Architecture and Performance Models for Scalable IP Lookup Engines on FPGA*

Yi-Hua E. Yang
Xilinx Inc.
Santa Clara, CA
edward.yang@xilinx.com

Yun Qu*
Dept. of Elec. Eng.
Univ. of Southern California
yunqu@usc.edu

Swapnil Haria**
BITS, Pilani
Rajasthan, India
swapnilster@gmail.com

Viktor K. Prasanna*
Dept. of Elec. Eng.
Univ. of Southern California
prasanna@usc.edu

***Abstract*—We propose a unified methodology for optimizing IPv4 and IPv6 lookup engines based on the balanced range tree (BRTree) architecture on FPGA. A general BRTree-based IP lookup solution features one or more linear pipelines with a large and complex design space. To allow fast exploration of the design space, we develop a concise set of performance models to characterize the tradeoffs among throughput, table size, lookup latency, and resource requirement of the IP lookup engine. In particular, a simple but realistic model of DDR3 memory is used to accurately estimate the off-chip memory performance. The models are then utilized by the proposed methodology to optimize for high lookup rates, large prefix tables, and a fixed maximum lookup latency, respectively. In our prototyping scenarios, a state-of-the-art FPGA could support (1) up to 24 M IPv6 prefixes with 400 Mlps (million lookups per second); (2) up to 1.6 Blps (billion lookups per second) with 1.1 M IPv4 prefixes; and (3) up to 554 K IPv4 prefixes and 400 Mlps with a lookup latency bounded in 400 ns. All our designs achieve 5.6x – 70x the energy efficiency of TCAM, and have performance independent of the prefix distribution.**

***Index Terms*—IPv4, IPv6, IP lookup, FPGA**

I. Introduction

Designing a scalable IP lookup engine remains a challenging problem due to the longer prefix length, larger prefix table, and higher lookup rate required by the growing Internet. IPv4 with 32-bit address length is running out of free address space; IPv6 has been developed to overcome the address shortage by extending the address length to 128 bits.[1] The size of the prefix tables in the Internet core routers continues to grow exponentially at 1.13x (IPv4) and 1.64x (IPv6) per year.[2] Furthermore, today's data center and core routers need to handle data traffic at 100 to 400 gigabits per second (Gbps); with 64-byte packets, these data rates translate to IP lookup rates of 200 to 800 million lookups per second (Mlps). The higher IP lookup rate in turn requires a higher (random) memory access rate to the large prefix tables.

* Supported by U.S. National Science Foundation under grant CCR-1018801.
** Participation sponsored by the Viterbi-India 2012 Program

[1]The IPv6 Addressing Architecture (RFC 4291) specifies IPv6 unicast address to consist of a (64-bit) subnet prefix followed by a (64-bit) interface ID. However, an IPv4-embedded IPv6 address can have a longer-than-64 bits routing prefix (RFC 6052).

[2]Based on the number of active BGP entries since 2005 (IPv4) and 2007 (IPv6) obtained from the BGP Analysis Reports [1].

Various IP lookup architectures have been proposed in the past with promising performance improvements. The range tree architecture is found to scale well to both high lookup rates and large prefix tables [2], [3], [4], [5], [6], [7]. State-of-the-art VLSI chips, such as Field-Programmable Gate Arrays (FPGAs), can be built with massive amount of on-chip and off-chip memory resources and allow the construction of highly complex IP lookup solutions. On the other hand, designing scalable IP lookup on VLSI/FPGA is complicated by many design-time and run-time issues. First, the massive on-chip and off-chip resources exhibit complex performance tradeoffs, making optimizations difficult. Second, a robust IP lookup engine should be evaluated with guaranteed worst-case performance, making cache-based optimizations undesirable. Third, the Internet is expected to evolve and expand in both structure and composition; thus IP lookup should achieve performance independent of network-specific statistics. Last, realistic but simple models are needed to allow accurate design-time exploration, particularly for the off-chip (DDR3) memory access.

With these challenges in mind, we develop a unified optimization methodology for IPv4 and IPv6 lookup engines based on the balanced range tree (BRTree) architecture on FPGA. By modeling the architecture with various design parameters and resource requirements in a concise set of equations, a broad range of design tradeoffs and system performance can be explored quickly and accurately at design time. To the best of our knowledge, our work is the first to propose a unified set of performance models for fast design space exploration of IPv4 and IPv6 lookups on FPGA. In summary:

- We propose a flexible BRTree pipeline architecture for scalable IP lookup on FPGA. Both IPv4 and IPv6 are supported with a unified set of design parameters.
- We develop accurate resource and performance models for the on-chip logic, memory, and off-chip DDR3 components used by the proposed architecture.
- Using the performance models, we provide prototype design scenarios for table size, lookup rate and latency, under different performance requirements.
- We evaluate the resulting IP lookup solutions on state-of-the-art FPGA.

The rest of the paper is organized as follows: Section II

summarizes background and prior work. Section III describes the BRTree pipeline architecture. Section IV details the performance models with various design parameters and resource constraints, which are used in Section V to optimize three prototype design scenarios. Section VI discusses performance comparisons and Section VII concludes the paper.

II. Background

A. Related Work

A TCAM-based architecture for IPv4/v6 lookup is proposed in [8], [9]. The TCAM-based IP lookup architecture does not depend on prefix distribution and is easy to design, but it is expensive and power-hungry as well. As an alternative solution, algorithms for IPv6 lookup on FPGA have involved trie-based, hash-based and tree-based approaches.

In trie-based approach [10], [11], [12], the path from the trie root to each "match" node in the trie represents the bit string of a prefix. In [11], compression is employed to remove redundancy in trie nodes. In [12], only the next-hop bit map and the offset value are stored in each node. In general, trie-based lookup requires memory bandwidth proportional to the prefix length. In hash-based approaches [13], [14], [15], [16], hash functions or bloom filters are used to improve memory efficiency. In particular, the Concise Lookup Tables (CoLT) [17] is a unified hash table-based architecture which handles both IPv4 and IPv6 lookups simultaneously. The FlashTrie [15] proposes a hash-based trie architecture for both high lookup rate and large IPv4 and IPv6 prefix tables. However, these architectures provide point solutions with various high level assumptions on hardware resource and prefix distribution; as such, it is more difficult to scale up their performance for larger prefix table or longer prefix length (e.g. 64 or 128-bit IPv6) given more available resources.

Various modifications of tree-based IP lookup schemes have been developed. In [3], IP lookup is performed using a multi-way range tree algorithm, which achieves a worst-case search and update time of $O(\log N)$. The range trie data-structure was extended in [5] to support both LPM and incremental updates. [6] introduces a recursive balanced multi-way range tree. A different tree-based IP lookup architecture is developed in [7] where specialized prefix matching and prefix partitioning are employed to minimize the total memory requirement.

B. Range Tree-based IP Lookup

Range trees have been proposed as efficient and robust data structures for storing and searching prefix tables [2], [4]. In a range tree, each routing prefix is converted to a *range* in the address space, where all overlapping ranges are "flattened" to a set of *subrange boundaries* (SB). On average, each routing prefix requires at least one and at most two SBs. A *subrange* between any two adjacent SBs stores is identified by the next-hop information (NHI) of the longest prefix covering that subrange. A balanced range tree (BRTree) is constructed from the set of SBs and NHIs.

IP lookup with BRTree has robust size and throughput performance. For any set of N prefixes, there are at least $N+1$ and at most $2N$ flattened subranges. The height of the BRTree, is at most $\log N$. Both the memory space and the lookup time of a BRTree depend only on the prefix table size N. Another advantage of BRTree is its scalability to large tables and long prefixes. The table size grows exponentially with tree levels, while longer prefixes can be handled efficiently by wider key comparison without issuing separate memory accesses.

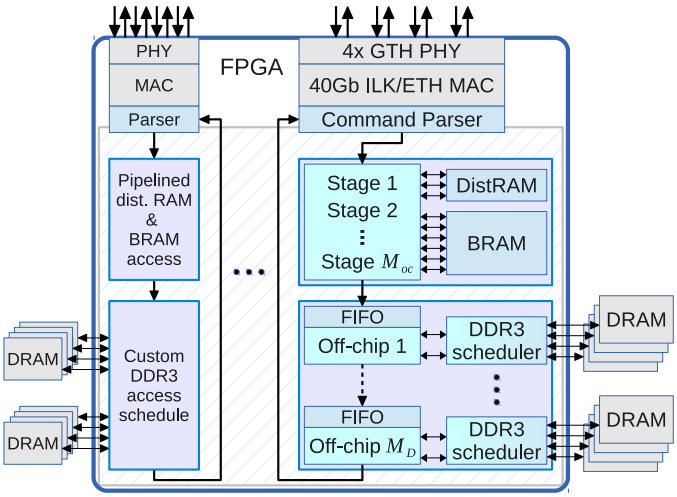

Figure 1: High-level architecture of the proposed scalable IP lookup engine on FPGA.

III. Scalable IP Lookup Architecture on FPGA

A. Architecture Overview

Figure 1 shows the overall architecture of the proposed scalable IP lookup engine on FPGA. Specifically, the FPGA serves as an offload engine for accelerating IP lookup with large prefix tables. The engine consists of multiple parallel pipelines. All pipelines can be used to lookup the same prefix table, or each a different one, depending on the run-time performance requirement. Each pipeline receives commands and returns results through a 40 Gbps Interlaken or Ethernet MAC over 4x GTH transceivers. Each lookup command consists of an 8-bit virtual table ID, a 32 or 64-bit IP address, and a 24-bit local sequence number; each lookup result consists of a 16-bit NHI and the local sequence number. The 40 Gbps serial I/O supports up to 426 million IPv6 lookup commands per second.

In this paper, we focus on scaling and optimizing the hachured part of Figure 1. Once a command is parsed, its IP address is sent through the on-chip and off-chip stages for pipelined BRTree lookup. The on-chip stages access successively larger distributed (distRAM) and block (BRAM) memory on FPGA. For large prefix tables, the memory size of the largest (last) few tree levels grows proportionally. Off-chip DDR3 memories with customized scheduling for guaranteed bandwidth are used for these levels (see Section III-C below).[3]

[3]Theoretically, both SRAM and DRAM are valid technologies to be used for the off-chip memory. In practice, DRAM is highly preferred, since it consumes ~1/10 the power and costs ~1/100 the price per bit of SRAM.

Table I: Memories accessed by a BRTree pipeline.

Memory types	On-chip distRAM	On-chip BRAM	Off-chip DRAM
Organization	LUT-based SRAM	Medium-sized block SRAM	Individual DDR3 chips
Typical size	$4 \sim 17$ Mbits	$16 \sim 66$ Mbits	$1 \sim 8$ Gbits

Table I shows a summary of the three types of memories accessed in the proposed architecture.

B. Balanced Range Tree Pipeline

We adopt the balanced range tree (BRTree) approach in Section II-B as the basis of our IP lookup algorithm. The BRTree is fully pipelined where every level is accessed from a dedicated memory stage. With dual-port distRAM and BRAM, up to $2K$ IP lookup operations can be issued to K BRTree pipelines per clock cycle. Parallel DDR3 buses with replicated data are used to achieve high read bandwidth from off-chip DRAM, as to be discussed in Section III-C.

For simple prototyping, on-chip BRTree nodes are binary only, although in general a higher node degree can be used. Off-chip BRTree nodes are generally non-binary since every DDR3 access involves a burst of 128 or 256 bits of data transfer. An internal BRTree node stores one or more subrange boundaries (SBs). A leaf BRTree node stores multiple ($n \geq 1$) SBs and ($n + 1$) next-hop information (NHIs).

The BRTree pipeline employs *superpipelining*, which further pipelines each stage in 4 cycles: (1) preparing local address, (2) accessing local memory, (3) comparing returned keys, and (4) multiplexing output to the next stage. Additional FIFO buffers, as shown in Figure 1, are used in the off-chip stages to cope with the longer DDR3 access latency. At 200 MHz, a 20-deep FIFO is sufficient for up to 100 ns of DDR3 access latency.

C. Guaranteed DDR3 Access Rate

State-of-the-art DRAM (e.g. DDR3) achieves very high maximum bandwidth but exhibits complex and address-dependent access behaviors. In many cases, the actual (random access) bandwidth is significantly lower than the maximum (sequential access) bandwidth. Our approach achieves a sustained access rate by (1) replicating the same data in all banks within a DDR3 chips, and (2) accessing each bank according to a simple but accurate DDR3 timing model.

Internally, a DDR3 chip is organized in 8 banks, each composed of a large array of rows ($n = 8\text{K}, 16\text{K}, 32\text{K}$) and columns ($m = 1\text{K}, 2\text{K}$). A fixed number of data bits ($w = 4,\ 8,\ 16$) are located at every valid [bank, row, column] address. This width determines the *bus width* of the DDR3 chip. In principle, every read or write to the DDR3 chip initiates a burst of 8 sequential w-bit data.

Non-sequential DDR3 accesses are subject to mainly four timing requirements. A row must be opened (activated) prior to its access. It is fast to access an open row in the current bank (t_{CCD}); it is slightly slower to activate and access a row in a different bank (t_{RRD}); it is much slower to access a different row in the same bank (t_{RC}), where the open row must be closed (precharged) before any new one can be activated. In addition, thermal and power restrictions require a minimum t_{FAW} for any 4 row activations. In summary [18]:

- t_{CCD} Minimum time between successive accesses to the same bank and row, $\simeq 5$ ns.
- t_{RRD} Minimum time between successive row activations to different banks, $\simeq 7.5$ ns.
- t_{RC} Minimum time between successive row activations to the same bank, $\simeq 50$ ns.
- t_{FAW} Minimum time to perform 4 row activations in different banks, $\simeq 40$ ns.

Ideally, a maximum of $1/t_{CCD} \simeq 200$ Maccs/sec can be achieved when successively accessing the same bank and row. In practice, the *worst-case* access rate can be as low as $1/t_{RC} \simeq 20$ Maccs/sec. One way to avoid the long t_{RC} delay between two successive accesses is to replicate the same data in every bank and perform accesses to all 8 banks in a round-robin fashion, where the maximum guaranteed DDR3 read rate can be described by the following formula:[4]

$$\min\left(\frac{1}{t_{RRD}}, \frac{8}{t_{RC}}, \frac{4}{t_{FAW}}\right) \simeq 100\,\text{Maccs/sec}$$

Using DDR3-1600 memory, each read can return up to $\frac{1600\text{ M}}{100\text{ M}} \times w = 16w$ bits, which defines the maximum node size of our BRTree levels in the DDR3 stages.

Note that these timing constraints affect the DDR3 access *throughput*, while the *latency* between a DDR3 read/write request and its response is usually much higher (up to 100 ns). This fixed latency can be handled by buffering all pending requests in a FIFO before their responses are received. Furthermore, the above access schedule trades off memory size efficiency and write throughput for a higher guaranteed read throughput. It is different from and not to be confused with the scheduling made by a DDR3 controller which reorders access requests to minimize the average memory latency.

IV. Resource and Performance Modeling

The performance of the BRTree pipeline can be described by three main metrics: table size N, lookup rate R, and lookup latency T. At design time, these performance metrics can be modeled in three types of parameters: the problem specifications (which could also include the performance metrics themselves), the BRTree design parameters, and the resource usage and constraints.

A. Problem Specifications

- N No. prefixes in the prefix table.
- R Minimum lookup rate.
- T Maximum lookup latency.
- b No. bits per subrange boundary (SB).
- h No. bits per next-hop information (NHI).

[4] In particular, all the timing constraints were not realistically considered in [15], even though a similar bank replication was used.

N, R and T can be specified as performance requirements. In practice, at least one and usually two of them are pre-defined, leaving the other(s) to be optimized. State-of-the-art IP lookup engines have $N \geq 1$ Mpfx (million prefixes), $R \geq 200$ Mlps (million lookups per second), and $T \leq 1$ us.

Both b and h are specified by the IP lookup problem. b is 32 for IPv4 and 64 (or 128) for IPv6. h is router dependent; in our case, $h = 16$, but in general it can be as low as 8 and as high as 36.

B. BRTree Design Parameters

K	No. BRTree pipelines.
M_C	No. on-chip memory stages.
M_D	No. off-chip memory stages.
L	Degree of superpipelining.
W_{dist}	Bits per access from distRAM.
W_B	Bits per access from BRAM.
W_D	Bits per access from DRAM.

The seven design parameters specify the "shape" of a BRTree pipeline discussed in Section III-B. K is usually between 1 and 5; a very large K would produce a very wide multipipeline with sub-optimal performance. In general, M_C can be high (10~20) due to the high on-chip memory bandwidth available on FPGA; M_D is limited by the number of available I/O pins (see parameter d in the next section). In our design, $L = 4$ accounts for the number of clock cycles required for a packet to go through the logic part of each stage.

W_{dist} can be flexible as it is supported by the fine-grain look-up tables (LUTs) on FPGA. W_B is usually a multiple of 36 (Virtex) or 40 (Stratix). W_D is upper-bounded by $16w$, w is 8 or 16, as discussed in Section III-C.

C. Resource Usage and Constraints

s_{dist}	Total no. bits available in distRAM.
s_B	Total no. bits available in BRAM.
d	Total no. DDR3 data buses available.
s_{last}	Total no. bits available in the last DDR3 stage.
f_C	Clock rate of the on-chip pipeline.
f_D	Access rate of the DDR3 data bus.
t_D	Latency of each DDR3 access.

The largest Virtex 7 FPGA (XC7VX1140T) has $s_{dist} \leq 17$ Mbits and $s_B \leq 66$ Mbits. The 1,100 user I/O pins on the largest Virtex 7 package can support about 20 DDR3 data buses; thus $d \leq 20$. State-of-the-art DDR3 supports up to 8 Gbits per chip; thus $s_{last} \leq 20 \times 8 = 160$ Gbits.

With a good superpipelined design, f_C can be > 200 MHz. Assuming DDR3-1600, $f_D = 100$ Maccs/sec and $t_D \leq 100$ ns, as discussed in Section III-C.

D. Performance Models

1) Table size: As discussed in Section III-B, each routing prefix requires at most one NHI and two SBs. In other words, $2n$ pairs of [SB, NHI] can define at least n and at most $2n$ routing prefixes.

Assume only on-chip memories are used. The maximum table size N (per pipeline) is at least half the number of [SB, NHI] pairs stored in the $(s_{dist} + s_B)/K$ memory size:

$$N \geq \frac{1}{2} \times \frac{s_{dist} + s_B}{K(b+h)}, \quad M_D = 0 \tag{1}$$

Suppose $M_D \geq 1$ off-chip stage is used. The on-chip memory can now hold up to $N_C = (s_{dist} + s_B)/b$ SBs, allowing access to $N_C + 1$ ($\simeq N_C$) nodes in the first off-chip stage. The number of leaf nodes is also limited by $s_{last} / \left(8W_D \left\lceil \frac{2f_C}{f_D} \right\rceil\right)$; the factor 1/8 is needed since data is replicated in all 8 banks of each DDR3 chip, as discussed in Section III-C; another factor of $\left\lceil \frac{2f_C}{f_D} \right\rceil$ data replication is required in DDR3 to account for its access rate difference from the (dual-port) on-chip memory. Therefore,

$$N \geq \frac{1}{2K} \min\left(\frac{s_{dist} + s_B}{b} N', \frac{s_{last}}{8W_D \left\lceil \frac{2f_C}{f_D} \right\rceil} \right) \cdot v \tag{2}$$

where

$$N' = \left(\left\lfloor \frac{W_D}{b} \right\rfloor + 1 \right)^{M_D - 1}, \; M_D \geq 1 \tag{3}$$
$$v = \left\lfloor \frac{W_D + b}{b + h} \right\rfloor$$

N' is the factor by which the number of leaf nodes increases with the $M_D - 1$ additional off-chip stages; v is the number of NHIs per leaf node. If a leaf node holds $v - 1$ SBs and v NHIs, then $W_D \geq (v-1)b + vh$ and $v \leq \frac{W_D + b}{b+h}$.

2) Lookup rate: As discussed in Section III-A, the lookup rate can be scaled up by instantiating K identical BRTree pipelines on the FPGA. The maximum lookup rate of a K-way BRTree multipipeline is limited by both the on-chip and off-chip memory access rates:

$$R = K \cdot \min\left(2f_C, \left\lfloor \frac{d}{KM_D} \right\rfloor f_D \right) \tag{4}$$

The left and right terms inside the $\min(\cdot)$ above calculate the per-level on-chip and off-chip memory access rates per pipeline, respectively. The on-chip memories are dual-ported. There are $\lfloor d/(KM_D) \rfloor$ DDR3 data buses per off-chip stage per pipeline. To scale up the (read) access rate, the same set of BRTree nodes are replicated to all the DDR3 data buses at the same stage.

3) Lookup latency: The latency of the BRTree pipeline can be estimated as follows:

$$T = M_C \cdot \frac{L}{f_C} + M_D \cdot \left(\frac{L-1}{f_C} + t_D \right) \tag{5}$$

The first term on the right side describes the latency through the M_C on-chip memory stages, where each (superpipelined) stage requires L cycles to traverse. The second term describes the latency through the M_D off-chip stages, where the 1-cycle (out of L) memory access time is replaced by the DDR3 access latency t_D.

Table II: Design goals of the example scenarios.

Case	Goal	Requirement	Optimize	b	h
#1	Table size	$R \geq 400$ Mlps	$\max(N)$	64	16
#2	Lookup rate	$N \geq 1$ Mpfx	$\max(R)$	32	16
#3	Latency	$T \leq 550$ ns	$\max(NR)$	32	16

Table III: Virtex 7 XC7VX1140T FLG1930 [19].

Logic Slices	DistRAM Kb	BRAM Kb	User I/O # pins	GTH # serdes
178,000	17,700	67,680	1,100	24

V. Prototype Design Scenarios

Using the performance models in the previous section, we can quickly explore the design tradeoffs in our BRTree multipipeline architecture for various performance requirements and resource constraints.

We demonstrate this process by optimizing an IP lookup solution for three different prototype design scenarios summarized in Table II. We target the Virtex 7 XC7VX1140T FPGA in the FLG1930 package; the resource specification of the FPGA and the specific usage constraints to the prototype scenarios are listed in Table III and Table IV, respectively. The resulting design parameters and performance values are summarized in Table V.

A. Case #1 — Maximize Table Size

1) Rationale: In Case #1, we design an IPv6 lookup engine with at least 400 Mlps, while at the same time maximizing the table size. The lookup rate, for example, would correspond to 'the maximum packet rate of two 100 Gbps links.

2) Design process: According to Eq. (4) in Section IV, for $R \geq 400$ Mlps, both $2f_C K$ and $\lfloor d/(KM_D) \rfloor f_D$ need to be $\geq 400\,\text{MHz}$. With $f_C = 200\,\text{MHz}$, $d = 20$ and $f_D = 100\,\text{MHz}$, we have $K \geq 1$ and $M_D \leq 5$.

Let $K = 1$, where all on-chip memory is used for a single BRTree pipeline. Assume $s_{dist} + s_B \simeq 36$ Mbits of on-chip memory are actually utilized, providing access to 512K 64-bit SBs.[5] This can be implemented in on-chip memory as a 19-level BRTree. The first 12 levels utilize $4095 \times 64 \simeq 256$ Kbits of distRAM; the last 7 levels occupy $508\text{K} \times 72 \simeq 35.7$ Mbits of BRAM.

To optimize for large table size, we shall also maximize N' in Eq. (3) by employing as many off-chip stages (M_D) as possible. With $M_D = 1$, up to 512K nodes can be addressed in the first off-chip stage. To achieve the required lookup rate, the same node data need to be replicated to $R/f_D = 4$ independent DDR3 data buses. Each data bus needs to provide access to at least $512\text{K} \times 8W_D = 1024$ Mbits of memory, which can be satisfied by a 1-Gbit DDR3-1600 chip.

With $M_D = 2$ and $\left\lfloor \frac{W_D}{b} + 1 \right\rfloor = 5$, up to $512\text{K} \times 5^{2-1} = 2.5\text{M}$ nodes can be addressed in the second off-chip stage. Again, the same node data need to be replicated to four independent DDR3 data buses; each data bus needs to access at least $2.5\text{M} \times 8W_D = 5$ Gbits of memory, sufficed by an 8-Gbit DDR3-1600 chip.

It is possible to have a third off-chip stage, which could potentially address up to $512\text{K} \times 5^{3-1} \simeq 12.5\text{M}$ nodes. Since the largest DDR3 chip available today is 8 Gbits, which is enough for only $8192\text{M}/(8W_D) = 4\text{M}$ nodes, multiple DDR3 chips are needed to store all the nodes addressable in this stage. We partition the 12 remaining DDR3 data buses into four sets. The three buses in each set connect to three 8-Gbit DDR3-1600 chips, respectively, providing access to up to 12M nodes. The same node data are then replicated across all four sets to achieve the required 400 Maccs/sec.

3) Results: Utilizing all 20 available DDR3 data buses in 3 off-chip stages, a table size of $N = \frac{1}{2} \times 12\text{M} \times \left\lfloor \frac{W_D+b}{b+h} \right\rfloor = 24$ Mpfx and a lookup rate of 400 Mlps can be achieved.

As shown in Figure 2, the clock rate remains at least 200 MHz when more off-chip DDR3 stages are utilized, allowing the table size to scale up significantly while sustaining a high lookup rate.

[5] Although $b = 64$, BRAM widths on Virtex 7 are naturally 18-bit multiples, making the smallest node 72 bits in BRAM. Thus $36\text{M}/72 = 512\text{K}$.

Table IV: Resource constraints to the prototype scenarios.

s_{dist}	s_B	s_{last}	d	f_C	f_D	t_D
<2 Mbits	<50 Mbits	≤ 160 Gbits	≤ 20 buses	200 MHz	100 Maps	100 ns

Note: There are over 17 Mbits distRAM and 66 Mbits BRAM on the target FPGA. However, we utilize < 2 Mbits of distRAM for power consideration, and reserve at least 16 Mbits BRAM for the serial I/O and DDR3 controllers.

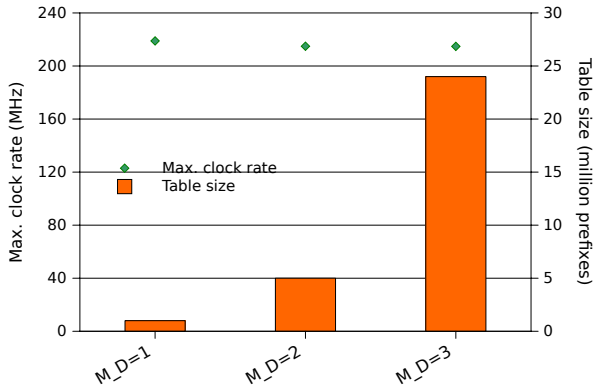

Figure 2: Scalability in prefix table size.

B. Case #2 — Maximize Lookup Rate

1) Rationale: In Case #2, we design an IPv4 lookup engine with a table size of at least 1 million prefixes, while at the same time maximizing the lookup rate. This table size is over twice as many as the number of BGP entries in the largest backbone router today.

2) Design process: According to Eq. (1), using 52 Mbits on-chip memory only, the IP lookup engine can access at most $52\,\text{Mbits}/(2 \cdot (32+16)\,\text{bits}) \simeq 554\text{K}$ prefixes. This does not meet the 1 Mpfx requirement even with only $K = 1$ pipeline. Therefore, at least one off-chip stage is needed.

Table V: Design parameters and performance results of the prototype scenarios.

Case	*IP*	K	M_C	M_D	L	W_{dist}	W_B	W_D	N	R	T
#1	v6	1	19	3	4	64	72	256	24 Mpfx	400 Mlps	725 ns
#2	v4	4	19	1	4	32	36	256	1.1 Mpfx	1.6 Blps	495 ns
#3	v4	1	20	0	4	32	36	256	554 Kpfx	400 Mlps	400 ns

Applying $M_D = 1$ to Eq. (2), with $N' = 1$, $v = 6$,

$$N \geq \frac{1}{2K} \min\left(\frac{52\,\text{Mbits}}{36}, \frac{160\,\text{Gbits}}{8 \times 256 \times 4}\right) \cdot 6 \geq 1\,\text{M}$$

we get $K < 4.33$, or up to 4 pipelines can be constructed. Then, applying $K = 4$ to Eq. (4), we find

$$\begin{aligned} R &= 4 \times \min\left(2 \cdot 200\,\text{MHz}, \left\lfloor \frac{20}{4} \right\rfloor \cdot 100\,\text{MHz}\right) \\ &= 1600\,\text{Mlps} \end{aligned}$$

or a maximum lookup rate of 1.6 Blps can be achieved. Note that the lookup rate is limited by the on-chip memory access rate, as the left term inside the $\min(\cdot)$ above.

Specifically, based on our performance models, the lookup rate-optimized solution can be constructed as a 4-way multipipeline. Each pipeline utilizes $52\,\text{Mbits}/4 = 13\,\text{Mbits}$ of on-chip memory for accessing $13\text{M}/36 \simeq 369\text{K}$ SBs.[6] This can be implemented as a 19-level BRTree whose last level is incomplete with only $369\text{K} - 256\text{K} = 113\text{K}$ nodes (instead of 256K nodes). In each pipeline, the first 12 levels of the BRTree utilize $4095 \times 32 \simeq 128$ Kbits of distRAM; the last 7 levels utilize $365\text{K} \times 36 \simeq 12.8$ Mbits of BRAM. Each pipeline also employs one off-chip stage that connects to $2f_C/f_D = 4$ DDR3 data buses. Each data bus provides access to $369\text{K} \times 8W_D = 738\,\text{Mbits}$ of node data in a 1-Gbit DDR3-1600 chip.

3) Results: In total, 16 off-chip DDR3 data buses are utilized in one off-chip stage, achieving an aggregated lookup rate of 1.6 Blps, and a table size of $\frac{1}{2} \times 369\text{K} \times 6 \simeq 1.1\text{M}$ IPv4 prefixes.

C. Case #3 — Guaranteed Lookup Latency

1) Rationale: One major concern in IP lookup is latency. Specifically, while a packet header is going through the IP lookup pipeline, the entire packet must be buffered in some high-speed memory inside the main packet switch. Thus the maximum lookup latency is usually a hard limit defined by the offloading switch based on its buffer size limit.

In Case #3, we design an IP lookup pipeline with maximum $N \times R$ and a guaranteed lookup latency of $T \leq 400$ ns.

2) Design process: As shown in Eq. 5, with $L = 4$, $f_C = 200$ MHz and $t_D = 100$ ns, each on-chip stage would take 20 ns and off-chip stage 115 ns. There is only a finite way to divide the 400 ns limit into on-chip and off-chip parts:

$$T = \begin{cases} 400\,\text{ns} & M_C = 20, M_D = 0 \\ 395\,\text{ns} & M_C = 14, M_D = 1 \\ 390\,\text{ns} & M_C = 8\,,\ M_D = 2 \\ \quad\vdots & \end{cases}$$

When setting $M_C = 20$ and $M_D = 0$, only the on-chip memory are utilized. Similar to the initial discussion in Section V-B, by using 52 Mbits on-chip memory, the BRTree pipeline can have a table size of 554K prefixes and achieve a lookup rate of $2f_C = 400$ Mlps.

When setting $M_C = 14$ and $M_D = 1$, one off-chip stage is employed with up to 20 available DDR3 data buses. The lookup rate R can be defined in Eq. 4 as:

$$R = K \cdot \min\left(2 \times 200\,\text{MHz}, \left\lfloor \frac{20}{K} \right\rfloor \times 100\,\text{MHz}\right)$$

where $K = 5$ results in a maximum $R = 2000$ Mlps. The 5 pipelines implement 5x 14-level BRTrees in the on-chip memory, providing access to total $5 \times \left(2^{14} - 1\right) \simeq 80\text{K}$ nodes. Replacing $(s_{dist} + s_b)/b$ with 80K and setting $K = 5$, $N' = 1$ and $v = 6$ in Eq. 2, we have $N \geq 48\text{K}$ prefixes.

Comparing the above two configurations, we find that while adding an off-chip stage under the fixed 400 ns lookup latency increases the lookup rate to 5x, it also reduces the table size to 1/10 of the on-chip only configuration. Therefore, without enumerating all possible configurations, we conclude that the optimal design for Case #3 is to construct the IP lookup pipeline with on-chip memory access only.

3) Results: With $M_C = 20$ and $M_D = 0$, a 20-level BRTree pipeline with at least 554K prefixes (1108K nodes) and achieving 400 Mlps can be constructed entirely in the on-chip memory. The lookup pipeline can be traversed in 400 ns.

VI. Performance Evaluation

A. Implementation Detail

All the design experiments were conducted using Xilinx ISE Design Suite 14.1, targeting Virtex 7 XC7VX1140T FPGA in the FLG1930 package [19]. The BRTree pipeline architecture is first modularized into individual design blocks; for each design block, we specified the timing constraints and measured the power consumption. Finally, all design blocks were merged together as a complete design.

For the off-chip memory access, every two DDR3 data buses share one command bus, multiplexed by the chip-select signals. Up to 20 DDR3-1600 chips are connected via 20 data buses to the FPGA. As discussed in Section III-C, the same data is replicated in all 8 banks inside each DDR3 chip; different banks are accessed in a round-robin fashion. A

[6] Again, we allocate 36 (instead of 32) bits per node because the BRAM width is naturally a multiple of 18 bits.

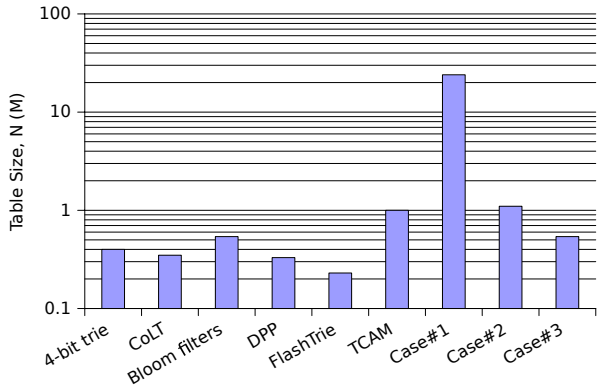

Figure 3: Supported IPv4/IPv6 table sizes.

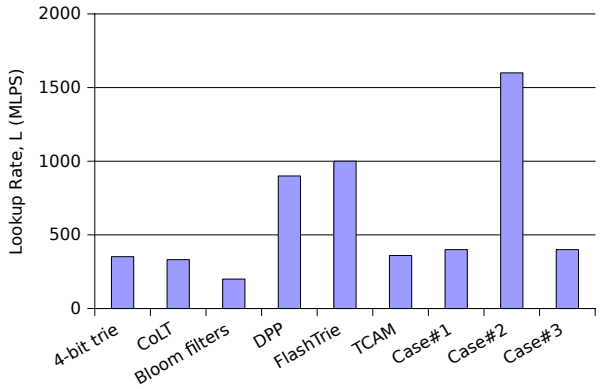

Figure 4: Maximum IPv4/IPv6 lookup rates.

simple in-order pipelined scheduler is implemented to buffer the read/write requests sent to the DDR3 chips before the corresponding responses are returned. Since the read latency is ~100 ns, a FIFO of depth 20 (at 200 MHz) is required within the DDR3 scheduler. Each read access returns up to $M_D = 256$ bits of data.

As discussed in Section III-A, each BRTree pipeline employs 4x GTH transceivers to support up to 426 million lookup commands (and responses) per second. We did not implement these industry-standard PHY and MAC circuit blocks.[7] [8]

Post place-and-route results are presented here for timing reports. The "power" optimization option was used in both synthesis and place-and-route settings; we used Xilinx ISE Design Suite 14.1 XPower Analyzer to measure the worst-case power usage after the place-and-route process. The BRTree pipeline architecture does not make any assumption on the prefix distribution, so no actual prefix table is necessary for a fair evaluation of its performance.

B. Comparison and analysis

The design parameters and performance results are summarized in Table V. The lookup rate and table size of our three prototype designs are compared respectively with 4-bit trie [10], CoLT [17], Bloom filters [20], DPP [7], FlashTrie [15] and TCAM [8]. We scaled the results of all prior work based on the current technology. The power performance of our three designs are compared only with TCAM; other work did not provide enough information for us to evaluate their power usage fairly.

No comparison to prior work is made with respect to lookup latency. To our knowledge, our work is the first one to provide a design methodology which guarantees a maximum latency for IP lookup on FPGA.

1) Table size: Figure 3 compares the table size of various IP lookup engines. The software-based 4-bit trie assumes a prefix table whose trie would fit completely inside the 4MB L2 cache of a state-of-the-art multi-core processor. The CoLT, Bloom filter and DPP-based results all assume using up to 60 Mbits BRAM on the Virtex 7 FPGA. The FlashTrie results assumes only ~8.2 Mbits BRAM for a prefix table size of < 320K IPv6 prefixes. Unlike the BRTree pipeline architecture, FlashTrie's table size cannot be scaled up simply with more BRAM resource; its BRAM usage depends on how a set of hash functions perform on the particular set of prefixes.

2) Lookup rate: The lookup rate of software-based 4-bit trie is scaled up linearly to a 2.4 GHz 8-core CPU. The lookup rate of CoLT depends highly on the speed of the on-chip memory (BRAM), which is revised to 500 MHz. The lookup rates of both the Bloom filter and DPP results are estimated with a 200 MHz clock rate, similar to our own designs. The lookup rate of FlashTrie, in particular, is scaled up by assuming 20 off-chip DDR3 data bus each supporting 100 Maccs/sec, the same assumption used by our BRTree pipeline.[9] Since each FlashTrie lookup engine requires two off-chip memory accesses, a maximum lookup rate of 1 Blps can be achieved, corresponding to 5 parallel FlashTrie engines running at 200 MHz. For IPv6, the 5 FlashTrie engines would utilize approximately $8.2\,\text{Mbits} \times \lfloor 5/2 \rfloor = 24.6\,\text{Mbits}$ of BRAM. The lookup rate is limited by the off-chip memory access rate.

3) Power usage and energy efficiency: Figure 5 compares the power requirement and energy efficiency of the three prototype designs with a TCAM solution. The power usage of all designs are plotted against the left y-axis. The energy efficiency is computed as $N \times R/P$ (table_size $\times$ lookup_rate / power_usage), in unit of Maccs/Joule, and plotted against the right y-axis of the graph (in log scale). The TCAM power requirement is estimated using 4x state-of-the-art 20 Mbits TCAM [21], each consuming 15W while running at 360 MHz. We ignore the power consumed by the logic for managing the TCAM access.

As shown in the figure, all of our prototype designs use less power and achieve much higher (from 5.6x to over 70x) energy efficiency than the TCAM-based solution. The difference is more significant with a larger table size, where the BRTree pipeline architecture scales up more efficiently than TCAM.

[7] http://www.xilinx.com/ipcenter/interlaken/interlaken_order.htm
[8] http://www.altera.com/products/ip/iup/high-speed/m-alt-interlaken.html

[9] While this is lower than the 200 Maccs/sec assumed in [15], based on the modeling in Section III-C, we believe this is the more realistic constraint.

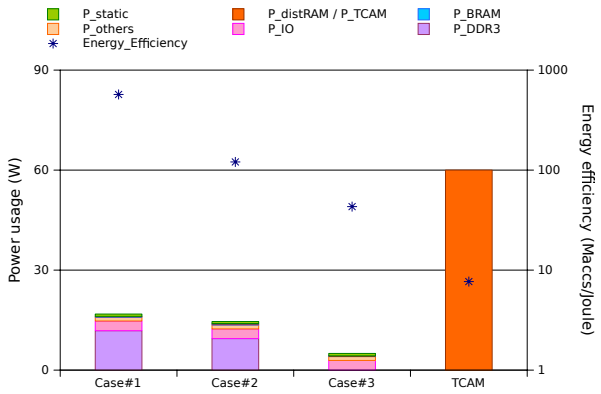

Figure 5: Power comparison between our prototype designs and TCAM.

VII. Conclusion and Future work

In this paper we presented a unified design methodology for a scalable balanced range tree (BRTree) based IP lookup architecture on FPGA. Concise and accurate resource and performance models were developed to characterize the design tradeoffs of a scalable IPv4 or IPv6 lookup engine on FPGA. Together with a simple but realistic off-chip DDR3 access model, a broad range of design parameters and system performance could be quickly explored at design time. Three prototype design scenarios were studied in detail to evaluate their lookup rate, table size, latency and power performance.

For future work, the proposed performance models and optimization methodology can be applied to other applications with similar architectures. It can also be extended to handle virtual lookup tables and to facilitate efficient dynamic prefix updates.

References

[1] "Growth of the BGP Table – 1994 to Present," http://bgp.potaroo.net/, Jul. 2012.

[2] B. Lampson, V. Srinivasan, and G. Varghese, "IP Lookups using Multiway and Multicolumn Search," in *INFOCOM'98*, May 1998.

[3] P. Warkhede, S. Suri, and G. Varghese, "Multiway Range Trees: Scalable IP Lookup with Fast Updates," *Computer Networks*, vol. 44, no. 3, pp. 289 – 303, 2004.

[4] H. Lu and S. Sahni, "A B-Tree Dynamic Router-Table Design," *IEEE Trans. Comput.*, vol. 54, no. 7, pp. 813–824, 2005.

[5] I. Sourdis and S. Katamaneni, "Longest Prefix Match and Updates in Range Tries," in *2011 IEEE Intl. Conf. on Application-Specific Systems, Architectures and Processors (ASAP'11)*, sept. 2011, pp. 51 –58.

[6] P. Zhong, "An IPv6 Address Lookup Algorithm based on Recursive Balanced Multi-way Range Trees with Efficient Search and Update," in *Proc. Int Computer Science and Service System (CSSS) Conf*, 2011, pp. 2059–2063.

[7] H. Le and V. Prasanna, "Scalable Tree-based Architectures for IPv4/v6 Lookup using Prefix Partitioning," *Computers, IEEE Transactions on*, vol. 61, no. 7, pp. 1026 –1039, july 2012.

[8] F. Zane, G. Narlikar, and A. Basu, "CoolCAMs: Power-efficient TCAMs for Forwarding Engines," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1, march-3 april 2003, pp. 42 – 52 vol.1.

[9] K. Zheng, C. Hu, H. Lu, and B. Liu, "A TCAM-based Distributed Parallel IP Lookup Scheme and Performance Analysis," *IEEE/ACM Trans. Netw.*, vol. 14, no. 4, pp. 863–875, Aug. 2006. [Online]. Available: http://dx.doi.org/10.1109/TNET.2006.880171

[10] V. Srinivasan and G. Varghese, "Fast Address Lookups using Controlled Prefix Expansion," *ACM Trans. Comput. Syst.*, vol. 17, no. 1, pp. 1–40, Feb. 1999. [Online]. Available: http://doi.acm.org/10.1145/296502.296503

[11] M. Hanna, S. Cho, and R. Melhem, "A Novel Scalable IPv6 Lookup Scheme using Compressed Pipelined Tries," in *Proceedings of the 10th international IFIP TC 6 conference on Networking - Volume Part I*, ser. NETWORKING'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 406–419. [Online]. Available: http://dl.acm.org/citation.cfm?id=2008780.2008820

[12] K. Huang, G. Xie, Y. Li, and A. Liu, "Offset Addressing Approach to Memory-efficient IP Address Lookup," in *INFOCOM, 2011 Proceedings IEEE*, april 2011, pp. 306 –310.

[13] H. Fadishei, M. Zamani, and M. Sabaei, "A Novel Reconfigurable Hardware Architecture for IP Address Lookup," in *Architecture for networking and communications systems, 2005. ANCS 2005. Symposium on*, oct. 2005, pp. 81 –90.

[14] H. Song, M. Kodialam, F. Hao, and T. Lakshman, "Scalable IP Lookups using Shape Graphs," in *Network Protocols, 2009. ICNP 2009. 17th IEEE International Conference on*, oct. 2009, pp. 73 –82.

[15] M. Bando and H. J. Chao, "Flashtrie: Hash-based Prefix-compressed Trie for IP Route Lookup beyond 100Gbps," in *Proceedings of the 29th conference on Information communications*, ser. INFOCOM'10. Piscataway, NJ, USA: IEEE Press, 2010, pp. 821–829. [Online]. Available: http://dl.acm.org/citation.cfm?id=1833515.1833653

[16] R. Sangireddy, N. Futamura, S. Aluru, and A. K. Somani, "Scalable, Memory efficient, High-speed IP Lookup Algorithms," *IEEE/ACM Trans. Netw.*, vol. 13, no. 4, pp. 802–812, Aug. 2005. [Online]. Available: http://dx.doi.org/10.1109/TNET.2005.852878

[17] F. Pong and N.-F. Tzeng, "Concise Lookup Tables for IPv4 and IPv6 Longest Prefix Matching in Scalable Routers," *Networking, IEEE/ACM Transactions on*, vol. 20, no. 3, pp. 729 –741, june 2012.

[18] "DDR3 SDRAM MT41J128M16 - 16 Meg x 16 x 8 Banks," http://www.micron.com/products/dram/ddr3-sdram, Micron Technology, Inc., 2006.

[19] "Virtex-7 FPGA Family," http://www.xilinx.com/products/virtex7.

[20] H. Song, F. Hao, M. Kodialam, and T. Lakshman, "IPv6 Lookups using Distributed and Load Balanced Bloom Filters for 100Gbps Core Router Line Cards," in *INFOCOM 2009, IEEE*, april 2009, pp. 2518 –2526.

[21] "20Mbit QUAD-Search Content Addressable Memory," http://am.renesas.com/products/memory/TCAM.