Reducing Dynamic Power Dissipation in Pipelined Forwarding Engines

Weirong Jiang and Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California, Los Angeles, CA 90089, USA
{*weirongj, prasanna*}*@usc.edu*

Abstract— **Power consumption has become a limiting factor in next-generation routers. IP forwarding engines dominate the overall power dissipation in a router. Although SRAM-based pipeline architectures have recently been developed as a promising alternative to power-hungry TCAM-based solutions for high-throughput IP forwarding, it remains a challenge to achieve low power. This paper proposes several novel architecture-specific techniques to reduce the dynamic power consumption in SRAM-based pipelined IP forwarding engines. First, the pipeline architecture itself is built as an inherent cache, exploiting the data locality in Internet traffic. The number of memory accesses which contribute to the majority of power consumption, is thus reduced. No external cache is needed. Second, instead of using a global clock, different pipeline stages are driven by separate clocks. The local clocking scheme is carefully designed to exploit the traffic rate variation and improve the caching performance. Third, a fine-grained memory enabling scheme is developed to eliminate unnecessary memory accesses, while preserving the packet order. Simulation experiments using real-life traces show that our solutions can achieve up to 15-fold reduction in dynamic power dissipation, over the baseline pipeline architecture that does not employ the proposed schemes. FPGA implementation results show that our design sustains 40 Gbps throughput for minimum size (40 bytes) packets while consuming a small amount of logic resources.**

I. Introduction

The primary function of network routers is to forward packets based on the results of IP lookup which matches the destination IP address of the packet to the entries in the routing table. As the network traffic grows rapidly, IP forwarding becomes a major performance bottleneck for network routers [1], [2]. For example, current backbone link rates have been pushed beyond OC-768 (40 Gbps) rate, which requires a throughput of 125 million packets per second (MPPS) for minimum size (40 bytes) packets. Meanwhile, as routers achieve aggregate throughputs of trillions of bits per second, power consumption by forwarding engines becomes an increasingly critical concern in backbone router design [3], [4]. Some recent investigation [5], [6] shows that power dissipation has become the major limiting factor for next generation routers and predicts that expensive liquid cooling may be needed in future routers. According to the analysis by [5], almost 2/3 of power dissipation inside a core router is due to IP forwarding engines.

Ternary Content Addressable Memories (TCAMs), where a single clock cycle is sufficient to perform an IP lookup, dominate today's high-end routers. However, as a result of the massive parallelism inherent in their architecture, TCAMs do not scale well in terms of clock rate, power consumption, or chip density [2]. It is estimated that the power consumption per bit of TCAMs is on the order of 3 micro-Watts, which is 150 times that of Static Random Access Memories (SRAMs) [3]. On the other hand, most SRAM-based algorithms can be implemented as some form of tree traversal, such as trie-based algorithms [7], which can be pipelined to achieve a high throughput of one packet per clock cycle [2]. Though SRAM-based pipeline solutions are being considered as attractive alternatives to TCAMs [2], they still suffer from high power consumption [8], due to the large number of memory accesses.

This paper exploits several characteristics of Internet traffic and of the pipeline architecture, to reduce the dynamic power consumption of SRAM-based IP forwarding engines. First, as observed in [2], Internet traffic contains large amount of locality, where most packets belong to few flows. By caching the recently forwarded IP addresses, the number of memory accesses can be reduced so that power consumption is lowered. Unlike previous caching schemes most of which need an external cache attached to the main forwarding engine, we integrate the caching function into the pipeline architecture itself. As a result, we do away with complicated cache replacement hardware and eliminate the power consumption of the "hot" cache [9]. Second, since the traffic rate varies from time to time, we freeze the logic when no packet is input. We propose a local clocking scheme where each stage is driven by an independent clock and is activated only under certain conditions. The local clocking scheme can also improve the caching performance. Third, we note that different packets may access different stages of the pipeline, leading to a varying access frequency onto different stages. Thus we propose a fine-grained memory enabling scheme to make the memory in a stage to sleep when the incoming packet is not accessing it. Our simulation results show that the proposed schemes can reduce the power consumption by up to 15-fold. We prototype our design on a commercial field programmable gate array (FPGA) device and show that the logic usage is low while the backbone throughput requirement (40 Gbps) is met.

The rest of the paper is organized as follows. Section II gives a brief overview of the router architecture and introduces SRAM-based pipelined IP forwarding engines. Section III analyzes the traffic characteristics using real-life

This work is supported by the United States National Science Foundation under grant No. CCF-0702784. Equipment grant from Xilinx Inc. is gratefully acknowledged.

traffic traces and presents our motivation. Section IV details our architecture. Section V evaluates the performance of our solution. Section VI reviews the related work and compares with our schemes. Section VII concludes the paper.

II. Background

A. Router Architecture

As shown in Figure 1, a router contains two main architectural components: routing engine and IP forwarding engine. The routing engine processes the routing protocol and produces the forwarding table. The IP forwarding engine receives packets, looks up the destination IP address against the forwarding table to identify the next-hop information, and forwards the packets to the identified next-hop port. The routing engine and the IP forwarding engine perform their tasks independently, although they constantly communicate through high-throughput links [10].

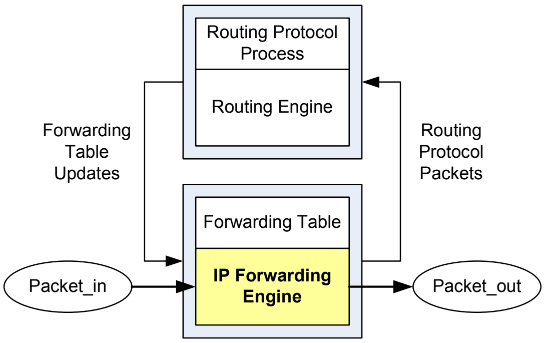
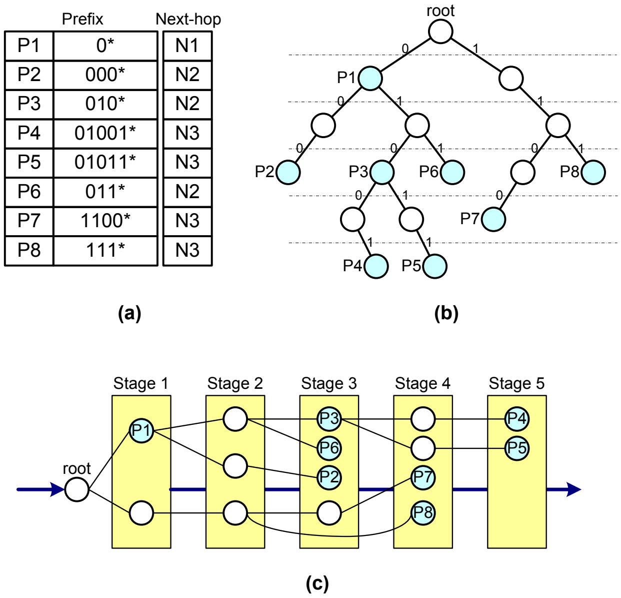


Fig. 1. Block diagram of the router system architecture

B. Pipelined IP Forwarding Engines

The entries in the forwarding tables are specified using prefixes, and the nature of IP forwarding is longest prefix matching (LPM). The most common data structure in algorithmic solutions for LPM is some form of trie [7]. A trie is a binary tree, where a prefix is represented by a node. The value of the prefix corresponds to the path from the root of the tree to the node representing the prefix. The branching decisions are made based on the consecutive bits in the prefix. A trie is called a uni-bit trie if only one bit at a time is used to make branching decisions. Figure 2 (b) shows the uni-bit trie for the prefix entries in Figure 2 (a). Each trie node contains both the represented prefix and the pointer to the child nodes.

Given a uni-bit trie, LPM is performed by traversing the trie according to the bits in the IP address. The information about the longest prefix matched so far, which is updated when meeting a node containing a prefix, is carried along the traversal. Thus, when a leaf is reached, the longest matched prefix along the traversed path is returned. The time to look up a uni-bit trie is equal to the prefix length. The use of multiple bits in one scan can increase the search speed. Such a trie is called a multi-bit trie. For simplicity, we consider only the uni-bit trie in this paper, though our ideas can be applied to other types of trie [1].



	Prefix	Next-hop
P1	0*	N1
P2	000*	N2
P3	010*	N2
P4	01001*	N3
P5	01011*	N3
P6	011*	N2
P7	1100*	N3
P8	111*	N3

Fig. 2. (a) Prefix entries; (b) Uni-bit trie; (c) Mapping the trie onto a linear pipeline using the fine-grained node-to-stage mapping scheme [2].

Pipelining can dramatically improve the throughput of trie-based solutions. A straightforward way to pipeline a trie is to assign each trie level to a separate stage, so that a lookup request can be issued every clock cycle. However, this results in unbalanced memory distribution across the pipeline stages, which has been identified as a dominant issue for SRAM-based pipeline architectures [11]. In an unbalanced pipeline, more time is needed to access the larger local memory. This leads to a reduction in the global clock rate. Furthermore, since it is unclear at hardware design time which stage will be the fattest, we must allocate memory with the maximum size for each stage. Such an over-provisioning results in memory wastage and excessive power consumption [11]. Although various pipeline architectures [11], [12] have been proposed recently, most of them balance the memory distribution across stages at the cost of throughput degradation. Our previous work [2] proposes a fine-grained node-to-stage mapping scheme for linear pipeline architectures. It allows the two nodes on the same level to be mapped onto different stages. Balanced memory distribution across stages is achieved, while a high throughput of one packet per clock cycle is sustained. Figure 2 (c) shows the fine-grained mapping result for the uni-bit trie in Figure 2 (b).

III. Analysis and Motivation

We obtained four backbone Internet traffic traces from the Cooperative Association for Internet Data Analysis (CAIDA) [13]. The trace information is shown in Table I, where the numbers in the parenthesis are the ratio of the number of unique destination IP addresses to the total number of packets in each trace.

A. Traffic Locality

According to Table I, regardless of the length of the packet trace, the number of unique destination IP addresses

TABLE I
REAL-LIFE IP HEADER TRACES

Trace	Date	# of packets	# of unique IPs
equinix-chicago-A	20090219	460448	31923 (6.93%)
equinix-chicago-B	20090219	2811616	182119 (6.48%)
equinix-sanjose-A	20080717	3473762	233643 (6.73%)
equinix-sanjose-B	20080717	2200188	115358 (5.24%)

is always much smaller than that of the packets. These results coincide with those previous work on Internet traffic characterization [14]. Due to TCP burst, some destination IP addresses can be connected very frequently in a short time span. Hence caching has been used effectively in exploiting such traffic locality to either improve the IP forwarding speed [14] or help balance the load among multiple forwarding engines [2]. This paper employs the caching scheme to reduce the number of memory accesses so that the power consumption can be lowered.

B. Traffic Rate Variation

We analyze the traffic rate in terms of the number of packets at different time. The results for the four traces are shown in Figure 3, where the X axis indicates the time intervals and the Y axis the number of packets within each time interval. As observed in other papers [15], the traffic rate varies from time to time. Although the router capacity is designed for the maximum traffic rate, power consumption of the IP forwarding engine can be reduced by exploiting such traffic rate variation in real life.

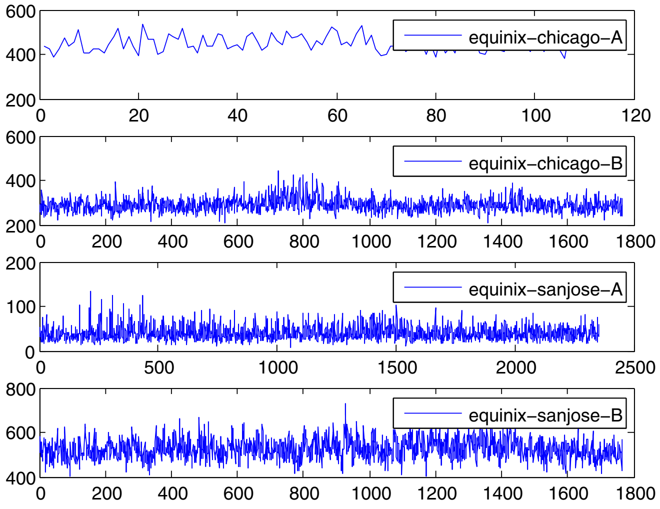


Fig. 3. Traffic rate variation over the time.

C. Access Frequency on Different Stages

The unique feature of the SRAM-based pipelined IP forwarding engine is that different stages contain different sets of trie nodes. Given various input traffic, the access frequency to different stages can vary a lot. For example, we used a backbone routing table from the Routing Information Service (RIS) [16] to generate a trie, mapped the trie onto a 25-stage pipeline and measured the total number of memory accesses on each stage for the four input traffic traces. The results are shown in Figure 4, where the access frequency of each stage is calculated by dividing the number of memory accesses on each stage by that on the first stage. The first stage is always accessed by any packet while the last few stages are seldom accessed. According to this observation, we should disable the memory access in some stages when the packet is not accessing the memory in that stage.

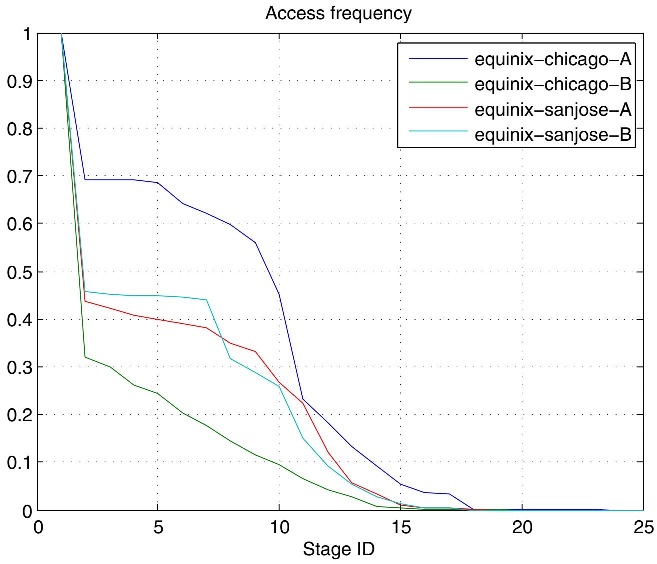


Fig. 4. Access frequency on each stage.

IV. ARCHITECTURE

We propose a caching-enabled SRAM-based pipeline architecture for power-efficient IP forwarding, as shown in Figure 5. Let H denote the pipeline depth i.e. the number of stages in the pipeline. These H stages store the mapped trie nodes. Every time the architecture receives a packet, the incoming packet compares its destination IP address with the packets which are already in the pipeline. It will be considered as “cache hit” if there is a match, even though the packet has not retrieved the next-hop information yet. To preserve the packet order, the packet having a cache hit still goes through the pipeline. However, no memory access is needed for this packet, so that the power consumption for this packet is reduced.

A. Mapping Trie onto Pipeline

The pipeline depth is bounded by the trie height i.e., the maximum directed distance from the root to a leaf in the trie. We use the similar idea as [2] to map a trie onto a linear pipeline while balancing the memory distribution across stages. The pipeline depth is $H = 25$. Each node stored in the local memory of a pipeline stage has three fields: (1) the pointer to the represented prefix and its next-hop information, (2) the distance to the pipeline stage where the child node is stored, and (3) the memory address of its child node in the pipeline stage where the child node is stored.

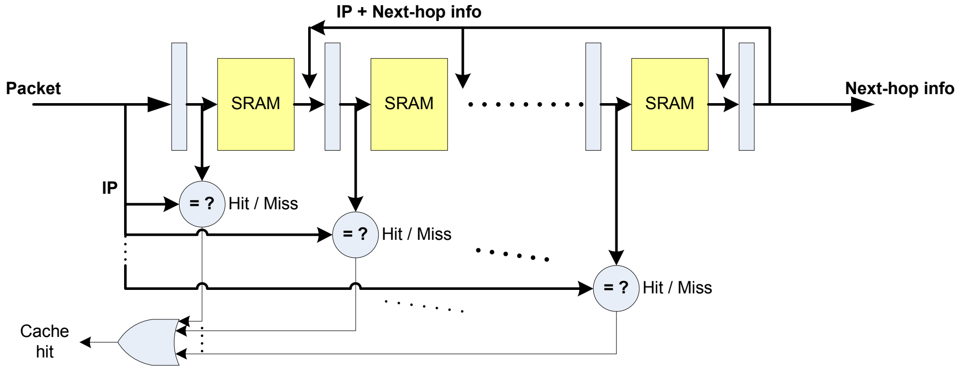


Fig. 5. Pipeline with inherent caching

When a packet is passed through the pipeline, the distance value is decremented by 1 when it goes through a stage. If the distance value becomes 0, the child node's address is used to access the memory in that stage.

B. Inherent Caching

Most existing caching schemes need to add an external cache to the forwarding engine. However, the cache itself can be power-intensive [9] and also needs extra logic to support cache replacement. The relatively long pipeline delay can also result in low cache hit rates in traditional caching schemes [2]. Our architecture implements the caching function without appending extra caches. As shown in Figure 5, the pipeline itself acts as a fully associative cache, where the existing packets in all the stages are matched with the arriving packet. If the arriving packet (denoted Pkt_{new}) matches a previous packet (denoted Pkt_{exist}) existing in the pipeline, Pkt_{new} has a cache hit even though Pkt_{exist} has not retrieved its next-hop information. Then Pkt_{new} will go through the pipeline with the cache hit signal set to '1'. On the other hand, Pkt_{new} does not obtain the next-hop information until Pkt_{exist} exits the pipeline. As shown in Figure 5, the packet exiting the pipeline will forward its IP address as well as its retrieved next-hop information to all the previous stages. The packets in the previous stages compare with the forwarded IP address. The packet matching the forwarded IP address will take the forwarded next-hop information as its own, and carry the retrieved next-hop information along when traversing the rest of the pipeline.

C. Local Clocking

Most of the existing pipelined IP lookup engines are driven by a global clock. The logic in a stage is active even when there is no packet to be processed. This results in unnecessary power consumption. Furthermore, since the pipeline keeps forwarding the packets from one stage to the next stage at the highest clock frequency, the pipeline will contain few packets if the traffic rate is low. Since the pipeline is built as a cache which is dynamic and sensitive to input traffic, few packets in the pipeline indicates a small number of cached entries, which results in low cache hit rate.

To address this issue, we propose a local clocking scheme where each stage is driven by an individual clock. Only one constraint must be met to prevent any packet loss:

Constraint 1: If the clock of the previous stage is active and there is a packet in the current stage, the clock of the current stage must be active.

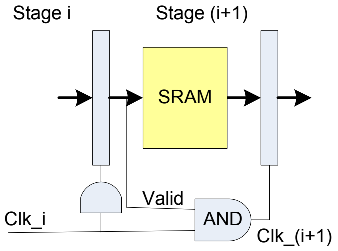


Fig. 6. Local clocking for one stage.

Hence, we design the local clocking as shown in Figure 6. The clock of a stage will not be active until the stage contains a valid packet and its preceding stage is forwarding some data to the current stage. To prevent clock skew, some delay logic is added in the data path of the clock signal of the previous stage. In a real implementation, we do not use the AND gate which may result in glitches. Instead, we use clock buffer primitives provided by Xilinx design tools [17].

D. Fine-Grained Memory Enabling

As discussed in Section III-C, the access frequency to different stages within the pipeline varies. Current pipelined IP forwarding engines keep all memories active for all the packets, which results in unnecessary power consumption. Our fine-grained memory enabling scheme is achieved by gating the clock signal with the read enable signal for the memory in each stage. Only when the packet goes to access the memory in the current stage, the read enable signal becomes active. In other words, the read enable signal will remain inactive in any of the following four cases: (1) there is no packet arriving; (2) the distance value of the arriving packet is larger than 0; (3) the packet has already retrieved its next-hop information; (4) the cache hit signal carried by the packet is set to '1'.

V. Performance Evaluation

A. FPGA Implementation

We prototyped our design on FPGA using Xilinx ISE 10.1 development tools. The target device was Xilinx Virtex-5 XC5VFX200T with -2 speed grade. Post place and route results showed that our design achieved a clock frequency of 126 MHz while consuming a small amount of the logic resources, as shown in Table II [1]. Such a clock frequency resulted in throughput of 40.32 Gbps for minimum size (40 bytes) packets, which meets the current backbone network rate.

TABLE II
Resource utilization

	Used	Available	Utilization
Number of Slices	748	30,720	2%
Number of bonded IOBs	73	960	7%
Number of Block RAMs	295	456	64%

B. Power Model

The average dynamic power consumption of a pipelined IP forwarding engine can be modeled as Equation (1), where p denotes the packet to be forwarded, H the pipeline depth, $N(p)$ the number of packets, and $Power_{memory}(i,p)$ and $Power_{logic}(i,p)$ denote the dynamic power consumption of the memory and of the logic in the ith stage accessed by p, respectively.

$$Power = \frac{\sum_p \sum_{i=1}^{H} [Power_{memory}(i,p) + Power_{logic}(i,p)]}{N(p)} \qquad (1)$$

We profiled the power consumption of the memory and of the logic based on our FPGA implementation results. Using the XPower Analyzer tool provided by Xilinx, we obtained the dynamic power consumption for our design, as shown in Figure 7. As we expected, the power consumption by memory dominated the overall power dissipation of the pipelined IP forwarding engine.

C. Evaluation Using Real-Life Traces

Based on the profile data of the power consumption in the architecture, we developed a cycle-accurate simulator for our pipelined IP forwarding engine. We conducted the experiments using the four real-life backbone traffic traces given in Table I, and evaluated the overall power consumption.

First, we examined the impact of the fine-grained memory enabling scheme. We disabled both inherent caching and local clocking, and then ran the simulation under two different conditions: (1) without fine-grained memory enabling (2) with fine-grained memory enabling. Figure 8 compares the results which are normalized by being divided by the results with condition (2). According to Figure 8, fine-grained memory enabling can achieve up to 12-fold reduction in power consumption.

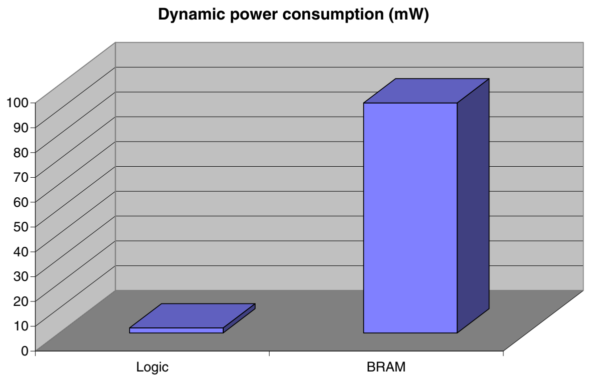


Fig. 7. Profiling of dynamic power consumption in a pipelined IP forwarding engine

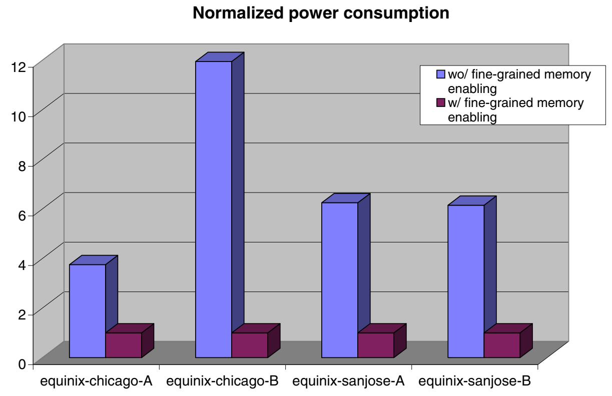


Fig. 8. Power reduction with fine-grained memory enabling

Second, we evaluated the impact of the inherent caching and the local clocking schemes. We enabled the fine-grained memory enabling scheme, and then ran the simulation under three different scenarios: (1) without either inherent caching or local clocking; (2) with inherent caching but without local clocking; (3) with both schemes. The results are shown in Figure 9 where the results without both schemes are set to be the baseline. Without local clocking, caching reduced the power consumption very slightly, because of the low cache hit rate (e.g. 1.65% for the trace equinix-chicago-B). Local clocking improved the cache hit rate (e.g. the cache hit rate for the trace equinix-chicago-B increased to 45.9%), which resulted in higher reduction in power consumption.

Overall, when all the three proposed schemes were enabled, the architecture achieved 6.3, 15.2, 8.1, and 7.8 -fold reduction in power consumption, for the four traffic traces, respectively.

VI. Related Work

Power-efficient IP forwarding engines have recently been an active research topic. However, to the best of our knowledge, there is little work done for pipelined IP forwarding engines. Some early work on TCAM-based engines [8] proposes various schemes to partition a forwarding table into several blocks and perform IP lookup on one of the

[1] Due to the limitation of the size of on-chip memory, our design supported 70K prefixes which was 1/4 of the current largest backbone routing table. However, our architecture can be extended by using external SRAMs.

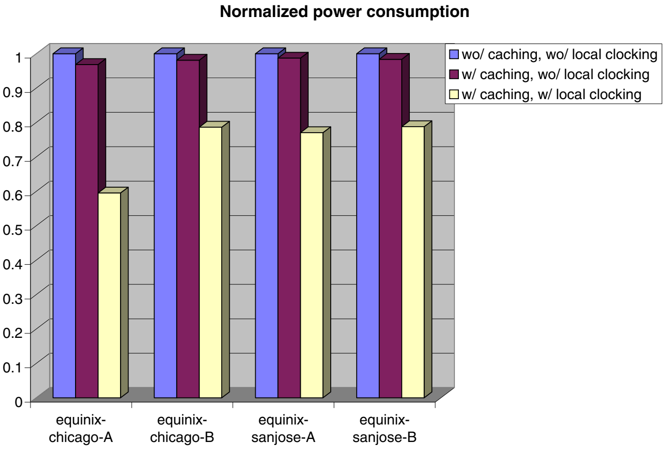


Fig. 9. Power reduction with inherent caching and local clocking

blocks. The aim is to ensure each block contains equal numbers of prefixes, so that the power consumption reduction is maximized. Similar ideas can be also applied for SRAM-based multi-pipeline architectures [18]. Those partitioning-based solutions do not consider the traffic characteristics and are orthogonal to our solutions.

Kaxiras et al. [19] propose a SRAM-based approach called IPStash for power-efficient IP forwarding. IPStash replaces the full associativity of TCAMs with set associative SRAMs to reduce power consumption. However, the set associativity depends on the forwarding table size and thus may not be scalable. For large-scale forwarding tables, the set associativity is still large, resulting in low clock rate and high power consumption.

Most of existing caching schemes are used to improve the throughput rather than the power efficiency. The only work we found on using caching to reduce power consumption is [20] where some trie nodes are cached to skip the access to the deeper trie levels. However, such a scheme requires an external cache which results in extra power consumption. The packets with cache hit will be output out of order. Since the solution is based on software-based implementation, it cannot meet the backbone network link rate.

Traffic rate variation has been exploited in some recent papers for reducing power consumption in multi-core processor based IP forwarding engines. In [21] clock gating is used to turn off the clock of unneeded processing engines of multi-core network processors to save dynamic power when there is a low traffic workload. Dynamic frequency and voltage scaling are used in [15] and [22], respectively, to reduce the power consumption of the processing engines. However, those schemes require large buffers to store the input packets so that they can determine or predict the traffic rate. The large packet buffers result in high power consumption. Also, these schemes do not consider the latency for the state transition which can result in packet loss in case of burst traffic.

Although some motivations of our paper have been exploited in previous papers, our schemes are tailored and specific for pipeline architectures. Both traffic locality and traffic rate variation are well exploited at minimum cost.

VII. Conclusion

This paper proposed several novel architecture-specific techniques to reduce the dynamic power dissipation in SRAM-based pipelined IP forwarding engines. First, the pipeline was built as an inherent cache, exploiting effectively the traffic locality with minimum overhead. Second, a local clocking scheme was proposed to exploit the traffic rate variation as well as to improve the caching performance. Third, a fine-grained memory enabling scheme was used to eliminate unnecessary memory accesses for the input packets. Simulation using real-life traffic traces showed that our solution achieved up to 15-fold reduction in dynamic power dissipation. FPGA implementation results showed that our design consumed a small amount of logic resources while satisfying the current backbone network link rate.

References

[1] W. Eatherton, G. Varghese, and Z. Dittia, "Tree bitmap: hardware/software IP lookups with incremental updates," *SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 97–122, 2004.

[2] W. Jiang and V. K. Prasanna, "Parallel IP using multiple SRAM-based pipelines," in *Proc. IPDPS*, 2008, pp. 1–14.

[3] D. E. Taylor, "Survey and taxonomy of packet classification techniques," *ACM Comput. Surv.*, vol. 37, no. 3, pp. 238–275, 2005.

[4] M. Gupta and S. Singh, "Greening of the Internet," in *Proc. SIGCOMM*, 2003, pp. 19–26.

[5] A. M. Lyons, D. T. Neilson, and T. R. Salamon, "Energy efficient strategies for high density telecom applications," *Princeton University, Supelec, Ecole Centrale Paris and Alcatel-Lucent Bell Labs Workshop on Information, Energy and Environment*, June 2008.

[6] J. Chabarek, J. Sommers, P. Barford, C. Estan, D. Tsiang, and S. Wright, "Power awareness in network design and routing," in *Proc. INFOCOM*, 2008, pp. 457–465.

[7] M. A. Ruiz-Sanchez, E. W. Biersack, and W. Dabbous, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, vol. 15, no. 2, pp. 8–23, 2001.

[8] F. Zane, G. J. Narlikar, and A. Basu, "CoolCAMs: Power-efficient TCAMs for forwarding engines." in *Proc. INFOCOM*, 2003, pp. 42–52.

[9] C. Zhang, "A low power highly associative cache for embedded systems," in *Proc. ICCD*, 2006.

[10] Juniper Networks T1600 Core Router, "http://www.juniper.net."

[11] F. Baboescu, D. M. Tullsen, G. Rosu, and S. Singh, "A tree based router search engine architecture with single port memories," in *Proc. ISCA*, 2005, pp. 123–133.

[12] S. Kumar, M. Becchi, P. Crowley, and J. Turner, "CAMP: fast and efficient IP lookup architecture," in *Proc. ANCS*, 2006, pp. 51–60.

[13] Colby Walsworth, Emile Aben, kc claffy, Dan Andersen, "The caida anonymized 2009 internet traces," *http://www.caida.org/data/passive/passive_2009_dataset.xml*.

[14] H. Liu, "Routing prefix caching in network processor design," in *Proc. ICCCN*, 2001, pp. 18–23.

[15] A. Kennedy, X. Wang, Z. Liu, and B. Liu, "Low power architecture for high speed packet classification," in *Proc. ANCS*, 2008.

[16] RIS Raw Data, "http://data.ris.ripe.net."

[17] Xilinx Virtex-5 FPGAs, "http://www.xilinx.com."

[18] W. Jiang and V. K. Prasanna, "Towards green routers: Depth-bounded multi-pipeline architecture for power-efficient IP lookup," in *Proc. IPCCC*, 2008, pp. 185–192.

[19] S. Kaxiras and G. Keramidas, "IPStash: a set-associative memory approach for efficient IP-lookup," in *INFOCOM*, 2005, pp. 992–1001.

[20] L. Peng, W. Lu, and L. Duan, "Power Efficient IP Lookup with Supernode Caching," in *Proc. Globecom*, 2007.

[21] Y. Luo, J. Yu, J. Yang, and L. N. Bhuyan, "Conserving network processor power consumption by exploiting traffic variability," *TACO*, vol. 4, no. 1, 2007.

[22] M. Mandviwalla and N.-F. Tzeng, "Energy-efficient scheme for multiprocessor-based router linecards," in *Proc. SAINT*, 2006, pp. 156–163.