The iFlow Address Processor

The iAP takes advantage of fast, wide embedded DRAM to perform fast forwarding-table lookups and statistics collection for the next generation of fast, intelligent routers. Its lookup rate of 65.5 million lookups per second allows two lookups per packet at 10 gigabits per second into a 256K entry table.

Mike O'Connor
Christopher A. Gomez
Silicon Access Networks

The Internet is growing at an astonishing rate. To keep pace with demand, the carriers that provide the Internet's backbone networks have rapidly added bandwidth. Many transport links within these networks have increased to a Synchronous Optical Network (Sonet) rate of OC-192c—roughly 10 gigabits per second (Gbps)—and OC-768 (roughly 40 Gbps) is on the horizon.[1] With this explosion in core bandwidth comes a growing expectation of higher functionality at the network's edge, where large enterprises and Internet service providers connect to the carrier networks. Carriers and ISPs are rolling out services requiring quality-of-service guarantees and detailed pay-for-use billing.

The fast, intelligent routers at peering points—where the backbone carriers, metropolitan area network providers, and large ISPs exchange data—require leading-edge accounting functionality and very high bandwidth links (10 Gbps and higher). To deploy the envisioned next-generation Internet services, providers must extend quality-of-service guarantees across these networks. Ensuring quality of service requires multiple lookups per packet into potentially large forwarding tables and associated statistics collection.[2]

In addition, these routers typically support high physical port densities by placing a number of line cards, each supporting multiple ports, together in a chassis. This leads to a general requirement for limited power consumption and board space. Furthermore, routers must be reliable, physically and under various network conditions. Routers should handle worst-case assumptions of packet characteristics, such as minimum size.

The iFlow Address Processor from Silicon Access Networks uses fast, wide, embedded dynamic RAM (DRAM) to perform lookups and statistics collection on the large forwarding tables required by the next generation of fast, intelligent routers. The chip offloads significant amounts of work from other system processors and allows lower-power, more compact network-processing solutions.

Forwarding

A router's primary function is to determine a packet's next-hop forwarding destination. The router typically bases this decision on a forwarding-table lookup of the destination address field from the packet header. Today's Internet protocol (IP) network uses a scheme called classless interdomain routing (CIDR), which organizes forwarding-table entries as a series of prefixes.[3] Each prefix has an associated mask that specifies the prefix's size and represents the network containing all the addresses beginning with the prefix. This

0272-1732/01/$10.00 © 2001 IEEE

scheme aggregates addresses in a hierarchy and does not consume global routing table entries as wastefully as the earlier, class-based addressing scheme did.

CIDR, however, requires that each lookup operation perform a longest-prefix match. Because the forwarding table can contain multiple entries matching a destination address, the lookup gives preference to the most exact match—a match of the entry with the longest matching prefix. Figure 1 shows an example of a longest-prefix match. The destination IP address 127.26.193.12 is compared to the entries in the forwarding table. The entries containing * in various positions are shorter prefixes that match any input in the position containing the *. The entries in bold are potential matches for the destination IP address, and the underlined bold entry is the longest-prefix match.

Typically, today's forwarding tables contain tens of thousands of prefixes.[4] Information, such as the egress port identifier, about the next hop to reach the destination is associated with each forwarding-table entry. In addition, the router often maintains statistics associated with each entry.

Processor overview

The iFlow Address Processor (iAP) is part of a chip set designed for 10-Gbps line cards for the fast, intelligent routers described here. A network processor or other application-specific IC uses the iAP as a coprocessor to offload much of the work associated with forwarding-table lookups. The iAP provides large CIDR forwarding tables that feature constant-time longest-prefix-match lookups with two levels of associated data supporting on-the-fly statistics updates. It offloads billions of operations per second from other processors on the line card.

The iAP holds the entire forwarding-table and statistics memory, using a total of 52 Mbits of embedded DRAM. It holds 256K (K = 1,024) prefixes up to 48 bits long, each with associated statistics. Multiple entries can combine to support prefixes up to 144 bits long. Thus, the iAP supports up to 128K prefixes up to 96 bits long or 80K prefixes up to 144 bits long. Users can organize the lookup table with any combination of prefix lengths. For instance, the table can consist of

Embedded DRAM enables high-bandwidth access to the lookup table and greater capacity than we can build with embedded SRAM.

- 128K 32-bit destination IP addresses stored as 48-bit prefixes,
- 32K 48-bit Ethernet media access control (MAC) addresses with 16-bit Virtual Private Network (VPN) identifiers stored as 96-bit prefixes, or
- 20K 128-bit flow identifiers consisting of several concatenated header fields stored as 144-bit prefixes.

The iAP provides 96 bits of per-route data for each entry in the lookup table. This 96-bit data word can include up to two counters such as byte and packet counters, for maintaining statistics. The iAP also maintains another level of associated data: per-hop data—2 Mbits of embedded DRAM organized as 8K 256-bit entries. Per-route data has a many-to-one relationship to per-hop data. Users often use per-hop data as billing data or per-port statistics. Per-hop data also typically includes forwarding-table information such as next-hop identifier and destination switch fabric identifier. It also supports two saturating counter fields. Both levels of associated data support on-the-fly counter updates—counters increment by a user-defined amount on every lookup.

Embedded DRAM enables high-bandwidth access to the lookup table. The table's

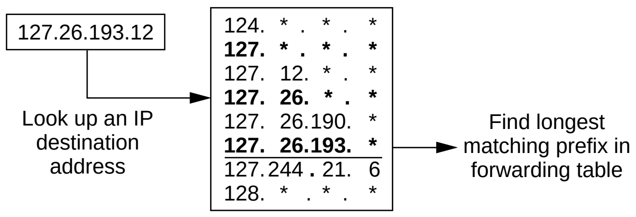

Figure 1. A longest-prefix match.

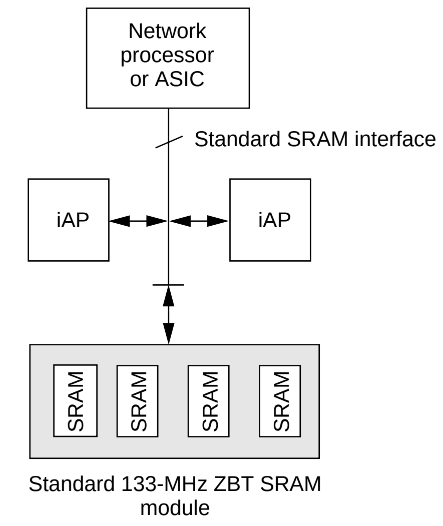

Figure 2. A typical network-processing system using the iAP.

capacity is larger than we can build with embedded static RAM—without the power, bandwidth, and board area costs of external, commodity SRAMs. The iAP supports a base lookup rate of 66.6 million per second when clocked at 133 MHz. If we account for the overhead of refreshing the on-chip DRAMs, the peak sustainable lookup rate is 65.5 million per second. This rate allows two lookups per packet at 10 Gbps, even at the worst case of 31 million 40-byte minimum-size packets per second. The iAP maintains this lookup rate regardless of key size, table fullness, or prefix length. The lookups are pipelined, and each completes in less than 200 nanoseconds.

The iAP handles table management operations as well, allowing incremental table updates such as insertions and deletions of entries. These operations proceed while lookups continue. With the table up to 90 percent full, the table management engine can perform 1 million random insertions or deletions per second, consuming only 10 percent of the lookup bandwidth. Even with massive changes to the forwarding table, lookups can continue uninterrupted at a rate of over 57 million per second. Thus, the iAP provides the reliability under worst-case network conditions that today's carriers require. Parity on all internal memories and external buses prevents silent failures, ensuring physical reliability. This feature supports highly available applications in which the system must detect failures quickly to initiate failover recovery by redundant hardware.

Interface

Figure 2 shows a typical system using the iAP. The iAP interfaces with network processors through a standard pipelined or zero-bus-turnaround (ZBT) synchronous SRAM bus. This high-bandwidth bus is present on most off-the-shelf network processors. To the network processor, the iAP appears to be a standard SRAM. The network processor writes to or reads from the iAP using standard SRAM protocols. Writes specify requests and provide data, and reads collect results and check status.

Users can configure the bus as 32, 64, or 96 bits wide, and it operates at up to 133 MHz. Additional iAP chips or standard SRAMs can also use the same bus. At 133 MHz, the SRAM bus interface supports four iAPs, providing an aggre-

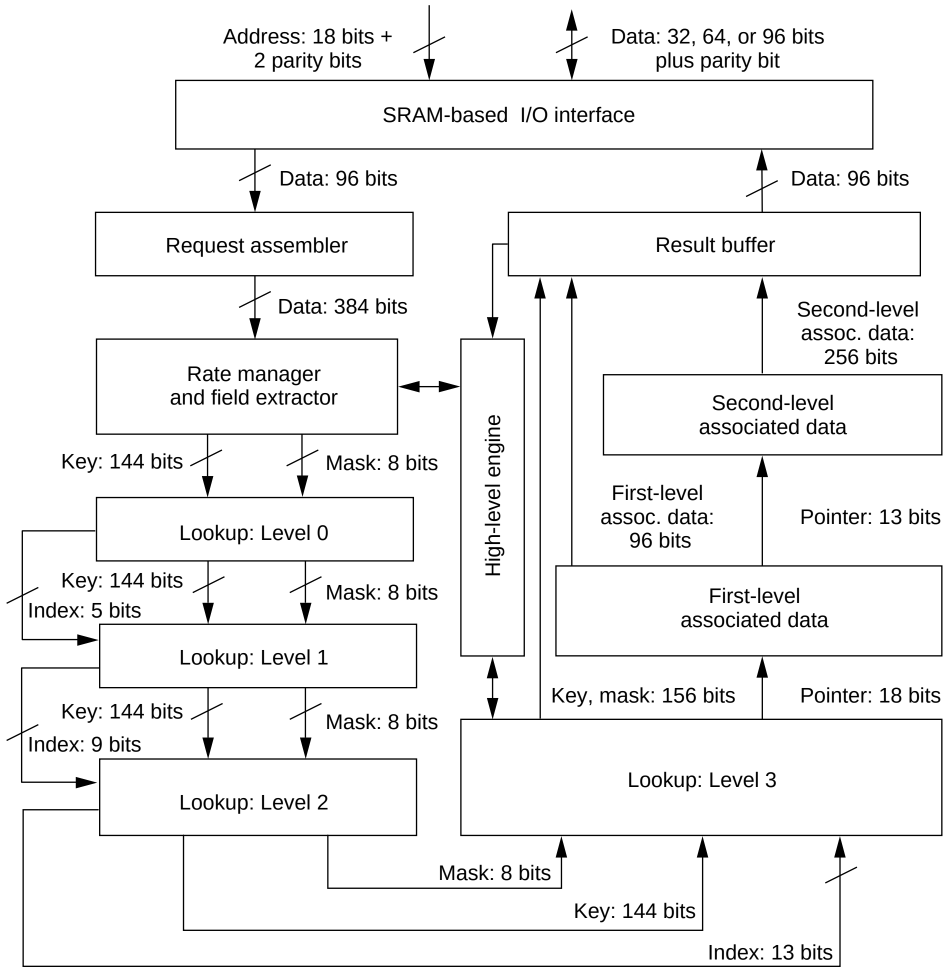

Figure 3. Organization of the iAP.

Other matching techniques

In addition to the iFlow Address Processor's tree-based technique, longest-prefix-matching approaches include tries, hash tables, and ternary content-addressable memories (CAMs).

Tries are a class of data structures commonly used for longest-prefix matching. The forwarding engine traverses these data structures using bits of the input key to specify the path to the lookup result. Two frequently used types are the Patricia (practical algorithm to retrieve information coded in alphanumeric)[1] and Luleä[2] tries. A central routing CPU generates these tries on the basis of a routing table and subsequently downloads them to SRAM on each line card. The Patricia and Luleä algorithms enable fast lookups, but they are not well suited for dynamic learning or incremental updates. They also require multiple memory accesses to perform a lookup, taxing the bandwidth of any bus to off-chip memory.

Hashing algorithms map a larger number space such as an IP address or MAC address into a smaller number space such as a memory index. They do this by applying a function to the larger key to generate the smaller index. This technique provides nearly constant-time lookups, but it causes hash collisions, in which two different numbers in the larger address space hash to the same value in the smaller number space. Collisions require additional memory accesses to resolve the lookup. Several hashing techniques exist,[3] but they generally require more memory space than the iAP algorithm.

Ternary CAMs support very fast constant-time lookups with no risk of collision and relatively inexpensive incremental updates.[4] Because the CAMs must contain two bits and a comparator per memory cell, they require more of the die area, thus limiting capacity and increasing power requirements over other solutions.

Each of these techniques handles only the longest-prefix match and does not address statistics processing and associated-memory bandwidth requirements. Therefore, the system must manage statistics separately, consuming additional computing cycles and memory bandwidth.

References

1. D. Morrison, "PATRICIA—Practical Algorithm to Retrieve Information Coded in Alphanumeric," *J. ACM*, vol. 15, no. 4, Oct. 1968, pp. 514-534.
2. M. Degermark et al., "Small Forwarding Tables for Fast Routing Lookups," *Proc. ACM Sigcomm*, ACM Press, New York, 1997, pp. 3-13.
3. M. Waldvogel et al., "Scalable High Speed IP Routing Lookups," *Proc. ACM Sigcomm*, ACM Press, New York, 1997, pp. 25-37.
4. D. Shah and P. Gupta, "Fast Updating Algorithms for TCAMs," *IEEE Micro,* vol. 21, no. 1, Jan./Feb. 2001, pp. 36-47.

gate capacity of over 1 million 48-bit prefixes.

Many network processors are multithreaded, using multiple contexts to process packets simultaneously. The iAP is designed for these processors, which use many contexts to hide lookup latency. The iAP allows up to 64 outstanding requests by supporting 64 independent request contexts.

Organization

Figure 3 diagrams the iAP's organization. SRAM write transactions go to the request assembler. This block assembles requests requiring multiple bus transactions, such as lookups of 144-bit keys, before launching them into the rate manager. The request assembler can assemble up to 384 bits into a single request. The rate manager places each request in one of three first-in, first-out queues (FIFOs) on the basis of the request's command type and user-specified priority.

The rate manager schedules requests from the three FIFOs into the pipeline. One FIFO issues high-priority requests (most requests). Another FIFO issues low-priority requests such as register reads and table maintenance helper commands. The third FIFO issues table maintenance requests. The rate manager schedules the FIFOs so that low-priority requests and table maintenance operations wait for high-priority requests. This scheduling is subject to user configuration, which can guarantee a certain amount of service to each FIFO to prevent starvation. The rate manager also issues DRAM refresh commands periodically down the pipeline.

A request issued by the rate manager goes to the field extractor, which extracts the key, statistics increment values, and other data from 384 bits of data provided by the request assembler. Each field's location is user configurable, and the field extractor provides default values for fields not sent by the user to the iAP. Thus, a user need not send the full 384 bits possible for every request. If a lookup requires only a 32-bit key, it requires only one 32-bit bus transaction.

Lookup procedure

We developed the iAP's lookup algorithm to provide fast, pipelined, constant-time longest-prefix match operations. The algorithm exploits the possibilities of very wide embedded memories. It is similar to lookup techniques described by Lampson, Srinivasan, and Varghese.[5] (See the "Other matching techniques" sidebar.)

The lookup table stores prefixes in sorted order, which treats a shorter prefix as greater than a longer prefix that begins with the same bits. Thus, the prefix 11101* is greater than the prefix 1110111*. The first step in finding the longest-prefix match is to find the smallest prefix in the table that is greater than or equal to the lookup key. The iAP uses a straightforward three-level B-tree[6] to perform this search. The three levels determine the correct row of the leaf level in which to find the first prefix longer than the lookup key.

Levels 0 through 2 of the lookup pipeline form the B-tree's three levels. Level 0 (L0) holds 31 prefixes. The iAP compares each of the 31 prefixes with the lookup key in parallel, and selects the index, 0 through 30, of the smallest prefix greater than or equal to the lookup key. If all prefixes in L0 are less than the lookup key, the iAP generates an index of 31. L1 contains 32 rows indexed by the result of the L0 lookup. Each row holds 15 prefixes. The L1 block performs 15 parallel comparisons and generates one of 16 possible results. The L0 and L1 results are concatenated to form a 9-bit index to one of 512 rows in the L2 memory. Each row contains 15 prefixes, again compared in parallel with the lookup key, with 16 possible results. Finally, the results of L0, L1, and L2 together refer to one of 8,192 rows in the L3 table.

These first three levels of SRAM-based memories total 1.2 Mbits. Each memory is logically over 2,000 bits wide to facilitate the wide parallel comparisons. To generate an index into the L3 memory, the first three levels require a total of four cycles, which can overlap other lookups in a pipelined manner.

Once it identifies the row containing the smallest prefix greater than or equal to the lookup key, the iAP reads that row from the large DRAM block holding the table in L3. In L3, up to 32 prefixes may be present in each row. Generally, however, it's preferable that each row have fewer than 32 prefixes because a distribution of "holes" throughout the table facilitates the quick insertion of additional prefixes without extensive shifting of entries.

The iAP compares each of the 32 potential prefixes with the lookup key in parallel and selects the first prefix that is a valid match. Because the prefixes are sorted with longer prefixes first, the first match in this comparison will be the longest prefix match. Unfortunately, it is possible that the comparison will find no match. Therefore, the table stores additional information in each row. The iAP uses this information to determine the correct match when none of the prefixes in the selected row are the longest prefix match.

If there is no match in a row, the longest prefix match is guaranteed to be a shorter version of the longest prefix in the indexed row. Therefore, each row maintains a list of up to 32 prefixes in the table that are shorter than the maximum prefix and that have the upper bits in common. With this information, the iAP can compare a lookup key that doesn't match a prefix in the indexed row with the maximum prefix in a row with successively fewer significant bits, until a match is found. The fixed maximum of 32 shorter prefix lengths per row limits the levels of nested prefixes possible to no more than 32. This places no practical restrictions on the tables, however. The DRAM access and comparisons require five cycles, but these can overlap other lookups in a pipelined manner.

Figure 4 illustrates two lookups in a smaller version of the iAP. Rather than the 48-bit prefix configuration, this version is a two-way L0, two-way L1, two-way L2, and four-way L3 configuration with up to 8-bit prefixes. The first lookup key, 00100101, is less than or equal to the first L0 prefix, 101*. This causes the iAP to select the first row of L1. The lookup key is less than or equal to the first L1 prefix of the first row, causing the first row of L2 to be selected. The lookup key is greater than the first prefix of the first row of L2 (001000*); thus, the second row of L3 is selected. The first and second prefixes of the second row of L3 (001001* and 00100*) are valid matches for the input key, but the smallest prefix (the first prefix, 001001*) is selected and is the longest-prefix match.

The second example, with lookup key 11000000, proceeds similarly through the L0, L1, and L2 memories, selecting the fifth row of L3. This row contains only a single prefix that is not a valid match for the input key. Associated with each row is a checklist of valid shorter prefixes. In this case, the prefix lengths

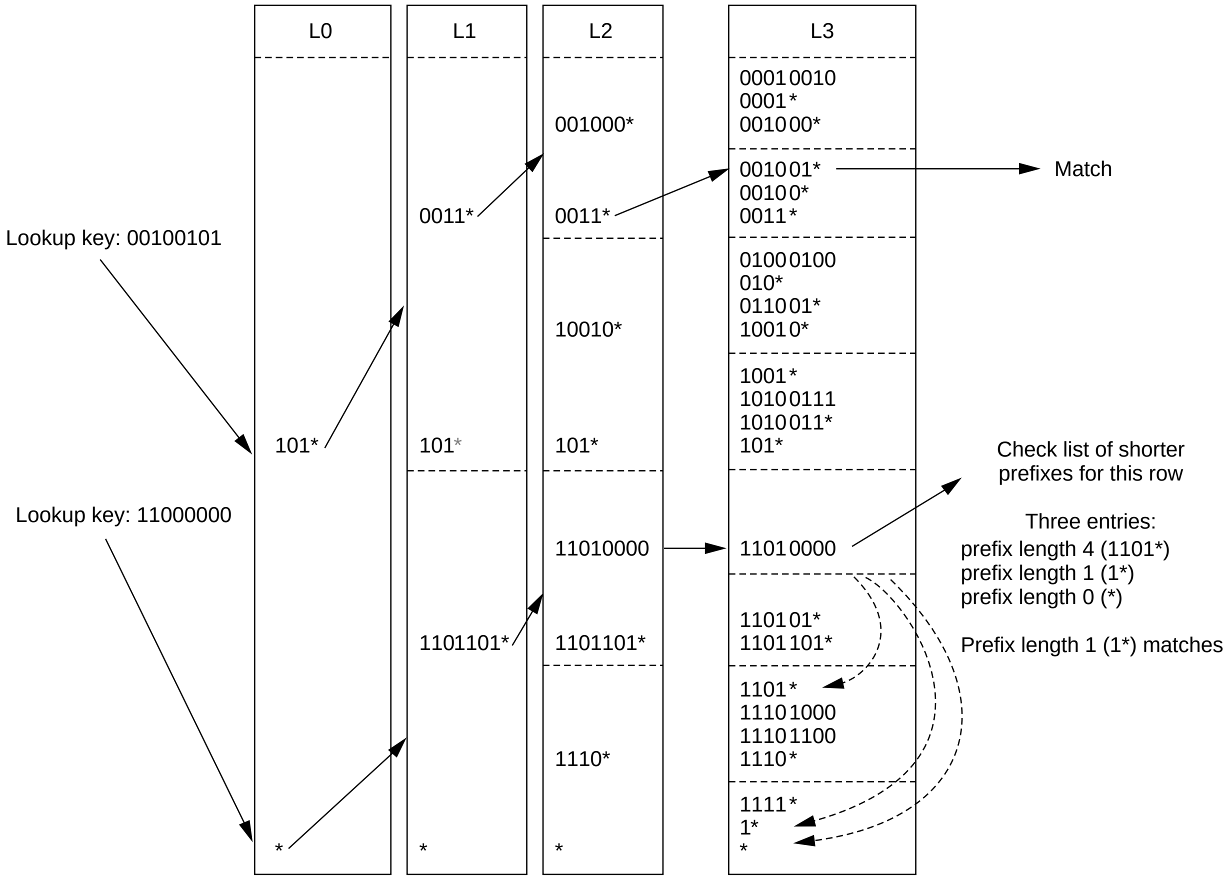

Figure 4. Two lookup examples.

4, 1, and 0 are associated with the selected row. The first valid match is with prefix length 1. Thus, the longest-prefix match for the input key is prefix 1*.

Maintenance operations

The integrated high-level engine state machine maintains the L0 through L3 table contents, handling insertions, deletions, learning, and other maintenance operations while lookups continue. The state machine maintains the table in proper order, with appropriate L0 through L2 indexing and correct shorter-prefix lists. Maintaining the shorter-prefix lists for each row can require many operations for an insertion or deletion. For instance, if the default key (*, prefix length 0) is inserted, every row must be updated to include prefix length 0 in its list. Fortunately, in practice, expensive operations like this are extremely rare. The state machine also dynamically rebalances the table, distributing holes evenly, to maintain good insertion performance. Approximately 20 percent of the cycles for each insertion serve this purpose.

Associated data

Each prefix in the table has an 18-bit pointer to the first level of associated data. After finding a matching prefix, the iAP uses this pointer to select a 96-bit row from the first-level associated memory. The iAP increments two counter fields in the data by values the user provides with the lookup request. Optionally, one counter field can be an autoincrementing field. Both counter fields can be configured to saturate or wrap when they overflow. The data from the first-level associated data contains a 13-bit pointer to one of 8,192 rows of the second-level associated memory. The second-level associated data also contains two counter fields, which are updated in the same way as the first level.

The iAP writes the results of the lookups

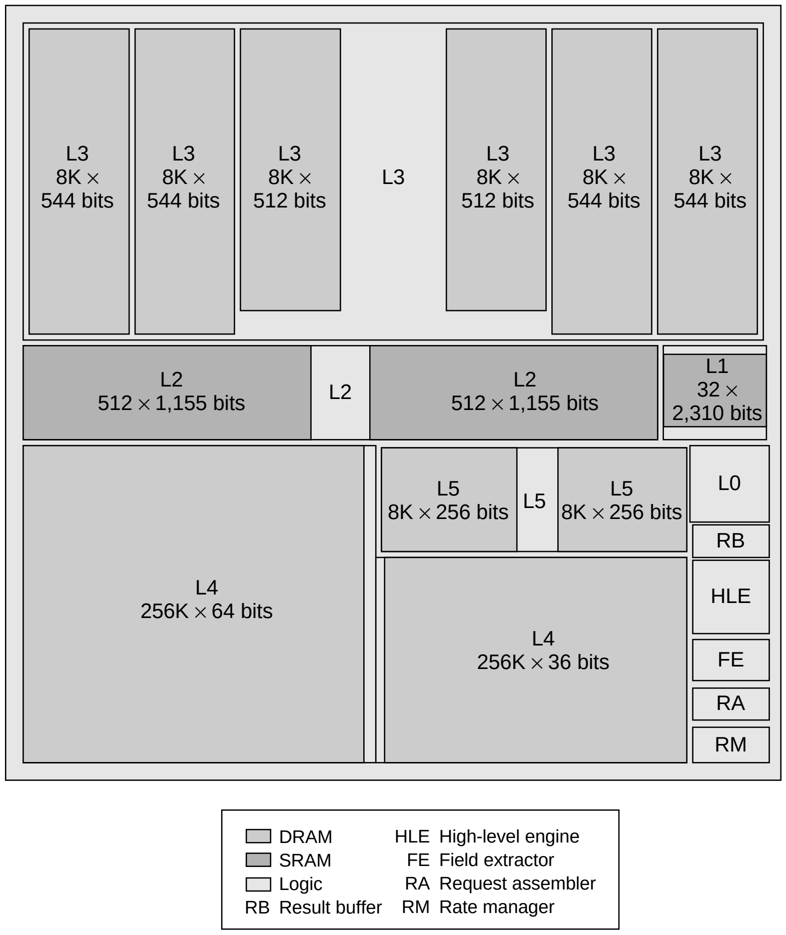

Figure 5. The iAP's layout.

and associated-memory accesses into the result buffer. The result buffer contains 64 outstanding request contexts, allowing multithreaded processors to read out results in any order. Host processors read request results by issuing SRAM read transactions and providing a context identifier to select the result.

Embedded DRAM

The iAP uses embedded DRAM for the third level of the lookup pipeline and the first and second associated-data levels. Silicon Access Networks custom-designed the embedded DRAM arrays, using the 0.18-micron Taiwan Semiconductor Manufacturing Company (TSMC) embedded DRAM cell.[7]

The DRAM array for the lookup pipeline's third level consists of 8,192 rows, each 3,200 bits wide. The memory has a 66-MHz cycle time, performing a new random access read every 15 nanoseconds. This very wide DRAM enables the iAP to simultaneously access all the prefixes in a row, with its associated shorter-prefix list and data pointers.

The DRAM array for the first-level associated data consists of 256K rows, each 100 bits wide—96 bits of associated data and 4 parity bits. The second-level associated memory consists of 8,192 rows of 256 bits. Parity is stored in a separate, small SRAM. For each lookup, the iAP must read a row from the associated memory, update the statistics, and write back the result into the associated memory. Therefore, the associated memory arrays operate at 133 MHz, performing a random access read or write every 7.5 nanoseconds. These very fast DRAMs provide performance similar to that of external SRAMs, but they avoid external SRAMs' costly pins and power requirements. The total aggregate bandwidth provided by the iAP's internal DRAM arrays is more than 260 Gbps.

Physical characteristics

We fabricated the iAP in TSMC's 0.18-micron embedded DRAM logic process. The 52 Mbits of embedded DRAM occupies roughly 70 percent of the die area. In addition, the iAP contains about 1.1 million logic gates. The die size is 14.8 × 12.8 mm. The processor dissipates approximately 5 W at 133 MHz. Figure 5 shows the iAP's layout.

By combining fast, wide embedded DRAM with a lookup algorithm that exploits the DRAM's capabilities, the iFlow Address Processor allows forwarding and switching to scale to 10 Gbps line speeds and beyond. The iAP's integration of billing and statistics accounting with forwarding-table lookups frees the other processors in the system from burdens that would otherwise require additional processing and memory bandwidth. By managing a large table and offloading billions of operations per second, the iAP frees the network processor for operations such as classification, policing, metering, rate shaping, scheduling, and queuing. It also reduces the network processor's memory bandwidth requirements, permitting lower-cost, lower-power options with fewer pins. In addition to forwarding-table lookups, the iAP provides fast constant-time exact matching (a special case of longest-prefix matching) and fast learning, making it well suited for network layer 2 switching applications. MICRO

References

1. "UUNet Network Maps," http://www.uunet.com/network/maps.
2. G. Armitage, *Quality of Service in IP Networks*, Macmillan Technical Publishing, Indianapolis, Ind., 2000.
3. V. Fuller, et al., "Classless Inter-Domain Routing (CIDR): An Address Assignment and Aggregation Strategy," Internet Engineering Task Force Request for Comments 1519, Sept. 1993, http://www.ietf.org/rfc.html.
4. "Internet Routing Table Trends," http://www.merit.edu/ipma/trends.
5. B. Lampson, V. Srinivasan, and G. Varghese, "IP Lookups Using Multiway and Multicolumn Search," *IEEE/ACM Trans. Networking*, vol. 7, no. 3, 1999, pp. 324-334.
6. T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*, MIT Press, Cambridge, Mass., 1990.
7. "TSMC Embedded DRAM," http://www.tsmc.com/technology/dram.html.

Mike O'Connor is the director of advanced architecture at Silicon Access Networks, where he is the chief architect of the iFlow Address Processor. He is responsible for designing a family of chips supporting high-speed next-generation networking equipment. O'Connor received an MS in electrical engineering from the University of Texas at Austin and a BSEE from Rice University. He holds 21 patents and is a member of the IEEE Computer Society.

Christopher A. Gomez is a staff architect at Silicon Access Networks, where he is responsible for the architecture and microarchitecture of next-generation network-processing elements.. His interests include computer architecture, network architecture, and architectural modeling and verification. Gomez is a member of the IEEE Computer Society.

Send questions and comments to Mike O'Connor, Silicon Access Networks, 211 River Oaks Parkway, San Jose, CA 95134; mike.oconnor@siliconaccess.com.