International Journal of Innovative Technologies
WWW.IJITECH.ORG
ISSN 2321-8665
Volume.06, Issue.01,
January-June, 2018,
Pages:0215-0219

A Novel High Speed Memory Solution using TCAM

P. POOJITHA[1], ASHOK KUMAR[2]
[1]PG Scholar, Dept of ECE, PBR Visvodaya Institute of Technology and Science, Kavali, Nellore(Dt), AP, India.
[2]Associate Professor, Dept of ECE, PBR Visvodaya Institute of Technology and Science, Kavali, Nellore(Dt), AP, India.

Abstract: Ternary content addressable memories (TCAMs) perform fast inquiry errand yet when differentiated and static random access memories (SRAMs), TCAMs have certain obstructions, for instance, low amassing thickness, for the most part direct access time, low adaptability, complex hardware, and are uncommonly exorbitant. In like manner, would we have the capacity to use the upsides of SRAM by planning it (with additional justification) to engage it to act like TCAM? This short proposes a novel memory architecture, named Z-TCAM, which impersonates the TCAM handiness with SRAM. Z-TCAM wisely portions the conventional TCAM table along areas and columns into cream TCAM sub tables, which are then dealt with to depict their relating memory pieces. Two case gets ready for Z-TCAM of sizes 512 × 36 and 64 × 32 have been executed on Xilinx Virtex-7 field-programmable gate array.

Keywords: Application-Particular Incorporated Circuit (ASIC), Memory Architecture, Priority Encoder, Static Random Access Memory(SRAM)- Based TCAM, Ternary Content Addressable Memory (TCAM).

I. INTRODUCTION

Ternary content addressable memory (TCAM) empowers its memory to be looked by contents rather than by an address and a memory territory among matches is sent to the yield in a reliable time. A customary TCAM cell has two static random access memory (SRAM) cells and a relationship equipment and can store three states − 0, 1, and x where x is a couldn't mind less state. The x state is always seen as facilitated paying little mind to the data bit. The unfaltering time quest for TCAM makes it a fitting rival in different applications, for instance, mastermind switches, data pressure, progressing illustration planning in contamination recognizable proof, and picture preparing [1]. TCAM gives single clock inquiry; in any case, it has a couple of damages differentiated and SRAM. TCAM isn't subjected to the genuine business contention found in the RAM grandstand [2]. TCAM is less thick than SRAM. The comparator's equipment in TCAM cell adds multifaceted nature to the TCAM architecture. The extra reason and capacitive stacking due to the tremendous parallelism extend the access time of TCAM, which is 3.3 times longer than the SRAM access time [3]. Common basic obstacles in like manner confine the total chip breaking point of TCAM. Complex joining of memory and method of reasoning moreover sets aside a couple of minutes consuming [1]. With the potential great conditions of SRAM over CAM, and common sense of FPGA development, we propose a memory architecture called Z-TCAM that mirrors TCAM value with SRAM and has been adequately completed on Xilinx Virtex-7 FPGA and moreover arranged using OSUcells library for 0.18 μm advancement. We ensure that the proposed TCAM offers comparative request execution, flexibility, and lower cost than set up TCAM contraptions, gave that SRAM devices are denser, more affordable, and work snappier than TCAM devices.

A. Related Work

We gather RAM-based responses for CAM in this fragment. The methods proposed in [2] and [12] use hashing to collect CAM from RAM however these methodologies encounter the evil impacts of crashes and holder surge. If various records have been placed in a surge zone, by then a question may not finish until the point that the moment that various bowls are looked for. In [12], when secured keys contain couldn't mindless bits in the bit positions used for hashing, by then such keys must be replicated in different buckets, which require extended utmost. On the other hand, if the interest scratch contains couldn't mindless bits which are taken by the hash work, different bowls must be accessed that results in execution defilement. In [2], the execution of the method ends up being easily degradable as the amount of set away segments increases. Additionally, it duplicates matched CAM, not TCAM. Thusly, hashing can't give deterministic execution inferable from potential crashes and is inefficient in dealing with extraordinary case. Standard algorithmic chase courses of action take different clock cycles [11] and besides result in inefficient memory use [10]. Strikingly, Z-TCAM has a deterministic request execution that is free of data, viably handles the trump cards, and has better memory utilize.

The technique proposed in [13] merges RAM and CAM to develop the CAM helpfulness. This approach makes packages of the customary TCAM table using some perceiving bits in CAM sections. In any case, making allocations of totally random data is a greatly dull and monotonous occupation. Since the methodology uses TCAM as a bit of the general architecture, it brings the inherent TCAM downsides in the general architecture of [13] however Z-TCAM is non particular and has a straightforward dividing. Crush based CAMs showed in [6] and [14] have an exponential augmentation in memory measure with the development in number of bits in CAM word, thusly making them prohibitive. For instance, if a CAM word has 36 bits, its size would be 236 = 64 GB in [6]. Also, the procedure in [14] just tackles rose data however in consistent CAM applications data are completely random. By arranging the data in rising solicitation, the main demand of sections is aggravated. Thusly, there must be a way to deal with store the principal areas, which is required by [14]. If one of a kind areas are seen as, the memory and power necessities also increase. Instead of [6] and [14], Z-TCAM reinforces a subjectively far reaching piece configuration, thinks about the limit of novel areas, while using fitting allocating.

TABLE I: Traditional TCAM Table And Its Hybrid Partitions (HP)

Address	Ternary Data		Layer
0	00	11	
1	01 HP_{11}	01 HP_{12}	1
2	$0x$	11	
3	11 HP_{21}	$1x$ HP_{22}	2

TABLE II: Z-TCAM Example: Data Mapping

Address	VM_{21}	VM_{22}	$OATAM_{21}$	$OATAM_{22}$	Original Address OAT_{21} 2	OAT_{21} 3	OAT_{22} 2	OAT_{22} 3
0	1	0	0	–	1	0	0	1
1	1	0	1	–	1	0	1	1
2	0	1	–	0	0	1	0	0
3	1	1	2	1	0	0	0	0

B. Paper Organization

Whatever is left of this brief is sorted out as takes after: Section II explains cross breed apportioning. Area III talks about the architecture of Z-TCAM. Segment IV clarifies Z-TCAM activities with cases. Segment V gives execution of Z-TCAM and Section VI finishes up the brief alongside featuring the future work.

II. HYBRID PARTITIONING OF TCAM TABLE

Hybrid partitioning (HP) is a total name given to vertical partitioning and level partitioning of the standard TCAM table. An instance of HP is given in Table I. HP fragments consistent TCAM table vertically (segment canny) and uniformly (push keen) into TCAM subtables, which are then arranged to be secured in their relating memory units. This handling (data mapping) has been illuminated in Section IV-A with an outline (Table II) to demonstrate the layer architecture of Z-TCAM. Vertical partitioning (VP) recommends that a TCAM articulation of C bits is separated into N subwords; each subword is of w bits. VP is used as a piece of Z-TCAM to decrease memory measure however much as could sensibly be normal. Level partitioning (HrP) isolates each vertical package using the primary address extent of customary TCAM table into L level portions. HrP can't be used alone as it is zone, power, and cost hungry yet is used to make layers. HP achieves a whole of L × N hybrid allocations. The estimations of each hybrid bundle are K × w where K is a subset from one of a kind areas and w is the amount of bits in a subword. Hybrid portions spreading over comparable areas are in a comparable layer. For example, HP21 and HP22 cross a comparable address go and are in layer 2.

III. ARCHITECTURE OF Z-TCAM

A. Overall Architecture

The general architecture of Z-TCAM is delineated in Fig. 1 where each layer speaks to the architecture appeared in Fig. 2. It has L layers and a CAM priority encoder (CPE). Each layer yields a potential match address (PMA). The PMAs are sustained to CPE, which chooses coordinate address (MA) among PMAs.

B. Layer Architecture

Layer architecture is appeared in Fig2. It contains N approval memories (VMs), 1-bit AND activity, N unique address table address memories (OATAMs), N unique address tables (OATs), K-bit AND task, and a layer priority encoder (LPE).

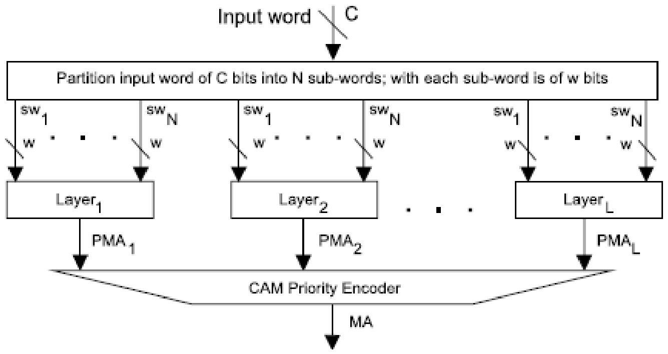

Input word
C
Partition input word of C bits into N sub-words; with each sub-word is of w bits
sw1
swN
w
Layer1
Layer2
LayerL
PMA1
PMA2
PMAL
CAM Priority Encoder
MA

Fig. 1. Architecture of Z-TCAM. (sw: subword, C: # of bits in the input word, PMA: potential match address, and MA: match address).

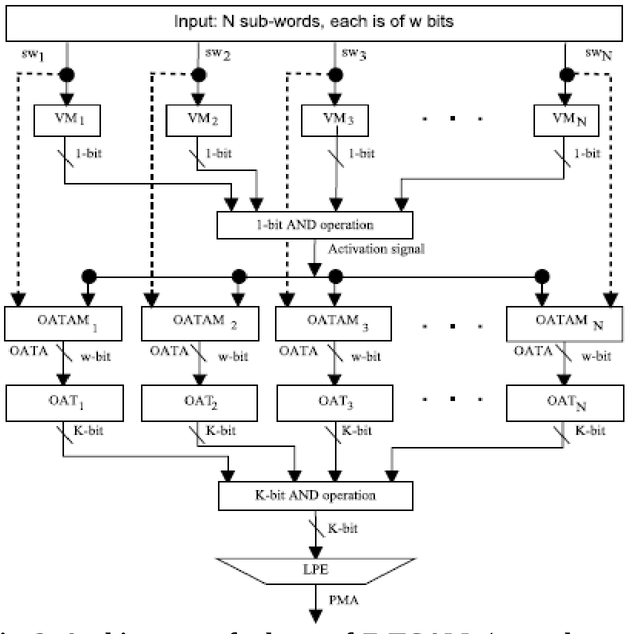


Fig. 2. Architecture of a layer of Z-TCAM. (sw: subword, VM: validation memory, OATAM: original address table address memory, OAT: original address table, and LPE: layer priority encoder).

Validation Memory: Size of each VM is 2w × 1 bits where w speaks to the quantity of bits in each subword and 2w demonstrates the quantity of lines. A subword of w bits suggests that it has add up to blends of 2w where every mix speaks to a subword. For instance, if w is of 4 bits, at that point it implies that there are aggregate of 24 = 16 blends. This clarification is likewise identified with OATAM and OAT. Each subword goes about as a deliver to VM. In the event that the memory area be summoned by a subword is high, it implies that the info subword is available, generally missing. Hence, VM approves the info subword, on the off chance that it is available. For instance, Table II demonstrates that subwords 00, 01, and 11 are mapped in VM21. This expresses memory areas 00, 01, and 11 ought to be high in VM21 and the rest of the memory areas are set to low in light of the fact that their comparing subwords don't exist.

1-bit AND Operation: It ANDs the output of all VMs. The output of 1-bit AND activity chooses the continuation of a pursuit task. In the event that the consequence of 1-bit AND activity is high, at that point it allows the continuation of a hunt task, generally crisscross happens in the comparing layer.

Original Address Table Address Memory: Each OATAM is of 2w × w bits where 2w is the amount of sections and each line has w bits. In OATAM, an address is secured at the memory region recorded by a subword and that convey is then used to summon a section from its relating OAT. In case a subword in VM is mapped, by then a relating address is furthermore secured in OATAM at a memory zone accessed by the subword. For example, Table II shows OATAM21 where addresses are secured at the memory zones 00, 01, and 11. The yield of OATAM is called as OATA. Hyphen "- " shows that the relating memory territory has no data in light of the fact that the looking at subword for the memory region is truant in VM.

Original Address Table: Dimensions of OAT are 2w× K where w is the number of bits in a subword, 2w represents number of rows, and K is the number of bits in each row where each bit represents an original address. Here K is a subset of original addresses from conventional TCAM table. It is OAT, which considers the storage of original addresses. An example of OAT is given in Table II, where 1 shows the presence of a subword at an original address.

K-bit AND Operation: It ANDs bit-by-bit the read out K-bit rows from all OATs and forwards the result to LPE.

Layer Priority Encoder: Because we emulate TCAM and multiple matches may occur in TCAM [15], the LPE selects PMA among the outputs of K-bit AND operation.

IV. Z-TCAM OPERATIONS

A. Data Mapping Operation

Set up TCAM table is honestly allocated hybrid portions. Each hybrid fragment is then wandered into a twofold frame. Thusly, we at first develop x into states 0 and 1 to be secured in SRAM. For example, in case we have a TCAM articulation of 010x, by then it is wandered into 0100 and 0101. Each subword, going about as an address, is associated with its relating VM and a justification "1" is formed at that memory territory. The same subword is moreover associated with its specific OATAM and w bits data are made at that memory region. In the midst of look for, these w bits data go about as a convey to the OAT. The K bits data are also formed at the memory region in OAT controlled by its relating OATA. In this way, thusly, each and every hybrid section are mapped. A subword in a hybrid portion can be accessible at different zones. Along these lines, it is mapped in its looking at VM and its one of a kind address(es) is/are mapped to its/their relating bit(s) in its specific OAT. Since a lone piece in OAT addresses an exceptional address, only those memory regions in VMs and address positions/novel areas in OATs are high, which are mapped while remaining memory territories and convey positions are set to low in VMs and OATs, independently. Instance of data mapping is showed up in Table II. We use Table I to be mapped to Z-TCAM. We take N = 2, L = 2, K = 2, and w = 2. After critical handling, HP11, HP12, HP21, and HP22 are mapped to their looking at memory units. In the representation, we outline bundles of layer 2 to their looking at memory units. Hybrid sections of layer 1 can be viably mapped in similar way.

B. Search Operation

Searching in a Layer of Z-TCAM: Calculation 1 portrays seeking in a layer of Z-TCAM. N subwords are simultaneously connected to a layer. The subwords at that point read out their relating memory areas from their separate VMs. In the event that all VMs approve their relating subwords (proportional to 1-bit AND activity in Fig. 2), at that point seeking will proceed, generally crisscross happens in the layer. Endless supply of all subwords, the subwords read out their individual memory regions from their looking at OATAMs at the same time and yield their relating OATAs. All OATAs by then read out K-bit segments from their looking at OATs in the meantime, which are then bitwise ANDed. LPE picks PMA from the result of the K-bit AND undertaking. Instance of an interest undertaking in layer 2 is showed up in Table III, after Algorithm 1. Memory impedes in Table II ought to be looked.

Algorithm 1 Pseudocode for Searching in a Layer of Z-TCAM

```
INPUT: N sub-words
OUTPUT: PMA
1:  → Apply N sub-words
2:  → Apply all sub-words simultaneously to their VMs
3:  → Read all VMs concurrently
4:  if all VMs validate their corresponding sub-words then
5:      → Sustain search operation
6:      → a. Read all OATAMs in parallel
7:      → b. Read all OATs simultaneously
8:      → c. AND bit-wise all K-bits rows
9:      → d. Select PMA/mismatch occurs
10: else
11:     → Mismatch occurs
12: end if
```

TABLE III: Example Of A Search Operation In Layer 2 of Z-Tcam

Steps	Activity

1	Sub-word$_1$ = 00
	Sub-word$_2$ = 11

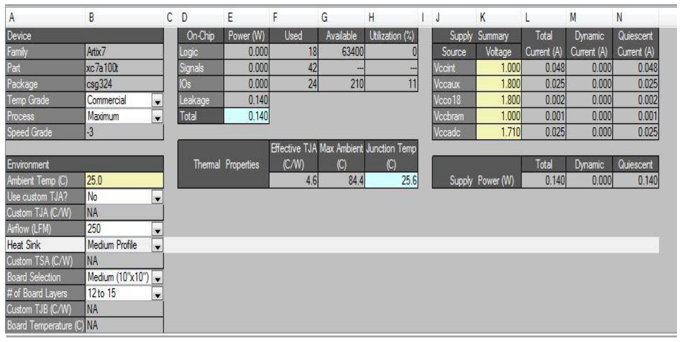
2	Sub-word$_1$ is applied to VM$_{21}$ Sub-word$_2$ is applied to VM$_{22}$
3	Read out bit from VM$_{21}$ = 1 Read out bit from VM$_{22}$ = 1
4	Have all VMs validated their corresponding sub-words?
5	Yes, so sustain search operation
6	Read out data from OATAM$_{21}$ = 0 Read out data from OATAM$_{22}$ = 1
7	Read out data from OAT$_{21}$ = 10 Read out data from OAT$_{22}$ = 11
8	K-bit AND operation result = 10
9	PMA = 2

Algorithm 2 Pseudocode for Searching in Z-TCAM

```
INPUT: Search Key
OUTPUT: MA
1: → Apply search key
2: → Divide search key into N sub-words
3: → All layers use algorithm 1 in parallel
4: → Select MA among PMAs/mismatch occurs
```

Searching in Z-TCAM: Chase errand in the proposed TCAM happens at the same time in all layers, which takes after Algorithm 2. Chase key is associated with Z-TCAM, which is then divided into N subwords. Consequent to looking for, PMAs are accessible from all layers. CPE picks MA among PMAs; for the most part a bewilder of the data word happens. Table IV gives general request movement in Z-TCAM, which takes after Algorithm 2. We use input word 0011 to be looked. In Table IV, for elucidation, we acknowledge that we have furthermore mapped layer 1.

V. Z-TCAM IMPLEMENTATION AND RESULTS



Device	
Family	Artix7
Part	xc7a100t
Package	csg324
Temp Grade	Commercial
Process	Maximum
Speed Grade	-3

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	18	63400	0
Signals	0.000	42	---	---
IOs	0.000	24	210	11
Leakage	0.140			
Total	0.140			

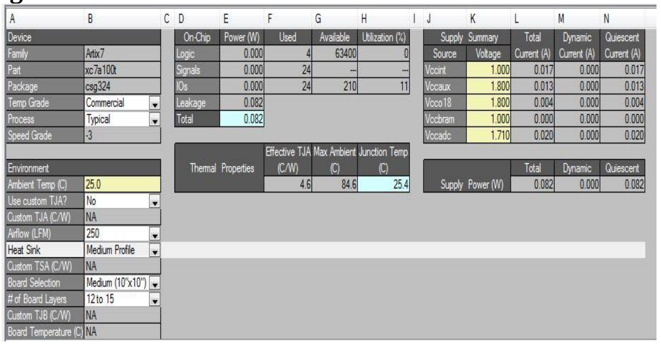
Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.048	0.000	0.048
Vccaux	1.800	0.025	0.000	0.025
Vcco18	1.800	0.002	0.000	0.002
Vccbram	1.000	0.001	0.000	0.001
Vccadc	1.710	0.025	0.000	0.025

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.6	84.4	25.6

	Total	Dynamic	Quiescent
Supply Power (W)	0.140	0.000	0.140

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

Fig.3.



Device	
Family	Artix7
Part	xc7a100t
Package	csg324
Temp Grade	Commercial
Process	Typical
Speed Grade	-3

On-Chip	Power (W)	Used	Available	Utilization (%)
Logic	0.000	4	63400	0
Signals	0.000	24	---	---
IOs	0.000	24	210	11
Leakage	0.082			
Total	0.082			

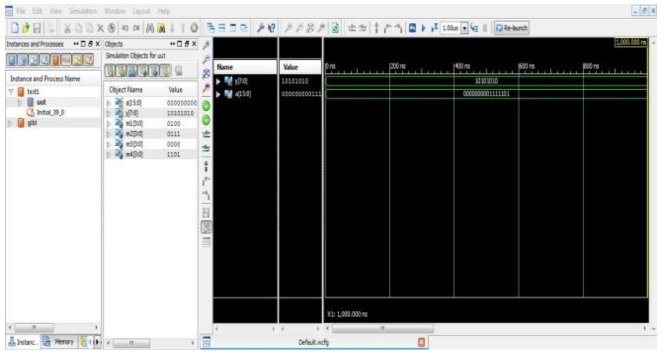
Supply Source	Summary Voltage	Total Current (A)	Dynamic Current (A)	Quiescent Current (A)
Vccint	1.000	0.017	0.000	0.017
Vccaux	1.800	0.013	0.000	0.013
Vcco18	1.800	0.004	0.000	0.004
Vccbram	1.000	0.000	0.000	0.000
Vccadc	1.710	0.020	0.000	0.020

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	4.6	84.6	25.4

	Total	Dynamic	Quiescent
Supply Power (W)	0.082	0.000	0.082

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	250
Heat Sink	Medium Profile
Custom TSA (C/W)	NA
Board Selection	Medium (10"x10")
# of Board Layers	12 to 15
Custom TJB (C/W)	NA
Board Temperature (C)	NA

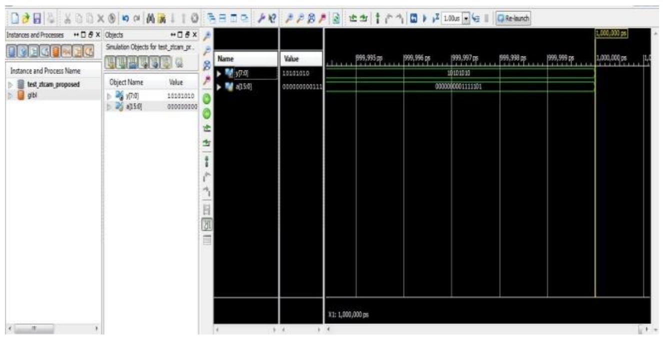
Fig.4.



Object Name	Value
a[15:0]	000000000
y[7:0]	10101010
m1[3:0]	0100
m2[3:0]	0111
m3[3:0]	0000
m4[3:0]	1101

Name	Value
y[7:0]	10101010
a[15:0]	000000000111

Fig.5.



Object Name	Value
y[7:0]	10101010
a[15:0]	000000000

Name	Value
y[7:0]	10101010
a[15:0]	000000000111

Fig.6.

VI. CONCLUSION

In this short, we have presented a novel SRAM-based TCAM architecture of Z-TCAM. We have executed two case designs of 512 × 36 and 64 × 32 of Z-TCAM on Xilinx Virtex-7 FPGA. We have similarly laid out 64 × 32 Z-TCAM in OSUcells library for 0.18 μm innovation, which insists its specific feasibility. FPGA use is a noteworthy notwith-standing for Z-TCAM. Resources utilize, speed, and power use for different conditions for the case anticipates FPGA and what's more in ASIC have been masterminded. Z-TCAM in like manner ensures enormous point of confinement TCAM however this capacity is required by customary ones. Additionally, the proposed TCAM has a less perplexing structure, and fundamentally, has a deterministic chase execution of single word relationship per clock cycle.

VII. REFERENCES

[1] N. Mohan, W. Fung, D. Wright, and M. Sachdev, "Design techniques and test methodology for low-power TCAMs," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 6, pp. 573–586, Jun. 2006.
[2] P. Mahoney, Y. Savaria, G. Bois, and P. Plante, "Parallel hashing memories: An alternative to content addressable memories," in Proc. 3rd Int. IEEE-NEWCAS Conf., Jun. 2005, pp. 223–226.
[3] S. Dharmapurikar, P. Krishnamurthy, and D. Taylor, "Longest prefix matching using bloom filters," IEEE/ACM Trans. Netw., vol. 14, no. 2, pp. 397–409, Apr. 2006.
[4] D. E. Taylor, "Survey and taxonomy of packet classification techniques," ACM Comput. Surveys, New York, NY, USA: Tech. Rep. WUCSE-2004-24, 2004.
[5] P. Mahoney, Y. Savaria, G. Bois, and P. Plante, "Transactions on highperformance embedded architectures

and compilers II," in Performance Characterization for the Implementation of Content Addressable Memories Based on Parallel Hashing Memories, P. Stenström, Ed. Berlin, Germany: Springer-Verlag, 2009, pp. 307–325.
[6] S. V. Kartalopoulos, "RAM-based associative content-addressable memory device, method of operation thereof and ATM communication switching system employing the same," U.S. Patent 6 097 724, Aug. 1, 2000.
[7] W. Jiang and V. Prasanna, "Scalable packet classification on FPGA," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 20, no. 9, pp. 1668–1680, Sep. 2012.
[8] M. Becchi and P. Crowley, "Efficient regular expression evaluation: Theory to practice," in Proc. 4th ACM/IEEE Symp. Archit. Netw. Commun. Syst., Nov. 2008, pp. 50–59.
[9] Xilinx, San Jose, CA, USA. Xilinx FPGAs [Online]. Available: http://www.xilinx.com
[10] W. Jiang and V. K. Prasanna, "Large-scale wire-speed packet classification on FPGAs," in Proc. ACM/SIGDA Int. Symp. Field Program. Gate Arrays, 2009, pp. 219–228.
[11] W. Jiang and V. Prasanna, "Parallel IP lookup using multiple SRAMbased pipelines," in Proc. IEEE Int. Symp. Parallel Distrib. Process., Apr. 2008, pp. 1–14.
[12] S. Cho, J. Martin, R. Xu, M. Hammoud, and R. Melhem, "CA-RAM: A high-performance memory substrate for search-intensive applications," in Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw., Apr. 2007, pp. 230–241.
[13] M. Somasundaram, "Memory and power efficient mechanism for fast table lookup," U.S. Patent 20 060 253 648, Nov. 2, 2006.
[14] M. Somasundaram, "Circuits to generate a sequential index for an input number in a pre-defined list of numbers," U.S. Patent 7 155 563, Dec. 26, 2006.
[15] K. Pagiamtzis and A. Sheikholeslami, "Content-addressable memory (CAM) circuits and architectures: A tutorial and survey," IEEE J. Solid- State Circuits, vol. 41, no. 3, pp. 712–727, Mar. 2006.
[16] Xilinx, San Jose, CA, USA. Xilinx Xpower Analyzer [Online]. Available: http://www.xilinx.com
[17] OSUCells, Stillwater, OK, USA [Online]. Available: http://vlsiarch.ecen. okstate.edu
[18] S.-J. Ruan, C.-Y. Wu, and J.-Y. Hsieh, "Low power design of precomputation-based content-addressable memory," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 16, no. 3, pp. 331–335, Mar. 2008.
[19] H. Noda et al., "A cost-efficient high-performance dynamic TCAM with pipelined hierarchical searching and shift redundancy architecture," IEEE J. Solid-State Circuits, vol. 40, no. 1, pp. 245–253, Jan. 2005.