Design of Low Power L1 Cache Using CBF Based TOB Architecture in Embedded Processors

Mrs.S.BRINDHA[#1], Ms.B.KAVITHA[#2]
[#1]*Assistant professor,* [#2]*P.G.Scholar, II M.E.VLSI Design*
[#1,#2]*Department of Electronics and Communication Engineering,*
[#1,#2]*Avinashilingam Institute for Home Science and Higher Education for Women-University*
Coimbatore, India

Abstract- **In the embedded processor, a cache could consume 40% of the entire chip power. So, reduce the high power utilization of cache is very important. To reduce the high power utilization a new cache method is used in the embedded processors. This is termed as an Early Tag Access (ETA) method. For the memory instructions, the early target way can be found by this ETA. Thereby it can reduce the high power utilization. If the ETA does not find the way, it search the way in L1 Cache. So automatically the power gets increased. Here a new energy efficient matching mechanism referred to as Counting Bloom Filter (CBF) based Tag Overflow Buffer (TOB) is proposed. This TOB uses reduced number of tag bits thereby the power gets reduced. The ETA can be activating only when the TOB hit is occurred. Compared to the previous technique the power consumption gets decreased up to 40%.**

Key words- **Cache, LSQ tag, LSQ TLB, LFSR, Comparator, Hass table, Low Power**

I. INTRODUCTION

In embedded processors the most common and critical problem is to reducing the large power utilization of cache memory. Because it focuses only the low-power consideration in embedded processors. In the total chip power, the cache consumes 40% [1]-[3]. Because of this large power utilization, thermal effects and reliability degradation issues may be generated in the cache. Usually the caches are performed critically because of this large power utilization. Therefore, reduce the large power utilization in the cache is more important. To show the tradeoff between the power and efficiency of the cache, many cache design methods [4]-[13] have been designed under various level of the design abstract. Here, to reduce the high power utilization a new cache matching mechanism termed as a CBF based TOB can be defined. The CBF is area-effective probabilistic data structure; it is used to check whether a tag bit is present in the cluster or set. From which it can reduce the power consumption. For sample, CBFs have been used to Increase the action in multiprocessor snoop-coherent multi-core system [14],[15]. It is also used to reduce the quick miss decision at the L1 cache [16] and to increase the adaptability of load/store ordering queues [17]. There are two types of CBFs are present. That is SRAM CBF (S-CBF) and LFSR CBF (L-CBF). Related to the S-CBF, the L-CBF is more efficient one. Because the LFSR is high efficiency feedbacks shift register. The TOB is the identical mechanism that handles minimum address bits i.e., it use only the most significant bits for matching [18]-[22].

On a hit in the TOB, the restricted-tag cache is approached usually through the ETA. On a miss, common miss method is used i.e., it goes to the L2 cache to searching the particular tag. The following are the main strength of the TOB: 1) In the cache it uses a planned count of address bits. 2) It uses much smaller hardware, so it accomplishes a tag power minimization similar to the other methods. 3) It achieves a reduction of leakage power. In the ETA, the Transition look ahead buffer (TLB) can perform the conversion between the Physical address and the Virtual address. During that conversion a portion of the physical tag is saved in the tag array, in a physical tag and virtual index cache. During the Load Store Queue (LSQ) stage by approaching tag arrays and TLB, the target way can be resolved previously approaching the L1 cache. At the final, the energy consumption can be reduced significantly by accessing single way in the L1 cache. Note that at the LSQ stage the TLB can generate the Virtual addresses that can also be used for succeeding cache approaches. The energy utilities of way resolution at the LSQ stage can be decreased by avoiding the TLB approaches over the cache approaches stage for most of the memory instruction. At the LSQ stage the destination ways cannot be determined for memory instruction. For that an improved technique of the ETA cache is designed at the cache access stage to minimize the number of ways accessed. Note that approaching L2 cache is complete correspond with the approaches to the L1 cache in many high-end processors. Our CBF based TOB technique is fundamentally performed at the L1 cache. For embedded processors, the proposed TOB cache is more effective to take advantage for reducing cache traffic and power utilization by series approaches to the ETA and L1 cache. As correlated with the associated work the output results determine that the considered TOB cache is higher efficient in power minimization.

II.WORKING PRINCIPLE

A. Early Tag Access

In a conventional cache, based on the cache hit the corresponding way of the data only activate. Other ways gets inactive. But the cache miss occurring means automatically it goes to the L1 cache to search the address, so the power gets increase. In this segment, a new cache matching mechanism termed as TOB will be established. This method will decrease the power consumption by reducing the number of tag bits.

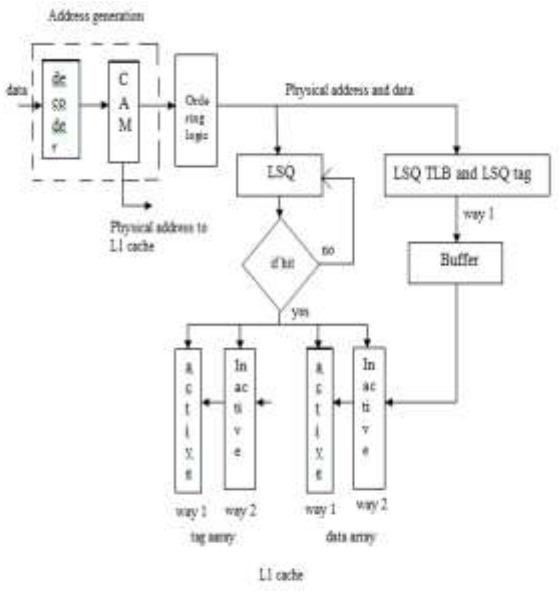

Fig. 1 Operation flow of L1 cache under the ETA cache [1]

To get various powers and efficiency condition in embedded processors, the ETA cache can be explored in the Basic mode.

A.1 Basic Mode

Due to the possibility of memory addresses it is likely to access an operation to the address arrays at the LSQ stage. In the basic mode of ETA cache, a novel group of address arrays and TLB (termed as LSQ tag arrays and LSQ TLB) is executed; when the memory information is go to the LSQ. This new group of LSQ tag array and LSQ TLB is the reproduction of the tag array and TLB of the L1 cache correspondingly, to ignore the data assertion with the L1 cache. In the course of the LSQ lookup performance if the hit happen, that specific way can be used as a target way of the information. If this target way is correct, then that specific way can be activated and other ways get inactive. This enables power saving. On the other hand, if the way miss occurring in the LSQ tag array or in the LSQ TLB during the lookup operation then the L1 cache will be performed in the Normal mode, i.e., in the tag array and data array of the L1 cache, all ways will be activated. From Fig. 1, shows the operation flow of the L1 cache under the ETA cache. Here by using the physical address and the data, the particular way can be determined and activated in the L1 cache.

B. Tag Overflow Buffer

The TOB is the matching mechanism; it uses minimum number of address bit. The main thought of this planned design is to send the most significant bit of the address bits from the cache into an independent register named as TOB, that act as an identifier of the memory instructions present location. In majority of the memory accreditation a minimized –tag cache can be accessed to achieve the dynamic energy efficiency. Various classes of methods are present to minimize the power utilization of the cache by decreasing the number of bits. In some cases a miss may be unfortunately represented as a hit. Because in these cases a subset can be used to compare the address. Therefore that case must be compact with misinterpretation. Some authors will use this design where the misinterpretation cost is low, such as in branch prediction engines [18], [19]. The approach designed by Peng et al.[23] mingle the minimized data array power induced by the modified way-predicting cache design as proposed in [24] with the minimum tag power achieved by restricted tag comparison.

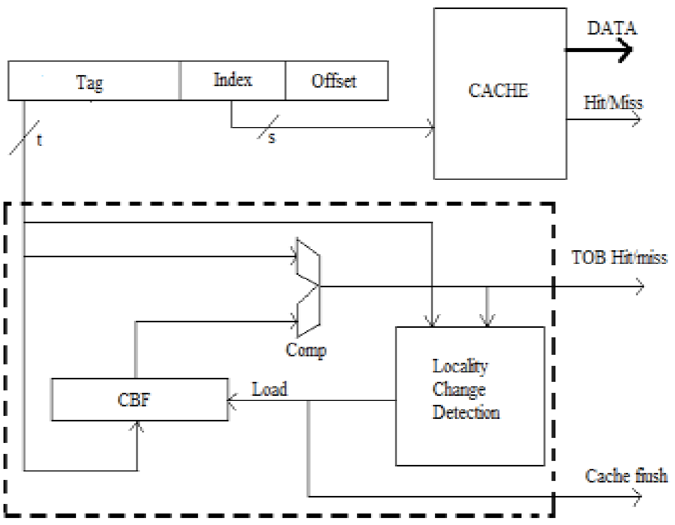

Fig. 2 CBF based TOB architecture

Fig.2 shows the Block Diagram of CBF based TOB architecture. The dotted lines in the figure denote the extra hardware needed to achieve the minimized-tag architecture. For every memory references the TOB and the cache are always performed compliment with each other. The block diagram operates as follows. For each memory references, the most significant bits of the tag are sent to the CBF. If both the address are equal (a CBF hit occurs), then we can easily perform the minimized-tag cache.

C. Counting Bloom Filter

Fig. 3 shows the CBF architecture. A CBF is a group of up/down linear feedback shift register and local zero detectors listed via hash function of the address in membership check.

Fig. 4 shows the user defined 32 bit hash table. Because of hashing, within the same array entry, multiple element addresses are mapped. Normally, the CBF is a limited cluster and the tags are hashed onto this little cluster only.

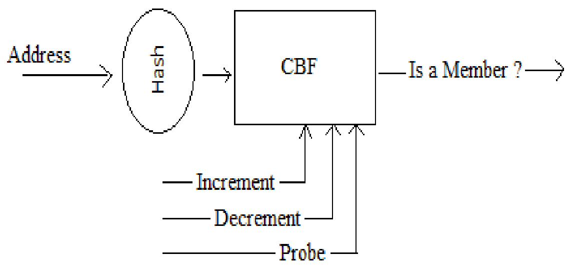

Fig. 3 CBF architecture

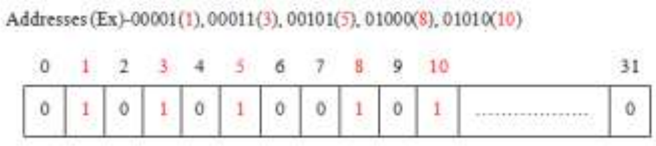

Fig. 4 Hash table (32 bit)

In this hash table, the address present in the cache is denoted as one and others are denoted as zero. For example, 00001,00011,00101,01000 and 01010 are present in the cache means that will be denoted as one. If the particular input address tag is present in cache, then the CBF will be activated. A CBF has three function: 1) increment count (INC); 2) decrement count (DEC); and 3) test if the count is zero or not (PROBE). The INC and DEC operations can increment or decrement the respective count by one based on the hit, and the third one checks weather the count is zero and returns true or false single-bit output. The first two functions are referred as a updates and the third operation will be consider as a probe. Based on the number of inputs and the size of the count per entry, a CBF is characterized. A CBF give any one of the two answers: 1) "Definite no," indicating that the particular address is surely not a member of large set and 2) "I don't know," denote that a CBF cannot find the address in the membership check, and must looking for the large group. The CBF is capable of determine the required answers to the membership check, it saves power and much faster on two conditions. First, CBF serves most membership tests. Second, compared to accessing the large group, the CBF is much sooner and needs less energy. The CBF is performed as follows, at first the large group is empty and all the count is put into zero. In the large set, when the address is added or deleted, the respected CBF gets incremented or decremented by one. The corresponding CBF denotes that in the large set, whether an element is exits or not. If the count is zero, the large set does not have that particular element. If the count is one, then the large set must be searched for that element.

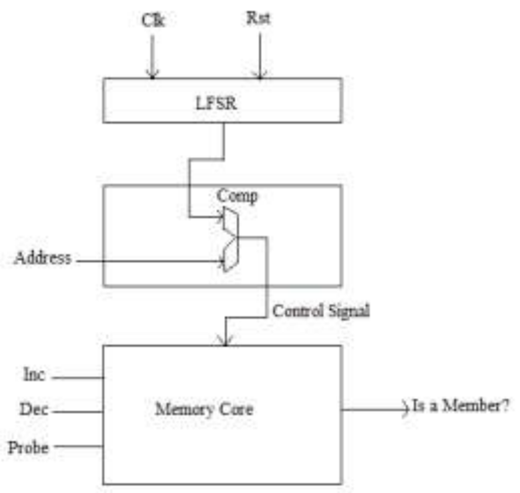

Fig.5 L-CBF architecture; LFSR holds the CBF count; INC/DEC: read-modify-write sequences; PROBE: read-compare sequence

Fig. 5 shows the L-CBF architecture. Here the up/down Leaner Feedback Shift Register is used to generate random number of addresses. By using the comparator both the addresses are correlated to test whether the address is member or not. Fig. 6 indicating the LFSR block diagram. The number of D-flip flops and the XOR gates are used here to generate different addresses. Every LFSR has the following limits: 1) the number of bits in the shift registers is parallel to the width and size of the LFSR. 2) in the LFSR, taps are the output of D-flip flop that have been connected with the feedback loop. 3) at starting state, the LFSR can be any value, except one. Here the total numbers of tag bits are 5. So, the feedback polynomial will be $X^5 + X^3 + 1$. So, the output is taken from 3rd and 5th output of D-flip flop and Xored both of them and feedback to the 1st flip flop.

The input of the LFSR is Clock and reset. The memory core is used to tell whether the element is member or not in the large set.

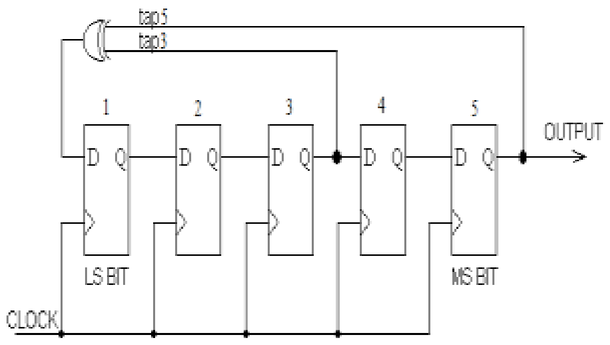

Fig. 6 LFSR Block diagram

It is a single bit output. It can increment the address bits and read the address bits. But it cannot able to decrement it.

D. LSQ Tag Array and LSQ TLB

The LSQ tag array and LSQ TLB are the reproduction of the tag array and TLB of L1 cache accordingly, to ignore the data assertion with the L1 cache. In the LSQ tag array and LSQ TLB the lookup operation can be takes place. When the address go to the LSQ, both the LSQ tag array and LSQ TLB search the early destination way by using the lookup operation. If the hit is occur (the particular address present in the LSQ tag array or TLB), then that particular way gets activate in the L1 cache. Otherwise, the information will be either an early tag miss or early TLB miss.

E. Way Decoder and Way Hit/Miss Decoder

Here the way decoder is used to decode the early destination way from the recently accessed ways by using the way enabling signal. Whether the particular destination way will be correct or not can be denoted by the way hit/miss decoder. The cache performance can be considered as a hit only when both the cache hit/miss and the way hit/miss signals denote a hit. If both the cache hit/miss and the way hit/miss indicating a miss signal means, then the cache process will be consider as a miss. A cache connection problem is found when the cache hit/miss denotes hit and the way hit/miss denotes miss or the early target way does not similar with the actual target way.

III. OVER ALL FLOW OF THE CBF BASED TOB

Fig. 7 shows the flow diagram of the CBF based TOB architecture. The dotted lines denote the CBF based TOB architecture and the ETA. Here the TOB is connected between the address generation part and the ETA. The particular tag bit can be retrieved from the ETA, only when it present in the TOB. If the tag bit not present in the TOB, then the L2 cache or main memory will be searched for the tag bits. It can reduce the search time. So, automatically the power is also gets reduced.

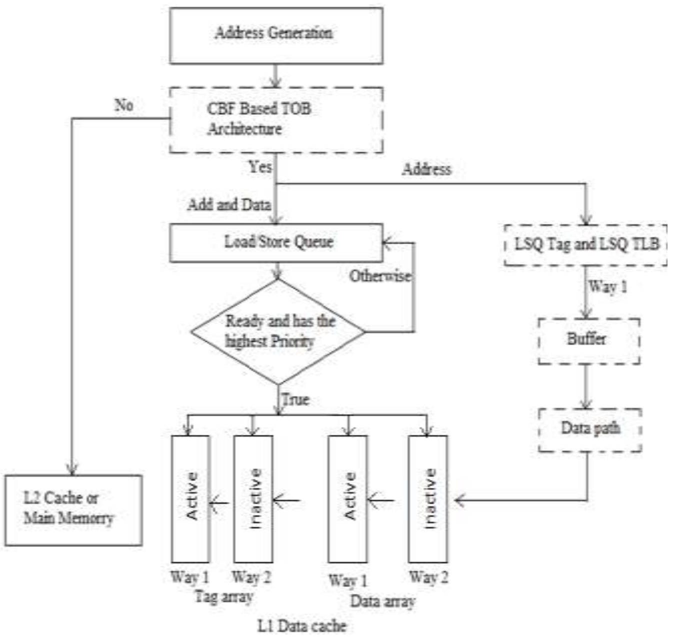

Fig. 7 Flow diagram of CBF based TOB

By using the hash table the CBF can find whether the address is member of a set or not. From that information, the TOB can determine the output. Each time when the address reaches the LSQ tag and LSQ TLB, lookup operation can be takes place to find the early destination way of that particular address in L1 cache. Here the TLB is nothing but the Transition Look ahead Buffer. It is used to convert the physical address to the virtual address. Basically, the physical address is used to find the data array and the virtual address is used to find the tag array. If the particular destination way can be found, then that way can be activated in the L1 cache. From that way we can easily retrieve the data of that particular address.

IV IMPLIMENTATION OF CBF BASED TOB

This segment furnish the VLSI implementation of CBF based TOB cache. Fig. 8 shows the two-way set-associative L1 cache for demonstration. The key component in the proposed is CBF, TOB, Hash table, LFSR, LSQ tag array, LSQ TLB, Information buffer, Way decoder, and way hit/miss decoder.

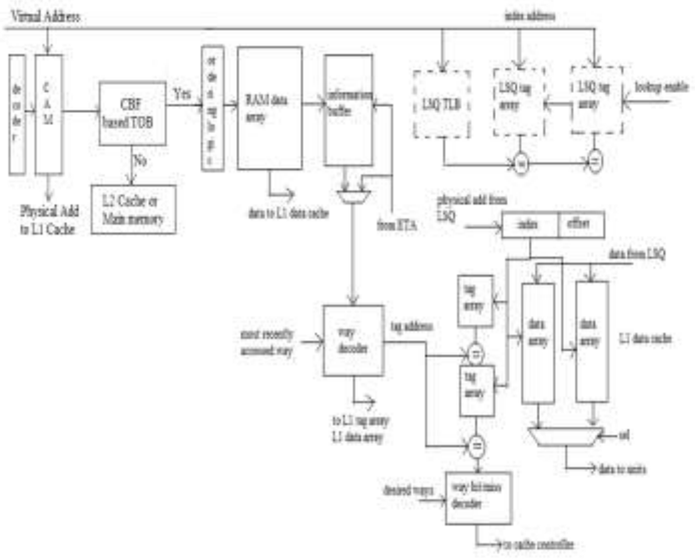

Fig. 8 proposed CBF based TOB cache architecture

Initially the address generation can be takes place by using the CAM. CAM is nothing but Content Addressable Memory. It can generate the address for the particular data. This address will be called as physical address. This address only sent to the L1 cache. In the 8 bit address, the first 5 bits are consider as a tag bits, the next 2 bits are consider as a index bits, and the final bit is consider as a offset bit. The tag bits only sent to the TOB. Because it uses minimum number of address bits to matching the address. If the most significant bits are equal, more over the other bits are also equal.

Then this tag bits are sent to the CBF. In the CBF user defined hash table can be used to search the address bits in the large group. If the address bit is currently in the Hash table means, then it activates the CBF. In the CBF, the LFSR is present to generate the random number of tag bits to compare with the input tag bit. If both the tag bits are equal means, then it sent the control signal '1' to the memory core. In the memory core three types of operations are present. 1) INC, 2) DEC, and 3) PROBE. Increment is used to increase the tag bits in the CBF. The PROBE is used to read the tag bits in the CBF. The CBF cannot able to access the decrement operation.

If the output of LFSR is one means, then the memory core said that "yes, the element is a member of a set". If the output of LFSR is not equal to '1' means, then the memory core output will be "I don't know whether the element is a member of a set or not". Then the CBF output will be compared with the input tag bits in the TOB. If both the tag bits are equal means, TOB hit is occurred. If the TOB miss occurring means, then that particular tag bit is added to the CBF by using Locality Change Detection (LCD). This LCD is based on both the input tag bit and the TOB miss. If the TOB miss occurring, then this tag bit is sent to the LCD. Here based on the TOB miss, the counter gets increased by '1' and compared with the maximum threshold value of the cache. Compared to the threshold value, the input tag bit is minimum means it will load to the CBF. After this action, the cache will be flushed. Then the TOB hit information and the particular tag bit can be sent to the ordering logic. Here first in first out logic can be used. So, the highest priority tag bit is first stored in the information buffer and it is used for the temporary storage of the tag bits. The dotted lines denote the ETA. From the ETA the particular destination way can be determined and it sent to the information buffer. From the information buffer the way can be sent to the way decoder. In the way decoder the recently accessed ways are present. From that ways, the particular destination way can be decoded by using the way enabling signal produced by the comparison of the actual destination way and the early destination way.

Then the address of that destination way can be correlated with the address of the L1 cache. If both the addresses are equal the way hit is occurred, and that will sent to the controller. From that address, the data can be retrieved.

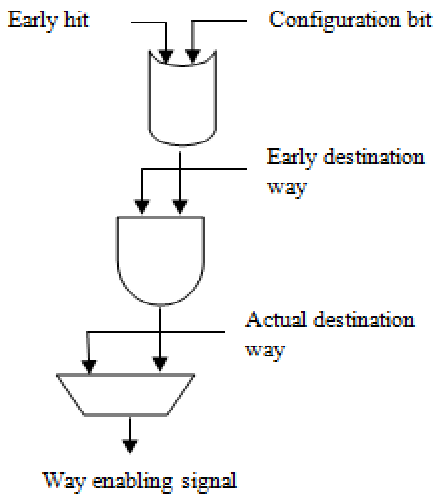

Fig. 9 Implementation of way decoder

Otherwise if the addresses are not equal, then way miss will be occurred. Suppose, if the tag bits are not present in the TOB means, it can directly go to the L2 cache or main memory to retrieve the data.

V SIMULATION RESULT AND COMPARISION TABLE

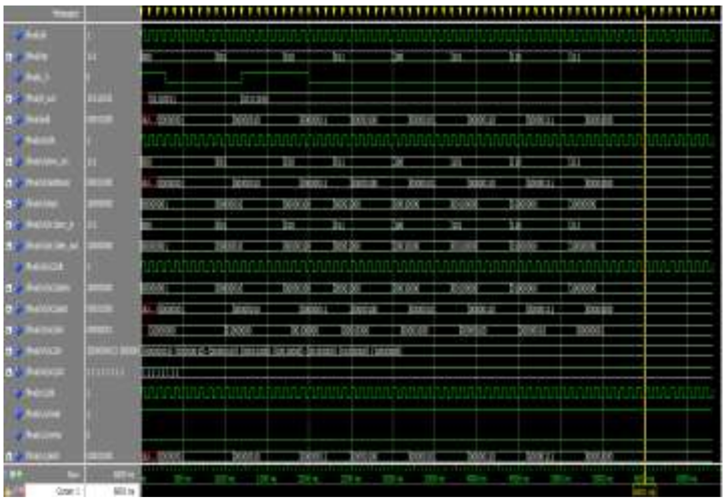

Fig. 10 Simulation result of CBF Based TOB Architecture

The following Table shows the time and power variation between the Early Tag Access and the Counting Bloom Filter based Tag Overflow Buffer.Compared to the ETA technique, CBF based

TOB architecture give the better power and time reduction.

TABLE I

COMPARISION TABLE

Methods Used	Time (ns)	Power (mW)
ETA	11.409	141
CBF Based TOB	6.420	133

VI CONCLUSION

A new power efficient cache design for low-power embedded processors has been implemented here. By using the reduced tag bits, The TOB can determine if the address is existing in the cache or not. If the tag bits existing in the TOB, it will go to the ETA and retrieve the data. Suppose the tag bit not present in the TOB, and then it directly goes to the L2 cache or Main memory to search the address. This can automatically reduce the power from compared to the ETA technique. Here the high power utilization can be minimized with no compromise in behavior. Simulation results show the efficiency of the CBF based TOB architecture as well as the performance impression and design utilities. The TOB architecture has been implemented for the L1 cache alone. Further work is being aimed toward extending this design to other levels of the cache.

REFERENCES

[1] Jianwei Dai, Menglong Guan, and Lei Wang, " Exploiting Early Tag Access for Reducing L1 Data Cache Energy in Embedded Processors," in IEEE transaction on very large scale integration (VLSI) systems, Vol. 22, No. 2, Feb 2014, pp. 396-407.
[2] Intel XScale Microarchitecture, Intel, Santa Clara, CA, USA, 2001.
[3] C. Zhang, F. Vahid, and W. Najjar "A highly-configurable cache architecture for embedded systems," in Proc. 30th Annu. Int. Symp. Comput. Archit., Jun. 2003, pp. 136–146.
[4] S. Segars, "Low power design techniques for microprocessors," in Proc. Int. Solid-State Circuits Conf. Tuts., Feb. 2001.
[5] S. Manne, A. Klauser, and D. Grunwald, "Pipline gating: Spculation conrol for energy reduction," in Proc. Int. Symp. Comput. Archit., Jun.–Jul. 1998, pp. 132–141.
[6] M. Gowan, L. Biro, and D. Jackson, "Power considerations in the design of the alpha 21264 microprocessor," in Proc. Design Autom. Conf., Jun. 1998, pp. 726–731.
[7] A. Malik, B. Moyer, and D. Cermak, "A Low power unified cache architecture providing power and performance flexibility," in Proc. Int. Symp. Low Power Electron. Design, 2000, pp. 241–243.
[8] T. Lyon, E. Delano, C. McNairy, and D. Mulla, "Data Cache Design Considerations for the Itanium Processor," in Proc. IEEE Int. Conf. Comput. Design, VLSI Comput. Process., 2002, pp. 356–362.
[9] D. Nicolaescu, A. Veidenbaum, and A. Nicolau, "Reducing power consumption for high-associativity data caches in embedded processors," in Proc. Design, Autom., Test Eur. Conf. Exhibit., Dec. 2003, pp. 1064–1068.
[10] C. Zhang, F. Vahid, Y. Jun, and W. Najjar, "A way-halting cache for low-energy high-performance systems," in Proc. Int. Symp. Low Power Electron. Design, Aug. 2004, pp. 126–131.
[11] J. Montanaro, R. T. Witek, K. Anne, A. J. Black, E. M. Cooper, D. W. Dobberpuhl, P. M. Donahue, J. Eno, W. Hoeppner, D. Kruckemyer, T. H. Lee, P. C. M. Lin, L. Madden, D. Murray, M. H. Pearce, S. Santhanam, K. J. Snyder, R. Stehpany, and S. C. Thierauf, "A 160-MHz 32-b 0.5- W CMOS RISC microprocessor," IEEE J. Solid-State Circuits, vol. 31, no. 11, pp. 1703–1714, Nov. 1996.
[12] S. Santhanam, A. J. Baum, D. Bertucci, M. Braganza, K. Broch, T. Broch, J. Burnette, E. Chang, C. Kwong-Tak, D. Dobberpuhl, P. Donahue, J. Grodstein, K. Insung, D. Murray, M. Pearce, A. Silveria, D. Souydalay, A. Spink, R. Stepanian, A. Varadharajan, V. R. van Kaenel, and R. Wen, "A low-cost, 300-MHz, RISC CPU with attached media processor," IEEE J. Solid-State Circuits, vol. 33, no. 11, pp. 1829–1838, Nov. 1998.
[13] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: A framework for architectural-level power analysis and optimizations," in Proc. Int. Symp. Comput. Archit., Jun. 2000, pp. 83–94.
[14] A. Moshovos, "RegionScout: Exploiting coarse-grain sharing in snoop-coherence," in Proc. Ann. Int. Symp. Comput. Arch., Jun. 2005, pp. 234–245.
[15] A. Moshovos, G. Memik, B. Falsafi, and A. Choudhary, "Jetty: Filtering snoops for reduced energy consumption in smp servers," in Proc. Ann. Int. Conf. High-Performance Comput. Arch., Feb. 2001, pp. 85–96.
[16] J. K. Peir, S. C. Lai, S. L. Lu, J. Stark, and K. Lai, "Bloom filtering cache misses for accurate data speculation and prefetching," in Proc. Ann. Int. Conf. Supercomput., Jun. 2002, pp. 189–198.
[17] S. Sethumadhavan, R. Desikan, D. Burger, C. R. Moore, and S. W. Keckler, "Scalable hardware memory disambiguation for high-ILP processors," IEEE Micro, vol. 24, no. 6, pp. 118–127, Nov. 2004.
[18] B. Fagin, "Partial resolution in branch target buffers," IEEE Trans. Comput., vol. 46, no. 10, pp. 1142–1145, Oct. 1997.
[19] B.-S. Choi and D.-I. Lee, "Cost-effective value prediction micro-operation using partial tag and narrow-width operands," in Proc. IEEEPacific Rim Conf. Commun., Comput. Signal Process., Aug. 2001, pp. 319–322.
[20] L. Liu, "Partial address directory for cache access," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 2, no. 2, pp. 226–240, Jun. 1994.
[21] R. Min, Z. Xu, Y. Hu, and W.-B. Jone, "Partial tag comparison: A new technology for power-efficient set-associative cache designs," in Proc. 17th Int. Conf. VLSI Des. (VLSID), Jan. 2004, pp. 183–188.
[22] P. Petrov and A. Orailoglu, "Data cache energy minimizations through programmable tag size matching to the applications," in Proc. Int. Symp. Syst. Synth. (ISSS), Sep./Oct. 2001, pp. 113–117.
[23] M. Peng, Y. Pan, and B. Liu, "Low energy partial tag comparison cache using valid-bit pre-decision," in Proc. IEEE Region 10 Conf. (TENCON), Nov. 2006, pp. 1–4.
[24] H.-C. Chen and J.-S. Chiang, "Low-power way-predicting cache using valid-bit pre-decision for parallel architecture," in Proc. 19th Int. Conf. Adv. Inf. Netw. Appl., Mar. 2005, pp. 203–206.