Hardware-Assisted Design for Fast Packet Forwarding in Parallel Routers

Nian-Feng Tzeng
Center for Advanced Computer Studies
University of Louisiana at Lafayette
Lafayette, Louisiana 70504, U.S.A.
tzeng@cacs.louisiana.edu

Abstract

A hardware-assisted design, dubbed cache-oriented multistage structure (COMS), is proposed for fast packet forwarding. COMS incorporates small on-chip cache memory in its constituent switching elements (SE's) for a parallel router to interconnect its line cards (LC's) and forwarding engines (FE's, where table lookups are performed). Each lookup result in COMS is cached in a series of SE's between the FE (which performs the lookup) and the LC (where the lookup request originates). The cached lookup results fulfill subsequent lookup requests for identical addresses immediately without resorting to FE's for (time-consuming) lookups, thus reducing the mean lookup time tremendously. COMS calls for partitioning the set of prefixes in a routing table into subsets (of roughly equal sizes) so that each subset involves only a small fraction of the table for one FE. This leads to a substantial savings of SRAM required in each FE to hold its forwarding table, and the total savings of SRAM in a parallel router far exceeds the amount of SRAM employed in all SE's of COMS combined. A COMS-based router of size 16 exhibits over 10 times faster mean packet forwarding than its compatible router without caching nor table partitioning. The worst case lookup time in COMS depends on the matching algorithm employed in FE's and can often be shorter than that in a compatible router. With its ability to forward packets swiftly, COMS is ideally suitable for the new generation of parallel routers.

1. Introduction

Rapid growth in the Internet prompts continuing expansion of the routing tables, in particular, those tables in the core routers. Meanwhile, explosively increasing traffic over the Internet demands the routers to handle faster links, operating up to OC-192 or even OC-768 (40 Gbps). This calls for high-performance routers able to forward hundreds of millions of packets per second. Such high forwarding rates undoubtedly pose a router design challenge, which has been addressed by different approaches, including enhanced routing/forwarding table lookup algorithms [1-4], hardware-based lookup designs, and hardware-assisted lookups [5, 14]. An extensive survey of IP address lookup approaches is given in [6].

Search in the routing/forwarding tables is complex since table entries follow classless inter-domain routing (CIDR) [7] intended to yield more efficient use of the IP address space and to moderate routing/forwarding table growth. Typically, *longest prefix matching search* is adopted because its search result is most *specific* to a given IP address under search. Such a search can be carried out effectively if prefixes are organized as a tree-like structure called a *trie*, whose nodes either correspond to prefixes or form paths to prefixes [8].

A software lookup algorithm is enhanced either by lowering its memory requirement for the routing table in order to be fit in static RAM (SRAM) or by devising more effective longest prefix matching aimed to reduce the number of memory accesses per lookup in the routing tables [2-4]. For a given lookup algorithm, the memory requirement tends to increase as the number of prefixes grows. It is prohibitively expensive, if not impossible, to hold all prefixes in on-chip SRAM, leaving relatively slower off-chip SRAM (with access time mostly within 8 – 15 *ns*) as a feasible option. A software lookup search includes multiple memory accesses, and its lookup time contains two components: one due to multiple memory accesses and the other due to program execution (involving some one hundred instructions or so). The former time component ranges from tens to hundreds of *ns* when prefixes are held in off-chip SRAM, while the latter one is machine-dependent but typically is no less than 100 *ns*. If the mean lookup time equals, say, 200 *ns* when forwarding IP packets of 256 (or 512) bytes in length on an average, there will be 5 millions lookups carried out per second, amounting to the line speed of 10 (or 20) Gbps.

Table lookups in a high-performance router are done by multiple forwarding engines (FE's) independently and concurrently. Such a router accommodates many line cards (LC's) for external links to terminate. In a parallel router, LC's share an array of FE's, which are separated from the LC's [12], as illustrated in Fig. 1. The number of FE's is determined according to the aggregate capacity of all links connected to the router, and it could be different from the number of LC's, allowing higher port density in LC's and more flexibility in router configuration. For example, the Juniper M160 backbone router belongs to parallel router architecture and it contains 4 FE's, which

This work was support in part by NSF under Grants EIA-9871315 and CCR-0105529 and by the Board of Regents of the State of Louisiana under Contract No. LEQSF(2000-01)-ENH-TR-90.

are shared by up to 8 LC's. This work focuses on the **parallel-router architecture**, where each packet arriving at an LC sends its header to one FE for the table lookup and the lookup result is then sent back to the arrival LC over the switching fabric(s). Multistage interconnects are considered to be the switching fabrics here, and such an interconnect provides connections dynamically between LC's and FE's in the router.

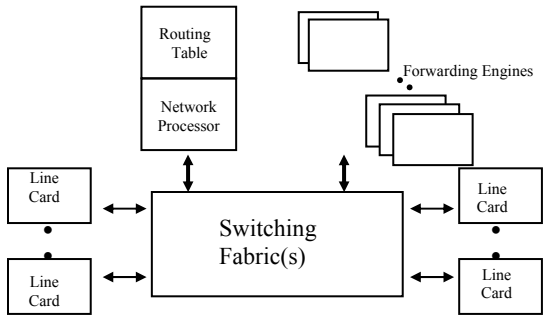

Fig. 1. Parallel router architecture.

This work proposes to equip the multistage interconnect with caches (made from on-chip SRAM) and to partition the routing table into small forwarding tables held in the FE's. Caches in the multistage interconnect serve to capture IP lookup results obtained from FE's along the paths between lookup request originators (i.e., LC's) and FE's (to perform the requested lookups) so that subsequent lookup requests, when hit in the interconnect, can be satisfied immediately without proceeding to the FE's for (time-consuming) table lookups. Our proposed multistage interconnect is referred to as the ***cache-oriented multistage structure*** (COMS), which comprises multiple stages of 2×2 switching elements (SE's) equipped with caches. Given 6 bytes in a block for IPv4 addressing, the amount of cache in each SE typically equals 24 Kbytes (= 6 bytes/block × 4K blocks), while the savings of SRAM resulting from a smaller trie in each FE after routing table partitioning usually amounts to hundreds of Kbytes. For example, under the Lulea trie [2] (whose storage requirement is often the lowest) with a router of size N = 4 (or 16), the partitioned table in any FE requires no more than 92 (or 39) Kbytes, as opposed to some 260 Kbytes in an FE of a conventional router without partitioning, when the FUNET routing table in [4] is considered, according to our actual implementations. For the LC trie [4] with fill factor 0.25 under N = 4 (or 16), the saving amount of SRAM in each FE due to partitioning is over 815K (or 1008) Kbytes, far bigger than the size of SRAM used for each SE cache (i.e., 24 Kbytes). Likewise, the DP trie [1] sees SRAM reduction in each FE by similar amounts. COMS is an effective hardware-assisted design for fast packet forwarding, usually reducing the total amount of SRAM (in all FE's and SE's combined) tremendously.

A parallel router with COMS can yield significantly better forwarding performance, on an average, than an existing router without caching nor table partitioning. Our extensive simulation results indicate that a COMS-based parallel router with 16 LC's and FE's can forward more than 1120 millions packets per second, if the COMS cache involves 4K blocks, when the Lulea trie [2] is adopted for longest prefix matching. This average forwarding ability is 10 times faster than that of an existing router. In addition, a COMS-based router may exhibit a shorter lookup time in the worst case when compared with a conventional router under the same longest prefix matching algorithm. Given COMS with N = 4 (or 16) composed of 2×2 SE's, for example, a packet takes 2×δ (or 4×δ) longer to travel over COMS than over a regular BMI (without caches, see Fig. 2), where δ is the cycle time difference of an SE with and without cache. If δ is assumed to be 3 *ns* (a reasonable value, as the SE cache is of very small on-chip SRAM), a round-trip delay over COMS takes 12 *ns* (or 24 *ns*) longer than over its BMI counterpart, for N = 4 (or 16). However, the matching algorithm executed in an FE under COMS can be shorter, as a result of partitioning the routing table. If the Lulea trie is adopted, for example, a far smaller forwarding table in each FE after partitioning may avoid the dense chunks and the very dense chunks in the 3rd level or even avoid all the 3rd level chunks of the trie [2], reducing 2 or even 4 memory accesses (from 4+4+4 down to 4+4+2 or even to 4+4) for the worst case; this memory access reduction translates to a savings of 20 *ns* or even 40 *ns* (assuming the access time of off-chip SRAM is 10 *ns*). Similarly, if the LC trie is employed, a smaller forwarding table under our router design usually reduces the maximum path length in the trie constructed. Considering the FUNET routing table under a fill factor of 0.25, for example, the search path depth in the LC trie after partitioning for N = 4 (or 16) is bounded by 6 (or 6), whereas the maximum trie depth without partitioning equals 5+3 = 8 (based on the implementation provided in [4]), giving rise to a savings of at least 2 memory accesses in the worst case. As a result, the worst case lookup time under COMS may be shortened.

While IP lookup traffic streams exhibit different characteristics than the data streams of typical computing applications, our extensive simulation studies using various recent traces collected over high-speed ports (including OC48c of Cisco GSR 12015 backbone routers during August 14–27, 2002) available at the NLANR's PMA trace archive [11] revealed the effectiveness of caches with respect to table lookups of IP traffic, requiring 2048 cache blocks in each SE to attain hit rates exceeding 0.97 for COMS with size 16 under all traces examined. For a given cache size, the larger the COMS is, the higher hit rate it attains; this results from (1) fragmenting the IP addresses (and also the set of prefixes) into more partitions through COMS, yielding better

address space coverage (thanks to fewer prefixes) by each SE connecting the FE's, and (2) a larger combined cache capacity. A cache-based solution stated here is deemed a viable approach to fast forwarding for the future Internet.

2. Pertinent Work

This section first reviews different tries (for longest prefix matching) and multistage interconnects briefly. The switch cache considered previously and cache memory for network processors are then highlighted.

2.1 Tries and Multistage Interconnects

A software-based packet forwarding approach often organizes all prefixes of a forwarding table as a particular *trie* to facilitate the longest prefix matching process. Initially, a variation of the (binary) trie obtained by compressing paths and with some modifications to support longest prefix matching, known as a *BSD trie* [8], was adopted in Berkeley Unix. Later, an enhanced trie implementation, called a *DP trie* (dynamic prefix trie), was considered to lower the number of memory accesses during search [1]. This DP trie yields a small code size and a low storage requirement, confining the effects of random insertion and deletion operations to be local for rapid updates. The Lulea algorithm constructs a 3-level compressed data structure, with the strides of 16, 8, and 8, respectively, for the first, the second and the third levels [2]. Another method replaces the largest full binary subtrie of a binary trie with a corresponding one-level multiple-bit subtrie recursively, starting with the root level, to produce an *LC trie* (level-compressed trie) [4]. Search in an LC trie requires an explicit comparison when arriving at a leaf to ensure a search match.

The multistage interconnect is particularly suitable for connecting LC's and FE's in a future router due to its good scalability. A bi-directional multistage interconnect (BMI) of size 8 is illustrated in Fig. 2, where the basic switching element (SE) is of size 2×2, and there are $\log_2(8)$ stages of SE's interconnected by bi-directional links. The routing decision in each SE is based on one bit of the routing tag. In the upward direction of the BMI depicted in Fig. 2 (i.e., from bottom to top), a routing tag is one top port number. For example, the routing tag from BP_2 to TP_6 is 110_2 (= 6), with the first bit for setting an SE in the bottom stage, such that a "0" (or "1") makes a connection to the left (or right) upstream port. Similarly, the second (or third) bit is to control an SE in the next (or top) stage. The path from BP_2 to TP_6 is marked in Fig. 2, so is the path from BP_4 to TP_3 (with its tag being 011_2 = 3). Routing in the downward direction takes the same path as in the opposite direction; it can be done easily by keeping arrival port information in each visited SE when the package was routed upward, as will be elaborated in Section 3.3. This way enables downward routing without using a tag and is applicable to any interconnect topology.

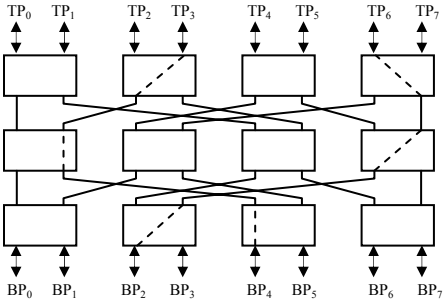

Fig. 2. Bi-directional multistage interconnect (BMI).

2.2 Network and Lookup Caches

Network Caches

Various caching strategies have been proposed for multiprocessor systems to alleviate the impacts of remote memory access latencies on system performance. In particular, a small fast SRAM cache was considered in each SE of an interconnect, called the *switch cache*, for capturing shared data as it flows through the interconnect, so as to serve future requests for the data quickly [13]. The switch cache operates as follows. After a memory read request is satisfied (by a memory module), the read result is sent back to the request originator (a processor). On the way back, the result is recorded on the switch cache of each visited SE. Along the direction to its memory module, a memory read request is compared against the cache contents at each visited SE to see if there is a *hit*. Any hit leads to a reply generated at the SE and sent along the reverse path back to the request originator, with the original memory request marked as *switch hit* and routed in the same way toward its destined memory module, where the corresponding cache block directory is updated accordingly [13]. A write request when passing through SE's, checks their caches and invalidates every hit cache block, if any, along the path.

Caching Lookup Results

Recently, a network processor equipped with hardware caches for capturing table lookup results has shown to improve overall packet forwarding performance significantly [5, 14]. The caching algorithm described in [14] mapped IP addresses carefully to virtual addresses so as to make use of CPU caches (both L1 and L2) for fast lookups, reaching more than 80 million lookups per second according to detailed simulation on a 500 MHz Alpha processor with 16 Kbytes of L1 cache and 1 Mbytes of L2 cache. Separately, a technique for improving the effective coverage of the IP address space has been considered by caching a *range* of contiguous IP addresses in each entry [5]. It yields better performance when the address range cached in an entry is larger, as the address space covered by a given cache structure is then bigger. To this end, address range merging is adopted to get a large range. Two steps of address range merging

were considered [5]. Simulation results have confirmed that address range merging after proper mapping may improve caching efficiency markedly.

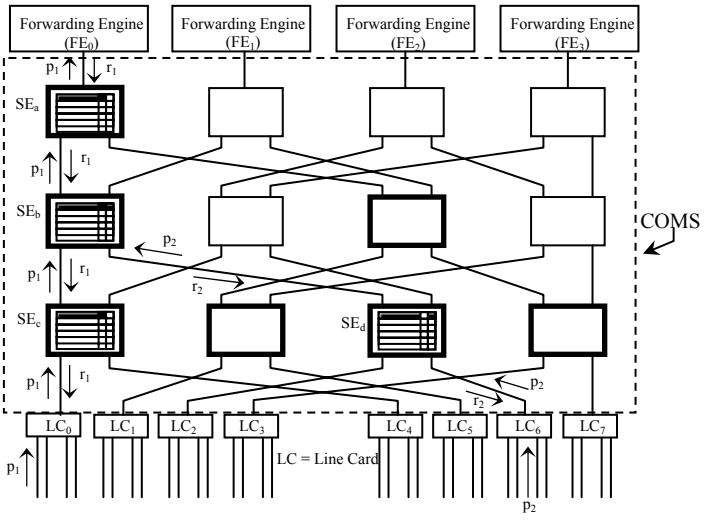

Fig. 3. Proposed cache-oriented multistage structure (COMS), derived from BMI shown in Fig. 2. (A tree of SE's rooted at FE_0 is highlighted.)

3. Cache-Oriented Multistage Structure

The cache-oriented multistage structure (COMS) is proposed as a scalable interconnect between LC's and FE's in a parallel router for high forwarding performance, providing connection paths for LC's to share FE's where packet lookups are performed. It comprises multiple stages of SE's, each of which is equipped with (on-chip) SRAM for caching lookup results returning from FE's. As shown in Fig. 3, COMS is a bi-directional multistage interconnect (BMI). Routing control in COMS is distributed, according to the routing tags stated earlier.

3.1 COMS Operations

Each FE in a router contains a forwarding table to enable lookups. The forwarding tables of all FE's reflect changes caused by updates to the core routing table. To ensure appropriate lookups, this work assumes that all cache contents in COMS are *flushed entirely after each table update*. In the next section, COMS will be demonstrated by simulation to arrive at very high forwarding performance, even under this simple cache invalidation (by flushing) method after each table update.

Consider a packet arrival immediately after a table update. The packet terminates at one LC (referred to as the *incoming LC*), where the packet header is extracted and delivered through the COMS to one FE for the longest-prefix matching lookup. Since the cache contents in COMS are all flushed at that time, the packet header cannot have a hit in any SE and will finally reach one FE (which is dictated by selected bits of the IP address), where the table lookup is conducted according to the longest-prefix matching algorithm implemented therein. After the lookup result is obtained, it is sent along the same path in the reverse direction back to the incoming LC. On its way back, the result is written to the cache of every visited SE. This cached result will satisfy later packet lookups for an identical destination address much faster. In general, a packet header delivered along COMS is checked against the cached entries in each visited SE; if there is a hit, a reply is produced by the SE and sent along the same path back to the incoming LC. Again, the reply is cached at each SE on its way back.

COMS is expected to yield high hit rates, as packets arriving at an LC closely together (in time) will have good chances to head for the same destination. Additionally, COMS can serve multiple packets (from different LC's) with the same destination concurrently at SE's in different stages of COMS, enjoying parallelism in packet lookups. In Fig. 3, for example, packet p_2 has a hit at SE_b after the lookup result of packet p_1 has flown back to its originating LC (i.e., LC_0). At the same time, another packet destined for the same address (not shown in the figure) from LC_1 can be served by SE_a. A lookup result obtained at an FE (say, FE_0 in Fig. 3) may be cached in COMS along a *tree* of SE's, rooted at FE_0 and spanned across all LC's. This permits to satisfy subsequent lookups of the same address for packets arriving at any LC's. Since every SE has a potential to serve one packet (with any destination address) in a cycle, the degree of parallelism offered by COMS could be massive, in particular for a large COMS. Our COMS keeps arrival port information of every packet which is recorded and waiting in the cache for completion, so as to simplify routing the lookup reply along the reverse direction. As a result, no routing tags are needed for delivering lookup replies. In addition, a simple but effective cache replacement mechanism is devised for COMS to enhance its hit rates, as elaborated next.

3.2 Cache Management

Cache management affects the hit rate of COMS, and thus the effective lookup times of packet addresses. To enhance the performance of COMS, appropriate status bits are required in each cache entry for implementing an efficient cache replacement mechanism. As a cache entry is at either an *invalid state* or a *shared state*, its status is denoted by one bit. In addition, two *duplication status bits*, called the *L-bit* and the *R-bit*, are introduced to each cache entry in an SE (say, SE_t) for signifying if the left-child and the right-child of SE_t contain a copy of the same entry in their respective caches. These two bits reflect the redundancy degree of a cache entry and are helpful to achieve higher hit rates under a given cache organization, resulting from better cache replacement.

In this work, the cache block size is chosen to hold only *one address lookup result*, because the devices with contiguous IP addresses usually have little direct

temporal correlation of network activities. The cache size and the degree of set associativity are left as design parameters for investigation. When a lookup result is sent back to the incoming LC, it is cached at each visited SE. If the SE has no free entry in the set of interest (decided by the destination IP address), one entry in the set has to be chosen for replacement, provided the degree of set associativity is larger than 1 (which is common). To minimize the adverse impact of replacement, it is obvious to choose an entry with *both L-bit and R-bit set*, if any, as the one to be evicted because replacing the entry will have little impact on future hits. These history status bits are examined first to decide which entry to be replaced. If multiple entries have their both duplication status bits being set, a conventional replacement strategy (such as LRU, FIFO, or random) is applied to break the tie. On the other hand, when no entry has its both bits set, a conventional replacement strategy is also followed to choose the entry for eviction. After a cache entry is created completely in an SE, its associated L-bit (or R-bit) is set if the return path is along the left (or right) downstream link of the SE.

Our 2×2 SE's are assumed to operate at 200 MHz. Each visit to an SE thus takes 5 *ns* (which is possible since on-chip SRAM is employed in SE's and the access time of such SRAM can be as low as 1 *ns*, if its size is small, as is the case of our SE's). Each SE cache also incorporates a *victim cache* to keep blocks which are evicted from a cache due to conflict misses. A victim cache is a small fully-associative cache [15], aiming to hold those blocks which get replaced so that they are not lost. Entry replacement in the victim cache follows a conventional replacement mechanism. When a packet is checked against an SE cache, its corresponding victim cache is also examined simultaneously. A hit, if any, happens to either the cache itself or its corresponding victim cache, but not both. The victim cache normally contains 4 to 8 entries, and it can effectively improve the hit rates by avoiding most conflict misses.

Early Cache Block Recording

To lower traffic toward FE's and the loads of FE's, a packet is recorded in the cache of an SE where a miss occurs. This early cache block recording prevents subsequent packets with the same destination from proceeding beyond the SE, and also makes it possible to deliver a packet reply back along the same path without resorting to a routing tag. COMS performance is thus enhanced and the SE design, simplified. As this recorded cache entry is not complete until its corresponding reply is back, a status bit (waiting bit, i.e., W-bit) is added to the cache entry, with the bit set until its reply is back and fills the entry. In addition, an indicator is employed to record the incoming (downstream) port from which the packet arrives at the SE, referred to as the A-bit. When a subsequent packet hits a cache entry whose W-bit is set in an SE, the packet is stopped from proceeding forward and is put in the waiting list of the arrival port. The packet is allowed to advance after the W-bit of the hit cache entry is cleared (by a reply). Once a reply comes back and hits the cache entry recorded earlier, the entry is completed with the lookup result and its W-bit is cleared. The reply is then moved to the (downstream) outgoing port indicated by the A-bit, ready for delivery backward.

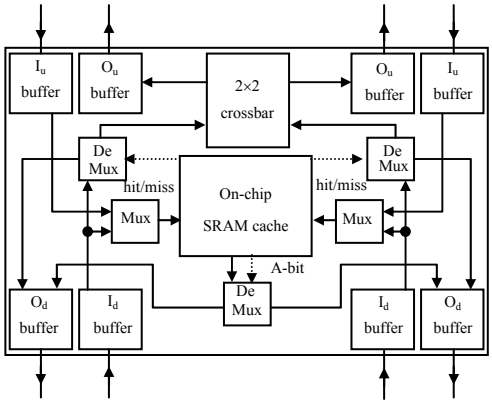

Fig. 4. Block diagram of bi-directional 2×2 SE's.

3.3 Details of SE's

The constituent SE's of COMS are bi-directional, as depicted in Fig. 4. Each incoming buffer from the downstream (denoted by I_d–buffer) can hold one packet, so can an incoming buffer from the upstream (I_u–buffer). On the other hand, each outgoing buffer to the downstream (O_d–buffer) has a capacity to hold two (2) packets, so does an outgoing buffer to the upstream (O_u–buffer). When two requests (or replies) compete for the same upstream (or downstream) link in a cycle, the two requests (or replies) are moved to the corresponding O_u–buffer (or O_d–buffer).

The cache in an SE is of on-chip SRAM and organized as a set-associative cache, with a block to hold one lookup result (i.e., <IP address, Next_hop_LC#>). When a packet arrives at the SE from the downstream, it is checked against the cache and also the associated victim cache simultaneously. This check results in different possible outcomes, as follows:

- The packet hits a cache entry with its W-bit not set. The reply is produced directly from the packet plus Next_hop_LC# found in the hit entry. The reply is then moved to the corresponding O_d–buffer (see Fig. 4). If O_d–buffer is full, the packet is left in its incoming I_d–buffer, waiting for the next cycle.
- The packet hits a cache entry with its W-bit set. The packet cannot proceed and is put in the waiting list associated with the hit entry, no matter whether the hit happens to the cache or its victim cache. After the reply for the hit cache entry comes back, its W-bit will be reset and the packet may advance (according to the above scenario).

- The packet misses in a cache (and also its corresponding victim cache). The packet takes up a cache entry, with its W-bit set. In addition, the A-bit of the entry records the downstream incoming port from which the packet arrived. This A-bit will be used for directing the reply of the packet to its appropriate downstream outgoing buffer. If this cache entry creation is to be done in a set which has no available block, one block in the set has to be chosen for replacement. The availability of a cache block is reflected by its associated I/S bit, with "I" (or "S") indicating the invalid (or shared) state. A block is available, if it is in the invalid state. Our replacement examines the L-bit and the R-bit of each block in the set to choose the block to be replaced, before resorting to a conventional replacement strategy.

When a reply (i.e., lookup result) arrives at an SE (from the upstream), it is checked against the cache (and the victim cache). This check must produce a hit, and the hit cache entry is completed by this lookup result through filling its Next_hop_LC# field and setting its L-bit or R-bit accordingly (i.e., based on the A-bit). The W-bit of the hit entry is cleared, and then the reply is forwarded to the O_d–buffer specified by the entry's A-bit. Also, replies for elements in the associated waiting list are produced and forwarded to their corresponding O_d–buffers.

3.4 Routing Table Partitioning

Routing tables in many backbone routers contain 120K+ prefixes currently, and its size is expected to grow rapidly. In fact, we have obtained one routing table with 140,838 prefixes recently [18] for our evaluation use. COMS allows each FE to hold only a subset of all prefixes, increasing the address space coverage of each FE and alleviating the memory requirement of FE's. This is realized by ***partitioning*** the set of prefixes in a routing table into subsets (of roughly equal sizes) so that each subset involves a fraction of all prefixes and constitutes a forwarding table held in one FE. The number of bits chosen for partitioning the set of prefixes depends on the number of FE's, and the bits chosen are determined by the prefixes themselves, namely, those bits which result in the size difference between the largest partition and the smallest one being *minimum*. Those same bits of the destination addresses of lookup packets are employed to control the packets routed through COMS. They separate packets across COMS into groups, one at a top output of COMS (connecting to an FE). As a larger (higher-end) router involves more FE's, each FE may carry out faster lookups and contains fewer prefixes, enjoying better address space coverage for each FE and thus increased cache hit rates and overall COMS performance.

Partitioning is done by searching for those bits which result in the size difference between the largest partition and the smallest one being *minimum*. It intends to balance and minimize the number of prefixes held in each FE, according to the routing table of a router; the partitioning is irrespective of traffic over the router and is not meant to balance the load on FE's. For a given routing table and the COMS size, however, a desirable partition often gives rise to reasonably balanced load at FE's for all traces we examined.

Note that a different technique has been considered recently [16] for partitioning the routing table into subsets for parallel search, with *all* partitioned subsets kept at each FE. Unlike COMS, the earlier design [16] forwards traffic randomly to the pool of FE's for table lookups.

4. Simulation Methodology and Results

Trace-driving simulation was adopted to evaluate the performance of scalable routers equipped with our COMS under different numbers of LC's and FE's, whose forwarding tables house subsets of prefixes obtained from partitioning the sets of prefixes (of routing tables) as stated above. This includes simulating different LC speeds and various earlier longest prefix matching algorithms under many traces available to the public.

4.1 Simulation Methodology

Our simulator takes as its input, the cache organization to be implemented in each 2×2 SE (which operates at 200 MHz), the packet streams fed to all LC's, and the longest prefix matching time per lookup in FE's. The packet streams were derived from various traces of actual packet destinations collected and posted [11, 17], one stream for each LC. For the Abilene-I, Abilene-II, and other data sets in the PMA long traces archive [11], the destinations of IP packet records (each consisting of 64 bytes) in the traces were employed as packet streams to drive our simulation studies. For the WorldCup98 data set [17], the clientID field of each request was employed to drive our simulator. Two different LC speeds were evaluated: 10 Gbps and 40 Gbps. Given a simulated LC speed of 10 (or 40) Gbps, packets of varying length are generated in such a way that on an average, they together amount to the given speed, with the mean packet length assuming to be 256 bytes and the smallest packet size equal to 40 bytes [10]. Given a trace, once a packet is generated at an LC, its destination was supplied by the trace. This approach enables one trace of addresses to feed different numbers of LC's whose packet generation processes can be specified individually and are dictated by their respective LC speeds under consideration. Meanwhile, traffic locality of the trace is reduced in this way, giving rise to pessimistic (i.e., conservative) simulation results. Each LC in our simulation produces 200,000 packets, which correspond to a time period of roughly 10 (or 40) *ms* for the mean packet length of 256 bytes under the LC speed of 40 (or 10) Gbps. This duration is so chosen since prefix changes occur some 20 times on an average and possibly up to 100 times [6] per second, and a prefix change leads to the cache contents in COMS being flushed entirely.

The cache organization in each 2×2 SE is specified by a set of parameters, including the size, the degree of set associativity, and the victim cache size. The cache block size is set to hold merely *one* lookup result. While a packet is of varying length, for the purpose of table lookups, only its header is extracted and forwarded over COMS. The packet header (referred to as the packet for short) moves from one SE to another in one cycle (of 5 *ns*) along each direction. An incoming (or outgoing) port of SE's has a buffer to hold one packet (or three packets). Routing packets to their respective target FE's follows *routing tags*, each of which comprises the bit(s) chosen from the same position(s) of the destination address of a packet. The number of bit positions chosen is $\log_2(\eta)$, where η is the number of FE's connected by COMS.

Our simulator implemented the cache operations upon each packet arrival from either direction. When a packet fails to have a hit in COMS, it reaches one FE eventually for a table lookup. Each table lookup consists of multiple memory accesses and the execution of the software code which realizes longest prefix matching. Our implementations found that the Lulea trie [2] requires 6.6 memory accesses per lookup for a large set of prefixes with 140838 entries [18], while the DP trie [1] yields about 16 memory accesses. The memory access time is assumed to be 10 *ns* and the code execution time is 100 *ns* (for executing some 100 instructions per lookup). This assumption leads to a matching search time of roughly 33 cycles (of 5 *ns* each) in FE's under the Lulea trie and of 52 cycles or so under the DP trie.

4.2 Simulation Results

The set of prefixes decides which address bits of each packet to constitute the routing tag for COMS. For COMS of size 4, the routing tag consists of bits 8 and 14. For the size of 16, the tag comprises bits 11, 13, 14, and 16. The outcomes of our extensive simulation studies confirm that typical packet streams indeed **have sufficient temporal locality** to make the COMS design effective, according to traces collected in 1998 and 2002 available to the public [11, 17]. We have simulated and gathered results of COMS for different cases: 10 Gbps & 33-cycle lookup, 10 Gbps & 52-cycle lookup, 40 Gbps & 33-cycle lookup, and 40 Gbps & 52-cycle lookup, and the results of these cases are found to follow a similar trend. In this article, we present the simulation outcomes only for the case of 40 Gbps & 33-cycle lookup, under three traces from WorldCup98, namely, D_74 (for July 8, 1998), D_78 (for July 12, 1998), D_81 (for July 15, 1998) and two traces from the Abilene-I data set in the PMA Long Traces Archive, namely, L_92-0 and L_92-1. The outcomes were obtained for associativity degree = 4, which gives rise to the smallest (or near smallest) lookup time for all the five traces demonstrated. A victim cache sized 8 is chosen for our design, as 8 blocks are found to be adequate for the victim cache in each SE.

We investigate the impact of the cache size on COMS behavior, with simulation results demonstrated in Fig. 5 – Fig. 7, where the COMS size equals 16. From Fig. 5, the hit rates exceed 90% for all traces when the cache size, β, is of no less than 1024 blocks. In addition, a larger β always leads to better hit rates, as expected.

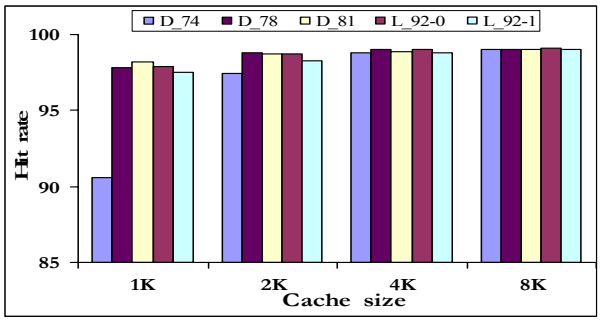

Fig. 5. Hit rate (%) versus cache size (in blocks, β).

The average time (in cycles) per lookup as a function of the cache size, β, under the five traces is depicted in Fig. 6, where a cycle equals 5 *ns* and a table lookup performed at an FE (without any hit in COMS) takes 33 cycles. For any given trace, a larger β consistently yields a shorter lookup time; with β = 4K, the mean lookup times drop below 2.8 cycles for all the traces shown, translating to a lookup speed of more than 70 millions packets per second for each line card. A COMS-based router sized 16 can thus forward more than 1120 millions packets per second, provided that $\beta \geq 4K$.

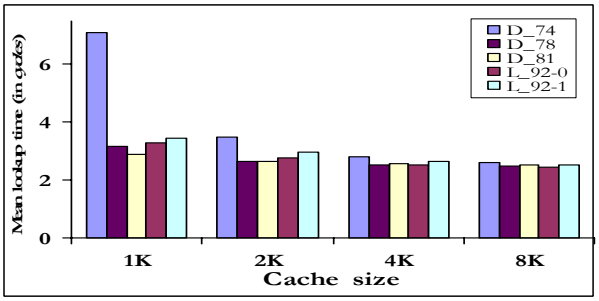

Fig. 6. Mean lookup time (in cycles) vs. cache size (β).

When compared with a current router without SE caching nor table partitioning, COMS clearly arrives at much faster lookups, according to a simple estimation next. Let the traveling time of a packet between its arrival LC to an FE in such a current router be totally ignored, then the packet lookup time in the FE on an average is 166 (= 6.6×10 + 100) *ns*, when the Lulea trie (which involves 6.6 memory accesses on an average) is implemented in the FE's, where the off-chip SRAM access time is assumed to be 10 *ns*. As a result, the current router under an optimistic assumption (of ignoring the traveling time over the switching fabric) and without regard to the queuing effect at the FE input

exhibits roughly 6 millions lookups per second per LC, in contrast to more than 70 millions for COMS of size 16, provided that $\beta \geq 4K$. When compared with an existing router, our COMS-based parallel router achieves faster mean forwarding performance by a factor more than 10, despite its reduced total SRAM amount and a possibly shorter worst-case lookup time.

It is interesting to examine a design alternative where caches in the SE's of COMS are distributed equally to all FE's, which are interconnected with LC's by a BMI without any cache incorporated. In this design alternative, there are 16 caches present in a router, and each cache has $2\times\beta$ blocks, where β is the number of cache blocks in SE's of a compatible COMS. To have a meaningful comparison, the cycle time of a BMI is assumed to be 2 *ns* (instead of 5 *ns* for COMS) in our simulation due to the absence of caches. A miss in the cache of an FE in this alternative now incurs 83 cycles. In Fig. 7, the ratio of the average lookup time in the alternative design to that in its COMS counterpart is shown as a function of the COMS cache size, β. COMS enjoys speedier lookups, except for trace D_74 under $\beta = 1024$. With the same amount of total cache capacity employed, COMS clearly outperforms its design alternative, for $\beta \geq 2K$.

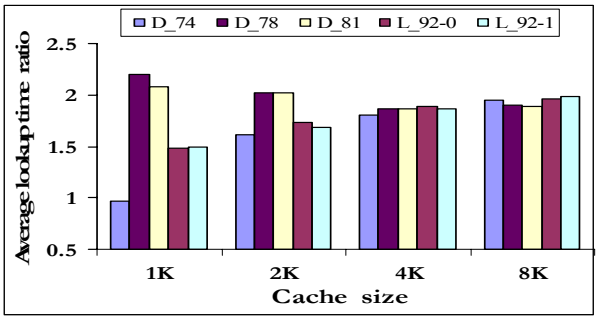

Fig. 7. Average lookup time ratio versus cache size (β).

5. Conclusion

A novel hardware-assisted design for fast packet forwarding in parallel routers has been investigated. The design incorporates a small on-chip cache, say typically $4K\times6$ bytes, in each switching element (SE) of a multistage interconnect aiming to serve as the switching fabric for interconnecting line cards (LC's) and forwarding engines (FE's) present in a parallel router. This cache-oriented multistage structure (COMS) is examined using trace-driven simulation to assess its performance measures of interest. Our simulation results under various traces demonstrate that the COMS-based router of size 16 exhibits over 10 times faster packet forwarding than its commercially available counterpart, if each SE cache contains 4K blocks. COMS achieves this substantial gain in mean forwarding performance, while usually reducing the overall SRAM amount (in both FE's and SE's combined) and possibly shortening the worst case lookup time, as a direct result of table partitioning. It is therefore ideally suitable for the new generation of parallel routers, which require fast IP packet forwarding and involve very large routing tables.

References

[1] W. Doeringer, G. Karjoth, and M. Nassehi, "Routing on Longest-Matching Prefixes," *IEEE/ACM Trans. on Networking*, vol. 4, no. 1, pp. 86-97, Feb. 1996.

[2] M. Degermark *et al.*, "Small Forwarding Tables for Fast Routing Lookups," *Proc. ACM SIGCOMM 1997 Conference*, Sept. 1997, pp. 3-14.

[3] V. Srinivasan and G. Varghese, "Fast Address Lookups using Controlled Prefix Expansion," *Proc. ACM Sigmetrics '98*, June 1998, pp. 1-11.

[4] S. Nilsson and G. Karlsson, "IP-Address Lookup Using LC-Tries," *IEEE J. on Selected Areas in Communications*, vol., no. 6, pp. 1083-1092, June 1999.

[5] T. Chiueh and P. Pradhan, "Cache Memory Design for Network Processors," *Proc. 6th Int'l Symp. on High-Performance Computer Architecture*, 2000, pp. 409-418.

[6] M. Ruiz-Sanchez, E. Biersack, and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms," *IEEE Network*, vol. 15, pp. 8-23, Mar./Apr. 2001.

[7] V. Fuller *et al.*, "Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy," RFC 1519, Internet Engineering Task Force, Sept. 1993.

[8] K. Sklower, "A Tree-Based Packet Routing Table for Berkeley Unix," *Proc. 1991 Winter Usenix Conf.*, 1991, pp. 93-99.

[9] G. Huston, "Analyzing the Internet's BGP Routing Table," *The Internet Protocol Journal*, vol. 4, no. 1, Mar. 2001.

[10] K. Thompson, G. Miller, and R. Wilder, "Wide-Area Internet Traffic Patterns and Characteristics," *IEEE Network*, vol. 11, pp. 10-23, Nov./Dec. 1997.

[11] PMA Long Traces Archive, URL – http://pma.nlanr.net/Traces/long/, Passive Measurement and Analysis, National Laboratory for Applied Network Research, Sept. 2002.

[12] H. Chan, H. Alnuweiri, and V. Leung, "A Framework for Optimizing the Cost and Performance of Next-Generation IP Routers," *IEEE J. Selected Areas in Communications,* vol. 17, pp. 1013-1029, June 1999.

[13] R. Iyer and L. Bhuyan, "Switch Cache: A Framework for Improving the Remote Memory Access Latency of CC-NUMA Multiprocessors," *Proc. 5th Int'l Symp. on High-Performance Computer Architecture*, 1999, pp. 152-160.

[14] T. Chiueh and P. Pradhan, "Cache Memory Design for Internet Processors," *IEEE Micro*, vol. 20, Jan./Feb. 2000.

[15] N. Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers," *Proc. 17th Annu. Int'l Symposium on Computer Architecture*, May 1990, pp. 364-373.

[16] M. Akhbarizadeh and M. Nourani, "An IP Packet Forwarding Technique Based on Partitioned Lookup Table," *Proc. 2002 IEEE International Conference on Communications (ICC '02)*, Apr./May 2002.

[17] WorldCup98 Dataset, URL – http://ita.ee.lbl.gov/html/contrib/WorldCup.html, The Internet Traffic Archive, Lawrence Berkeley National Laboratory, Apr. 2000.

[18] AS1221 BGP Table Data, URL – http://bgp.potaroo.net/as1221/bgp-active.html, routing table snapshot taken at 4:14pm, January 30, 2003.