in $\mathrm{SK}_4^1(1, 3)$ to $\mathrm{SK}_{n_2}^1(p_2, k_2) = \mathrm{SK}_3^1(1, 2)$ $(p_2 + k_2 = n_2 = k_1)$.



(a) $\mathbf{SK}_4^1(1, 3)$ **(b) $\mathbf{SK}_4^2(1, 1, 2)$**

Fig. 2. (a) The $\mathrm{SK}_4^1(1, 3)$ produced from K_4 and (b) the $\mathrm{SK}_4^2(1, 1, 2)$ obtained from $\mathrm{SK}_4^1(1, 3)$

The number of links and the diameter of SK_n^r are equal to $2^{n-1} \sum_{i=1}^{r+1}(2^{p_i} - 1)$ (where $p_{r+1} = k_r$) and $r + 1$, respectively. Since the number of links in the K_n graph is $\mathbf{O}(2^{2n-1})$, SK_n^r reduces the number by the factor of $\mathbf{O}(2^{p_i}/2^n)$.

3 The Semi-Completely-Connected Bus: SKB

This section describes the structure, layout, routing, and dynamic clustering of the SKB. The effect of the clustering as applied to a memory hierarchy is also analyzed. For space, the description of recursive SKBs is omitted.

3.1 Structure of the SKB

The SKB_n is organized from the SK_n and is laid out as follows. The number of buses in SKB_n equals one per node and the diameter equals 1 (bus step).

Definition 4. *The $SKB_n(p_1, k_1)$ is obtained if we replace all links incident to a node of the $SK_n(p_1, k_1)$ by a single bus for the node, for all nodes in the SK_n.*

Definition 5. *We arrange the nodes of the $SKB_n(p_1, k_1)$ into a $2^{p_1} \times 2^{k_1}$ array and put node $\langle s, \ell \rangle$ at the position with array index (s, ℓ).*

Example 4. The layout of $\mathrm{SKB}_6(3,3)$ is shown in Fig. 3. The BSK has 64 buses. The two-digit integer $s\ell$ in the node box stands for the node address $\langle s, \ell \rangle$. Each node has two sets of ports; one set for vertical lines, and the other for horizontal lines. For space, only even-numbered ports are shown. The bus for node $\langle s, \ell \rangle$ consists of a horizontal line along row s and a vertical line along column ℓ; the diamond ($\diamond$) at the cross point of the horizontal line and the stub to node $\langle s, \ell \rangle$ shows that the node is the owner of the bus. The bus of node, for instance, $\langle 2, 3 \rangle$ runs along row 2 and column 3 as shown by the bold line.

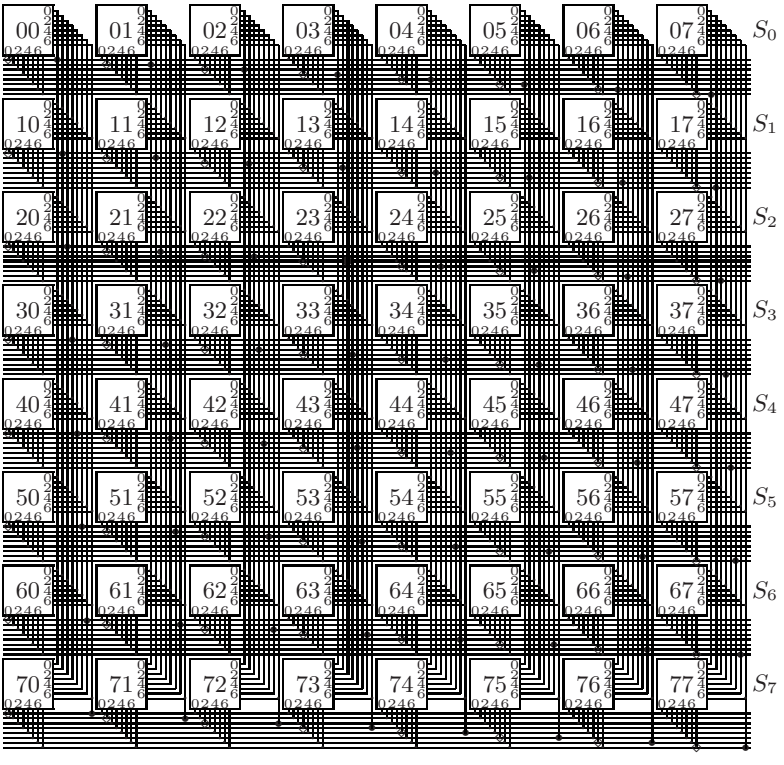


Fig. 3. The layout of $\mathrm{SKB}_6(3,3)$

Let the unit length of the bus be equal to the distance between the adjacent nodes on the SKB_n layout, and the *bus length* be defined as the maximum of the distances for a signal to traverse for all buses. Then the routing distance is largest when the source and target are located at, for instance, the top-left and bottom-right corners of the array, respectively. Then the bus length of the $\mathrm{SKB}_n(p_1, k_1)$ equals $2^{p_1} + 2^{k_1}$, that reduces to $2\sqrt{2^n}$ when $p_1 = k_1$.

3.2 Routing on the SKB

Let S-ports and L-ports of a node be the ports for choosing one of the suits and leaders, respectively, so are connected to the vertical and horizontal lines of the buses. Then S-port i of node $\langle s, \ell \rangle$ is connected to the bus of node $\langle i, \ell \rangle$, and L-port j to the bus of node $\langle s, j \rangle$ (see Fig. 3). Moreover, the bus of node $\langle s, \ell \rangle$ is connected to nodes $\langle s, *_{k_1} \rangle$ via the horizontal line. So the routing on the SKB is performed in the following way, and its diameter is given by the next theorem.

Definition 6. *To send a packet to the target node $\langle s_t, \ell_t \rangle$, the source node $\langle s, \ell \rangle$ puts the packet on its S-port s_t if $s \neq s_t$, or on its L-port ℓ_t otherwise.*

Theorem 1. *The diameter of $SKB_n(p_1, k_1)$ is equal to one bus step.*

Proof. When $s \neq s_t$, the S-port s_t of source $\langle s, \ell \rangle$ is connected to the bus of node $\langle s_t, \ell \rangle$ and that bus is connected with the target $\langle s_t, \ell_t \rangle$. Otherwise, the L-port ℓ_t of source $\langle s, \ell \rangle$ is connected to the bus of the target $\langle s, \ell_t \rangle = \langle s_t, \ell_t \rangle$. In both cases, the packet is sent from the source to the target in 1 bus-step.

Example 5. Assume the source node $\langle 2, 3 \rangle$ and the target node $\langle 7, 6 \rangle$ in Fig. 3. Then the source $\langle 2, 3 \rangle$ puts the packet on S-port 7. The port is connected to the bus of node $\langle 7, 3 \rangle$, and the bus is connected with the target $\langle 7, 6 \rangle$. When the target is $\langle 2, 7 \rangle$ for the same source, it sends the packet via L-port 7 connected to the target's bus. In both cases, the packet reaches the target in 1 bus-step.

3.3 Dynamic Clustering on the SKB

Suppose that a CMP consists of on-chip processing nodes and off-chip memory, and that the node has a processor, a level-1 (L1) cache, and a level-2 (L2) cache. *Dynamic clustering* of memory requests refers to the dynamic partitioning *only* of the nodes requesting for a specific memory block. So the size of a cluster is no more than the partition size 2^{p_1}.

Let M_s denote the off-chip shared memory unit consisting of memory blocks B_{s*} whose addresses $s*$ equal s in the upper p_1 bits. We associate unit M_s with suit S_s. When no copy of a memory block $B_{t\alpha} \in M_t$ is in the L2 cache of a node $\langle s, \ell \rangle$, we produce a set of dynamic clusters for block $B_{t\alpha}$, partitioning with suit S_t in the following way.

Definition 7. *In the dynamic clustering of the requests for a memory block $B_{t\alpha}$, a node $\langle s, \ell \rangle$ sends the request to the L2 cache of leader $\langle t, \ell \rangle$ in suit S_t.*

The L2 cache of leader $\langle t, \ell \rangle$ returns the copy of block $B_{t\alpha}$ if it has the copy. Otherwise, it produces a single request for the sake of the requests for block $B_{t\alpha}$ received in the cluster and sends it to unit M_t, crossing the chip boundary. The next theorem shows the effect of dynamic clustering.

Theorem 2. *The traffic to leaders for clustering memory requests reduces to at most $1/2^{p_1}$ of the traffic to the per-cluster units in the static clusters.*

Proof. Assume that the memory requests are uniformly distributed to all memory units. Then in the static clusters, the requests not satisfied in the L2 cache concentrate on a single per-cluster unit. On the other hand, with the dynamic clusters, the request is sent to one of the 2^{p_1} leaders depending on the address of requested memory block, so that the traffic to one leader reduces to one 2^{p_1}th of the traffic to the per-cluster unit.

When a miss occurs on block $B_{t\alpha}$ in the L2 cache of node $\langle s, \ell \rangle$, the copy of $B_{t\alpha}$ may be in some L2 caches in partition $P_\ell = \langle *, \ell \rangle$ and suit $S_t = \langle t, * \rangle$. To reuse the copy by an L2-L2 direct transfer, all L2 caches in P_ℓ snoop all memory requests issued in P_ℓ, while the L2 caches in S_t snoop the requests for the blocks in M_t. The clustering is performed only if P_ℓ and S_t have no copy of block $B_{t\alpha}$.

Example 6. Suppose in Fig. 3 that a miss on a memory block in unit M_0 (associated with S_0) occurs in the L2 cache of node $\langle 2, 3 \rangle$. Then the node puts the memory request on the bus of node $\langle 0, 3 \rangle$, that is snooped by the all L2 caches in partition P_3 and suit S_0. When no copy is found in those L2 caches, the clustering is performed with the L2 cache of leader $\langle 0, 3 \rangle$.

4 Evaluation

This section shows the results of evaluation on the 64-node SKB, comparing with the representative network, the 64-node hypercube.

4.1 Environments

We evaluate the SKB (denoted by `SKB`) and hypercube (`Hyp`) by a cycle-accurate simulator. They have the link- or bus-width of 1 byte, and perform wormhole routing. `Hyp` has simultaneously bidirectional links of which delay is fixed equal to one clock to use `Hyp` as the reference network. Let `SKB`d denote the `SKB` whose bus delay equals d clocks. We denote the SKB with dynamic and static clustering by appending `D` and `S` after the network notation, such as `SKB4D` and `SKB2S`.

Processing nodes simultaneously send 13-byte packets (header: 9 bytes, data: 4 bytes) to the individual target nodes chosen randomly in each node. The packet issue-rate in each node is varied from 1 packet per 1k clocks up to the rate at which the network's bandwidth saturates; the rate is fixed in one simulation run.

We simulate the 64-node networks; for `SKB`, we use $\mathrm{SKB}_6(3, 3)$, so the cluster size equals 8. In the dynamic clustering, the leader produces a packet every time it receives the same number of packets as the cluster size and sends the produced packet to the target. We implement static clustering by fixing the leader to a specific node in each cluster independent of the target nodes.

We measure the total number of clocks required for sending 2048 packets per node to the targets and calculate the network's bandwidth relative to the `Hyp`'s. We also calculate the average network delay from the source to the target.

4.2 Results

The bandwidth and network delay of SKBd (with no clustering) are shown in Fig. 4, where the bus cycle time d is varied from 1 to 4, and 8. The bandwidth of SKBd equals about 0.97, 0.53, 0.36, 0.28, and 0.14 respectively when d is equal to 1, 2, 3, 4, and 8, relative to the hypercube's. The network delay is then equal to about 18, 32, 45, 62, and 140 clocks respectively when the network are not saturated; the delay in Hyp equals about 24 clocks. Thus SKB1 and Hyp have almost the same performance. The performance of SKBd decreases almost proportional to d.

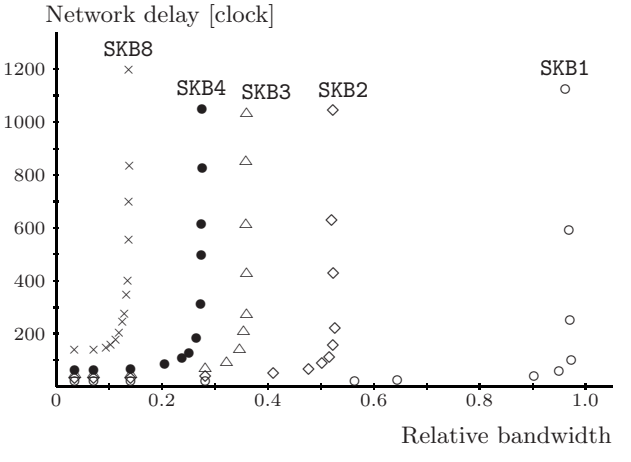


Fig. 4. Performance of SKBd relative to Hyp

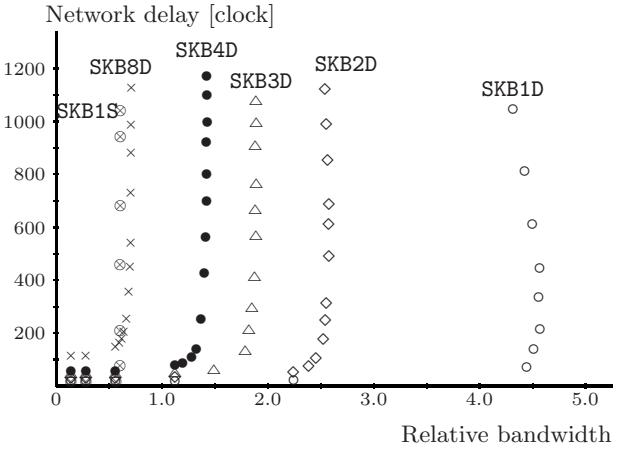


Fig. 5. Performance of SKBdD and SKB1S relative to Hyp

The dynamic clustering is effective to improve the performance of SKB as shown in Fig. 5. The relative bandwidth increases to about 4.57, 2.58, 1.89, 1.43, and 0.71 for SKB1 to SKB4, and SKB8, respectively; then the network delay decreases to 17, 30, 41, 55, and 117 clocks. This is because the packets issued in each cluster are sent to one of 8 leaders depending on the target address so that no traffic congestion on a specific leader occurs, though one extra-packet per 8 received packets is produced in each leader and is sent to the target.

On the other hand, static clustering SKB1S is not effective when the request rate is high: SKB1S decreases the SKB1's bandwidth of 0.97 to about 0.61 due to the traffic contention on the fixed leaders.

5 Related Work

One approach to NoCs is to achieve a topology with an easy layout and short wires, such as the tree and mesh. SPIN [7] adopts a fat-tree [8], where each node has four children and four parents to increase the bandwidth. A 2-dimensional torus is used to reduce the network delay of the mesh [9]. Octagon [10] exploits a recursive ring, where one recursive level is an extended ring of 8 nodes with extra links between the nodes located at the opposite locations on the ring. The hierarchical bus network [11] is also attractive since it reduces local communication overhead but also allows us to use snooping cache protocols.

An alternative is the interconnect-centric approach [5,6]. In [6], interconnects consist of wires with varying latency, bandwidth, and energy characteristics, and are mapped on appropriate interconnects for cache coherence operations. In [5], transmission line technology [4] is exploited to access level-2 on-chip caches. Our SKB study was partially stimulated by the results in [4,5].

To reduce memory traffic, the commercial multiprocessors (MPs), STiNG [12] and SGI Origin [13], adopt clusters connected to each other by the ring and the fat bristled hypercube, respectively. A research CMP, Hydra [14], has 4 processors and exploits two buses to interconnect their L1 and L2 caches, while another CMP, Piranha [15], uses a crossbar between the L1 and L2 caches for 8 processors. But no concept of dynamic clustering is found in those MPs.

The partitioning based on an extended Hamming code [16] is used to organize the networks [17,18]. However, no SK graph is derived from this partitioning.

6 Conclusions

We presented the topology, structure, routing, and dynamic clustering of the SKB to alleviate the long-wire and pin-neck problems against high-performance NoCs. The SKB is a bus-based semi-complete (SK) network, and the SK graph is derived from the relationship between the suits of leaders and the partitions produced with the all suits in a simple partitioning.

The 2^n-node SK graph, $\mathrm{SK}_n(p,k)$ $(p{+}k=n)$, consists of 2^p number of 2^k-node complete graphs, K_k's. Each node in a K_k is connected to the nodes scattered in the other K_k graphs in such a way that those nodes together organize a complete

graph K_p. The 2^n-node SKB, SKB_n, is obtained from the SK_n by replacing the links incident to a node by a single bus for the node.

We laid out the nodes of the SKB_n on the $2^p \times 2^k$ array. Then the buses have very regular wiring patterns and the maximum length of $\mathbf{O}(\sqrt{2^n})$; it is expected that the delay of this rather long wire is alleviated by the transmission line technology or fat and sparse wires.

We evaluated the SKB_6 by a cycle-accurate simulator. The results show that relative to the bandwidth of the hypercube with the link delay of 1 clock, the SKB's bandwidth is about 0.97, 0.53, 0.36, 0.28, and 0.14 when the bus delay equals 1, 2, 3, 4, and 8 clocks, respectively. The bandwidths increase to about 4.57, 2.58, 1.89, 1.43, and 0.71 when the dynamic clustering is exploited.

We are designing an on-chip cache hierarchy and its coherence protocol so that we can evaluate the effects of dynamic clustering of memory requests.

References

1. Matzke, D.: Will Physical Scalability Sabotage Performance Gains. IEEE Computer 30, 37–39 (1997)
2. Magen, N., Kolodny, A., Weiser, U., Shamir, N.: Interconnect-Power Dissipation in a Microprocessor. In: Proc. of System Level Interconnect Prediction, pp. 7–13 (2004)
3. Huh, J., Burger, D., Keckler, S.W.: Exploring the Design Space of Future CMPs. In: Proc. Int. Conf. on Parallel Architectures and Compilation Techniques, pp. 199–210 (2001)
4. Chang, R.T., Talwalkar, N., Patrick, C.P., Wong, S.S.: Near Speed-of-Light Signaling Over On-Chip Electrical Interconnects. IEEE Jour. on Solid-State Circuits 38, 834–838 (2003)
5. Beckmann, B.M., Wood, D.A.: TLC: Transmission Line Caches. In: Proc. Int. Symp. on Microarchitecture, pp. 43–54 (2003)
6. Cheng, L., et al.: Interconnect-Aware Coherence Protocol for Chip Multiprocessors. In: Proc. 33rd Int. Symp. on Computer Architecture, pp. 339–350 (2006)
7. Guerrier, P., Greiner, A.: A Generic Architecture for On-Chip Packet-Switched Interconnections. In: Proc. Design and Test in Europe (DATE), pp. 250–256 (2000)
8. Leiserson, C.E.: Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing. IEEE Trans. on Computer C-34, 892–901 (1985)
9. Dally, W.J., Towles, B.: Route Packets, Not Wires: On-Chip Interconnection Networks. In: Proc. Design Automation Conf. (DAC), pp. 683–689 (2001)
10. Karim, F., et al.: An Interconnect Architecture for Networking Systems on Chips. IEEE Micro 22, 36–45 (2002)
11. Kumar, R., Zyuban, V., Tullsen, D.M.: Interconnection in Multi-Core Architectures: Understanding Mechanisms, Overhead and Scaling. In: Proc. 32nd Int. Symp. on Computer Architectures, pp. 408–419 (2005)
12. Lovett, T., Clapp, R.: STiNG: A ccNUMA Computer System for the Commercial Marketplace. In: Proc. 23th Int. Symp. on Computer Architectures, pp. 308–317 (1996)
13. Laudon, J., Lenoski, D.: The SGI Origin: A ccNUMA Highly Scalable Server. In: Proc. 24th Int. Symp. on Computer Architectures, pp. 241–251 (1997)

14. Olukotun, K., et al.: The Case for a Single Chip Multiprocessor. In: Proc. 7th Int. Conf. on Architectural Support for Programming Languages and Operating Systems, pp. 2–11 (1996)
15. Barroso, L.A., et al.: Piranha: A Scalable Architecture Based on Single-Chip Multiprocessors. In: Proc. 27th Int. Symp. on Computer Architectures, pp. 282–293 (2000)
16. Takesue, M.: Ψ-Cubes: Recursive Bused Fat-Hypercubes for Multilevel Snoopy Caches. In: Proc. Int. Symp. on Parallel Architectures, Algorithms, and Networks (I-SPAN), pp. 62–67 (1999)
17. Takesue, M.: DC-Mesh: A Contracted High-Dimensional Mesh for Dynamic Clustering. In: Jin, H., Gao, G.R., Xu, Z., Chen, H. (eds.) NPC 2004. LNCS, vol. 3222, pp. 382–389. Springer, Heidelberg (2004)
18. Takesue, M.: The Psi-Cube: A Bus-Based Cube-Type Network for High-Performance On-Chip Networks. In: Proc 2005 Int. Conf. on Parallel Processing (ICPP) Workshops, pp. 539–546 (2005) (For the full version, see Parallel Computing, vol. 32, pp. 852–869, Elsevier B. V (2006))