International Journal of Science and Research (IJSR)
ISSN (Online): 2319-7064
Index Copernicus Value (2013): 6.14 | Impact Factor (2013): 4.438

Reducing Cache Energy in Embedded Processors Using Early Tag Access and Tag Overflow Buffer

Neethu P Joseph[1], Anandhi V.[2]

[1]M.Tech Student, Department of Electronics and Communication Engineering
SCMS School of Engineering and Technology, Karukuuty, Cochin, Kerala, India

[2]Associate Professor, Department of Electronics and Communication Engineering
SCMS School of Engineering and Technology, Karukuuty, Cochin, Kerala, India

Abstract: ***Here we propose a new cache design architecture, where two techniques are combined together, one is tag overflow buffer and other is early tag access (ETA). This helps to improve the energy efficiency of data caches in embedded processors. The proposed technique performs the first technique using Tag Overflow Buffer(TOB) and only if a hit occurs the second technique Early Tag Access is performed. Thus only after an initial check for the content in TOB the actual ETA module is made active. The second technique called ETA is used to determine the destination ways of memory instructions much before the actual cache accesses. It, thus, enables only the destination way to be accessed if a hit occurs during the ETA. The proposed ETA cache method can be configured under two operation modes to exploit the trade offs between the energy efficiency and performance. It is shown that our technique is very effective in reducing the number of ways accessed during cache accesses. This enables to achieve significant energy reduction with negligible performance overheads. Simulation results demonstrate that, proposed ETA cache achieves over 52.8% energy reduction on average in the L1 data cache and also in translation lookaside buffer. Compared with the existing cache design techniques, the ETA cache is very effective in energy reduction while maintaining better performance .And the proposed architecture with both TOB and ETA techniques together, have much better performance in terms of power reduction, delay minimization and area reduction. As a justification to above statement simulations using ModelSim and Xilinx are produced.***

Keywords: Early Tag Access (ETA), Tag Overflow Buffer(TOB), cache architecture, energy efficiency

1. Introduction

In embedded systems, energy consumption is a major issue. Each component present will consume some energy for its working. Several studies have shown that cache memories account for about 50% of the total energy consumed in these systems. Performance of the cache architecture is determined by the behavior of the application executing on the architecture. Desktop systems in computers have to accommodate a large range of applications so the cache architecture is often set by the manufacturer as a best compromise for a particular current applications, technology, and cost. Unlike desktop systems, embedded systems are designed in such a way that we can run it in a small range of well-defined applications. In this context, cache architecture that is tuned for this narrow range of applications can have an increased performance with a lower energy consumption. We introduce a novel cache architecture intended for embedded microprocessor platforms. Very large power dissipation could cause many other issues, such as thermal effects and reliability degradation. This problems are compounded by the fact that data caches are usually performance critical. Therefore, it is very important to reduce the energy consumption of cache while minimizing the processor performance. Many different types of cache design techniques have been proposed at different levels of the design abstract to exploit the trade offs between energy used and performance of cache. As caches are typically set-associative, most micro architectural techniques aim at reducing the number of tag arrays and data arrays activated during an access, so that cache power dissipation can be reduced. Phased caches access both tag arrays and data arrays in two different phases. Energy consumption can be reduced greatly because at most only any one data array corresponding to the matched tag, if any, is accessed. Also due to the increase in access cycles, phased caches are usually applied in the lower level memory hierarchy, such as L2 caches, whose performance is comparatively less critical. For L2 caches under the write through policy, a way-tagging technique [2] sends the L2 tag information"s to the L1 cache when the data is loaded from the L2 cache. During the subsequent accesses, L2 cache can be operated in an equivalent direct-mapping manner, there by improving the energy efficiency without incurring performance degradation. To reduce the energy consumption of L1 caches to a great extent, way-predicting techniques make a prediction on the tag and data arrays that the desired data might be correctly located. If the prediction is correct, only one way is accessed at a time to complete the operation; otherwise, all ways are accessed to search for the desired data that will consume a lot of energy. Mispredictions lead to cache re-accesses, which introduce a performance penalty. Another cache design technique, known as way-halting cache, stores some lower order bits of each tag in tag arrays into a fully associative cache and it compares them against the corresponding bits in the coming memory address in parallel with set index decoding. If there is a match, only the corresponding data arrays need to be activated, thereby reducing the energy consumption of cache accesses. This technique, however, requires architectural supports in some cases. In addition, since the search is done in a fully associative cache, the latency of the search might be longer than what the set index decoding takes if the number of sets is large. As a result, this technique could potentially increase critical paths and introduce performance penalty. For very high set-associative(e.g., 32-way) data caches in high end

microprocessors, a technique proposed in employs an additional fully associative memory to record the way information of the recent cache accesses at the Load/Store Queue (LSQ) stage. This memory will be searched prior to a cache access, and the potential destination way can be determined if the address is found. This technique, however, may not be effective for typical L1data caches (e.g., four-way set-associative) commonly used in embedded processors due to the overhead associated with the additional fully associative memory. In addition to static random-access memory (SRAM)-based cache designs, content addressable memory (CAM) is also an option for low-power embedded systems .However, the access latency of CAM is in general longer than SRAM-based caches, in particular when the cache associativity is low (e.g., four-way and eight-way), which is common in high-performance embedded systems. A cache technique, referred to as early tag access (ETA) cache, to improve the energy efficiency of L1 data caches is discussed here. In a physical tag and virtual index cache, a part of the physical address is stored in the tag arrays while the conversion between the virtual address and the physical address is performed by the TLB. By accessing tag arrays and TLB during the LSQ stage, the destination ways of most memory instructions can be determined before accessing the L1 data cache. As a result, only one way in the L1 data cache needs to be accessed for these instructions, thereby reducing the energy consumption significantly. Note that the physical addresses generated from the TLB at the LSQ stage can also be used for subsequent cache accesses. Therefore, for most memory instructions, the energy over head of way determination at the LSQ stage can be compensated for by skipping the TLB accesses during the cache access stage. For memory instructions whose destination ways cannot be determined at the LSQ stage, an enhanced mode of the ETA cache is proposed to reduce the number of ways accessed at the cache access stage. Note that in many high-end processors, accessing L2 tags is done in parallel with the accesses to theL1 cache .This technique is fundamentally different as ETAs are performed at the L1 cache. While many high-end processors utilize parallel LSQ andL1 data cache accesses for performance improvement at the cost of high cache traffic and energy consumption, the proposed ETA cache is more effective for embedded processors, where the accesses to the LSQ and L1 data cache are typically performed in series [7], to take advantage of load forwarding for reducing cache traffic and energy consumption. Note that the proposed ETA cache may also be applied to some general purpose processors. For example, some processors have a dispatch stage during the Load/Store phase, at which the effective addresses of memory operations are available. Thus, upon accessing the L1 data cache, the available destination way can be utilized to reduce the cache energy consumption. Simulation results show that the proposed ETA cache is more effective in energy reduction with better or equal performance as compared with the related work.As an extension to the Early Tag Access (ETA) method a new mechanism called Tag Overflow Buffer (TOB) is also used here so that the energy requirement can be further reduced. This energy-efficient cache architecture with ETA and TOB, is based on a partial-tag scheme, which relies on the idea of bringing most of the tag bits outside the cache into a register that identifies the current locality. On a memory access, this register is first checked against the most significant bits of the address to determine whether we are inside the current locality or not. On a hit, the partial-tag cache is accessed normally (yet with a smaller cost); on a miss, the normal miss procedure is followed, with minor modifications.

2. Literature Survey

2.1 LSQ and Cache Architecture

To reduce conflict misses, set-associative architectures are commonly employed in cache design. Below figure shows a simple two-way set-associative cache and TLB, where the tag and data arrays are the two major components. In the conventional L1 data cache, all tag arrays and data arrays are activated simultaneously for every read/write access to reduce the access latency. Typically, the latency of the L1 data cache is one clock cycle, though this latency could be higher in deeply pipelined processors. On the other hand, accesses to the tag arrays can always be finished in one cycle. Due to the temporal/spatial locality inherent in various programs, data will stay for a while once they have been brought into the data cache. This implies that the tag arrays will keep their contents unchanged except when a cache miss occurs.

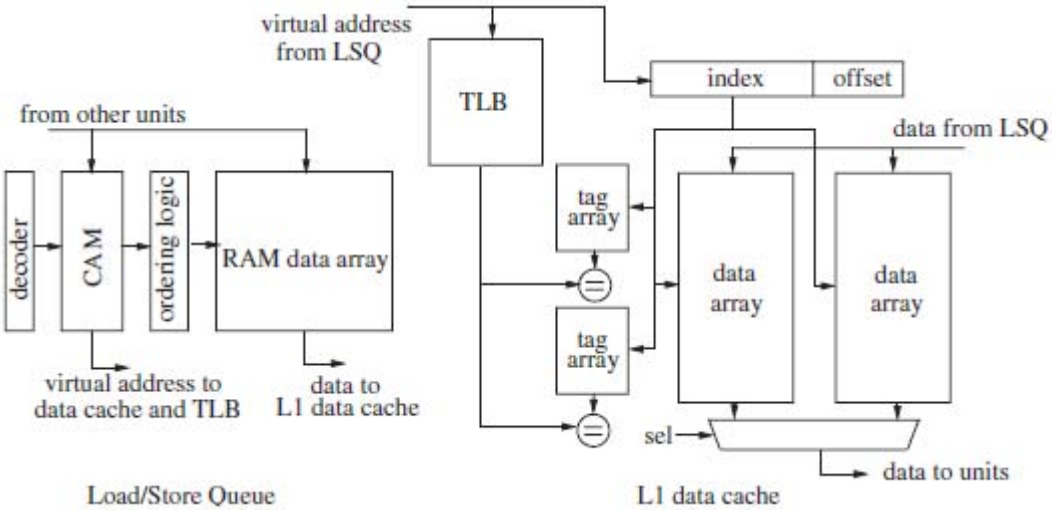

Figure 1: Conventional architecture of LSQ and L1 data cache

Next figure shows a portion of the pipeline between the address generation stage and the memory stage in a typical embedded processor, where the LSQ and L1 data cache are accessed in series [6] to take advantage of load forwarding for reducing cache traffic and energy consumption. A load/store instruction will be sent to the LSQ before being issued to the data cache. Meanwhile, the instruction is compared with the existing ones in the LSQ to determine whether the instruction can be issued to the data cache at the next clock cycle. If not, the instruction will stay in the LSQ stage for more than one clock cycle. From this architecture, we observe that: 1) the memory address of a load/store will be available in the LSQ stage for at least one clock cycle before being issued to the data cache, while the access to the tag arrays can be finished in one cycle and 2) due to the temporal/spatial locality, the tag arrays will not be updated until a cache miss occurs. Since in the conventional architecture, the destination way of a memory instruction cannot be determined before the cache access stage, all the ways in the L1 data cache need to be activated during a cache access for performance consideration at the cost of energy consumption. Based on these observations, we propose a new cache technique in the next section to improve the energy efficiency of L1 data caches.

2.2 ETA Cache Architecture

There are many cache types available out of which in conventional set-associative cache, all ways in the tag and data arrays are accessed simultaneously. The requested data, however, only resides in one way under a cache hit. The extra way accesses incur unnecessary energy consumption. In this section, a new cache architecture referred to as ETA cache will be developed. The ETA cache reduces the number of unnecessary way accesses, thereby reducing cache energy consumption. To accommodate different energy and performance requirements in embedded processors, the ETA cache can be operated under two different modes: the basic mode and the advanced mode.

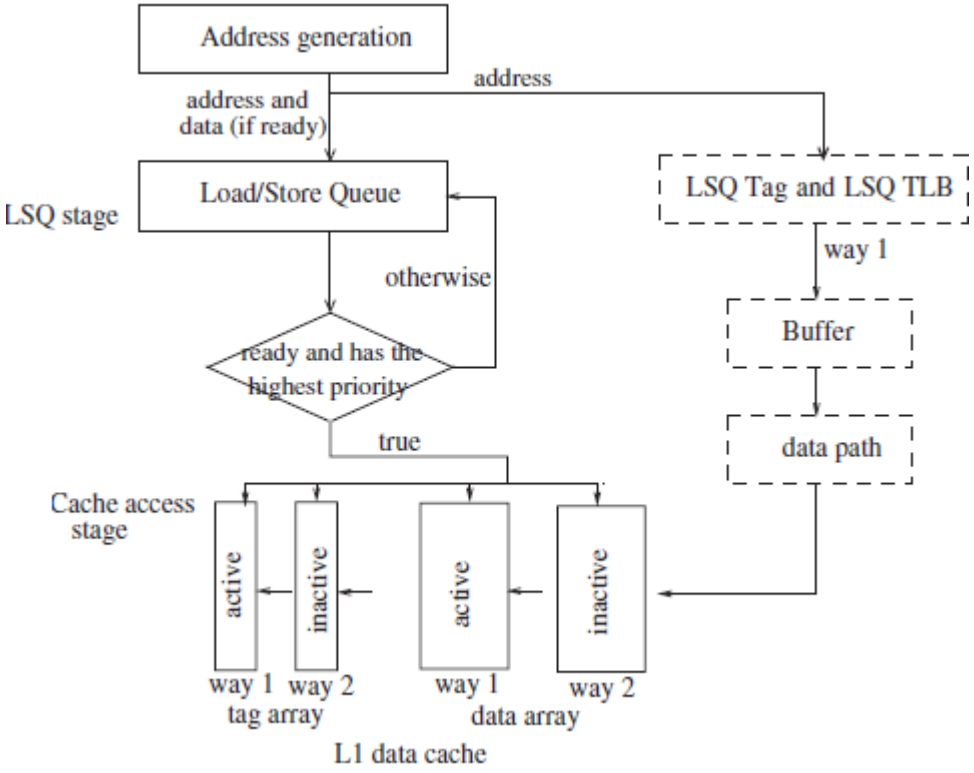

Figure 2: Architecture of First order masked AES

2.2.1 Basic Mode

It is possible to perform an access to the tag arrays at the LSQ stage due to the availability of memory addresses. In the basic mode of the ETA cache, each time a memory instruction is sent into the LSQ, an access to a new set of tag arrays and TLB is performed. This new set of LSQ tag arrays and LSQ TLB are implemented as a copy of the tag arrays and TLB of the L1 data cache, respectively, to avoid the data contention with the L1 data cache. If there is a hit during the LSQ lookup operation, the matched way in the LSQ tag arrays will be used as the destination way of this instruction when it accesses the L1 data cache subsequently. If this destination way is correct, only one way in the L1 data cache needs to be activated and thus enables energy savings. On the other hand, if a miss occurs during the lookup operation in the LSQ tag arrays or in the LSQ TLB, the L1 data cache will be accessed in a conventional manner, i.e., all ways in the tag arrays and data arrays of the L1 data cache will be activated. We can see that the two sets of tag arrays and TLB are accessed at two different stages: LSQ stage and cache access stage. It is possible that the early destination way of a memory instruction determined at the LSQ stage is not the same as the actual one determined at the cache access stage due to the cache misses that happen in between. This causes a cache coherence problem if the memory instruction simply follows its early destination way, if any, to access the L1 data cache. To avoid this problem, the ETA cache in the basic mode requires the memory instruction to access all the

ways in the actual tag arrays during the cache access stage. During the same time, the data arrays are also accessed in parallel using the early destination way only (as the actual destination way is not available yet) to reduce energy consumption while maintaining performance. The destination way obtained from the actual tag arrays is then compared with the early destination way to detect any cache coherence problem. If a cache coherence problem is detected, the accessed data from the data arrays is discarded and an additional access is performed.

2.2.2 Advanced Mode

There are some memory instructions whose early destination ways cannot be determined due to either early tag misses or early TLB misses. This could be significant for applications with a high miss rate. Early tag/TLB misses at the LSQ stage are usually associated with real misses at the cache access stage.When a memory instruction with either an early tag miss or an early TLB miss accesses the L1 data cache, the tag arrays of the L1 data cache are accessed first. In most cases this will be a real cache miss, and thus the L1 cache access is completed without the need to access the data arrays of the L1 data cache, thereby saving energy. In rare cases if a cache hit occurs, the actual destination way is obtained from the tag arrays and only this way is accessed in the data arrays at the next clock cycle. By applying this advanced mode, the energy consumption can be reduced because most of the memory instructions with early tag/TLB misses do not need to access the data arrays. Some of these instructions have real cache hits, thereby causing an extra cycle in the cache access. Since the number of such cases is very small, the performance overhead is negligible. The advanced mode can be turned off if performance is truly critical.

2.3 Tag Overflow Buffer

The idea behind the proposed architecture is based on moving a large number of the tag bits from the cache into an external register (Tag Overflow Buffer) that identifies the current locality of the memory references; additional hardware allows to dynamically update the value of the reference locality contained in the buffer. Energy efficiency is achieved by using, for most of the memory accesses, a reduced-tag cache. The TOB and the cache are always accessed in parallel at each memory reference. The outcome of the TOB lookup tells whether an access to the reduced-tag cache is feasible. On a memory reference, the t - k most significant bits of the address are fed to the TOB. If the two values match (a *TOB hit*), then we can safely access the reduced-tag cache without worrying about false hits. Clearly, the lookup in the cache may result in a hit or miss as any access to a regular cache. In case of cache miss, the missed line would be replaced using some replacement strategy.

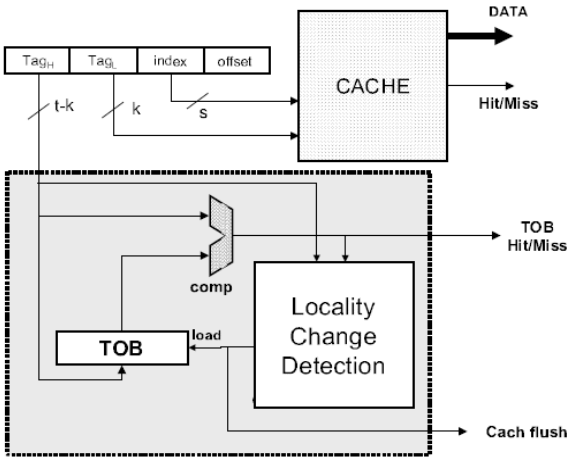

Figure 3: TOB based architecture

A TOB miss, conversely, regardless of the possible cache outcome, results in an equivalent cache miss. In fact, a TOB miss corresponds to a change in locality; a corresponding hit in the cache would result in a false hit, since the full tag does not actually match. The functional operations of the dynamic architecture are summarized in Table below.

Table 1: Functional operation of dynamic architecture

TOB	Cache	Description
H	H	Regular cache hit, no replacement
H	M	Regular cache miss, with replacement
M	-	If a chance in locality is detected Cache flush; otherwise, regular cache miss, without replacement.

Concerning the actual implementation of the locality change detection, we opted for a very simple realization, based on the sole observation of the TOB miss output. Specifically, a change in locality is determined as the number of consecutive TOB misses exceeding a given threshold.

2.4 Counting Bloom Filter (CBF)

An increasing number of architectural techniques rely on hardware counting bloom filters (CBFs) to improve upon the energy, delay and complexity of various processor structures. CBFs improve the energy and delay of membership tests by maintaining an imprecise and compact representation of a large set to be searched[6]. A Bloom filter is an inexact representation of a set that allows for false positives when queried; that is, it can sometimes say that an element is in the set when it is not. In return, a Bloom filter offers very compact storage: less than 10 bits per element are required for a 1% false positive probability, independent of the size or number of elements in the set. There has recently been a surge in the popularity of Bloom filters and variants, especially in networking .One variant, a counting Bloom filter, allows the set to change dynamically via insertions and deletions of elements.

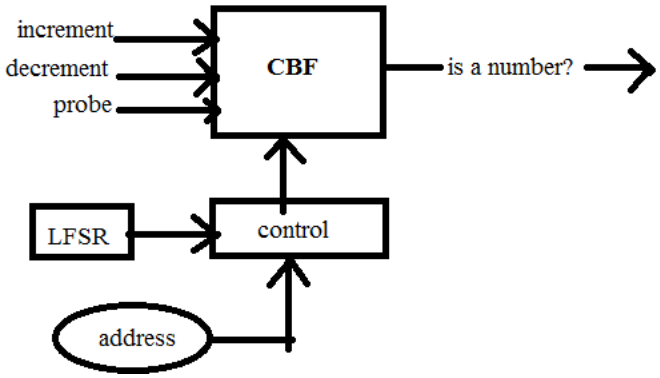

Figure 4: Block diagram Counting Bloom Filter

2.4.1 Some characteristics of CBF are:

1) CBF as a Black Box: a CBF is conceptually is an array of counts indexed via a hash function of the element under membership test. A CBF has three operations: (i) increment count (INC), (ii) decrement count (DEC), and (iii) test if the count is zero (PROBE). The first two operations increment or decrement the corresponding count by one, and the third one checks if the count is zero and returns true or false (single bit output).
2) CBF Characteristics: Membership tests using CBFs are performed by probe operations. In response to a membership test, a CBF provides one of the following two answers: (i)"definite no", indicating that the element is definitely not a member of the large set, and (ii) "I don't know", implying that the CBF cannot assist in a membership test, and the large set must be searched.
3) CBF Functionality: The CBF operates as follows: Initially, all counts are set to zero and the large set is empty. When an element is inserted into, or deleted from the large set, the corresponding CBF count is incremented, or decremented by one. To test whether an element currently exists in the large set, we inspect the corresponding CBF count.

5.2 Linear Feedback Shift Register (LFSR)

LFSR is used to give control signal to CBF for select member or not. A linear-feedback shift register (LFSR) is a shift register whose input bit is a linear function of its previous state. The most commonly used linear function of single bits is exclusive-or (XOR). Thus, an LFSR is most often a shift register whose input bit is driven by the XOR of some bits of the overall shift register value. The initial value of the LFSR is called the seed, and because the operation of the register is deterministic, the stream of values produced by the register is completely determined by its current (or previous) state. Likewise, because the register has a finite number of possible states, it must eventually enter a repeating cycle.

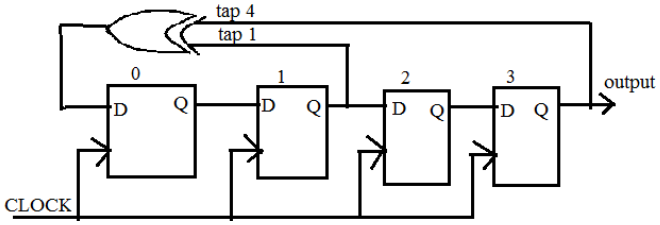

Figure 5: Linear Shift Register

3. Architecture of Modified Cache

First the address is generated and it is goes to TOB architecture. Then this architecture is using matching mechanism for partial tag comparison. So here take most significant bits of tag address for comparison. Counting Bloom Filter is used to check that address is in or not and The LFSR is give signal to control that is enable or disable. For example address is there means the signal is enable and the cache is hit otherwise cache miss. The Cache hit mean further action is performed that is Early Tag Access

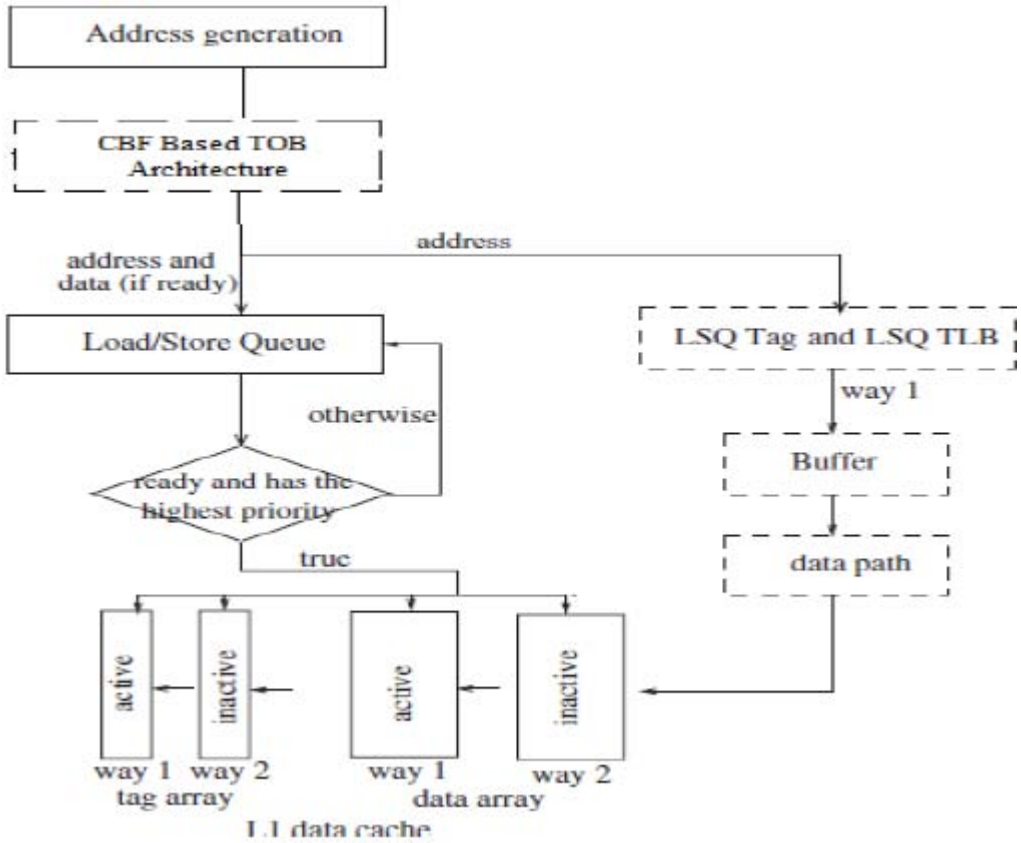

Figure 6: Architecture of Modified cache

This Early Tag Access cache reduces the number of unnecessary way accesses thereby reducing cache energy consumption. Two types of operations are performed in LSQ tag: Lookup (read) ,Update (write) This information stored in information buffer until next instruction will issue. It is keep that instruction in a period of one clock cycle. Then the desired way only active in L1 data cache stage using this technique. Suppose miss means to perform re-access.

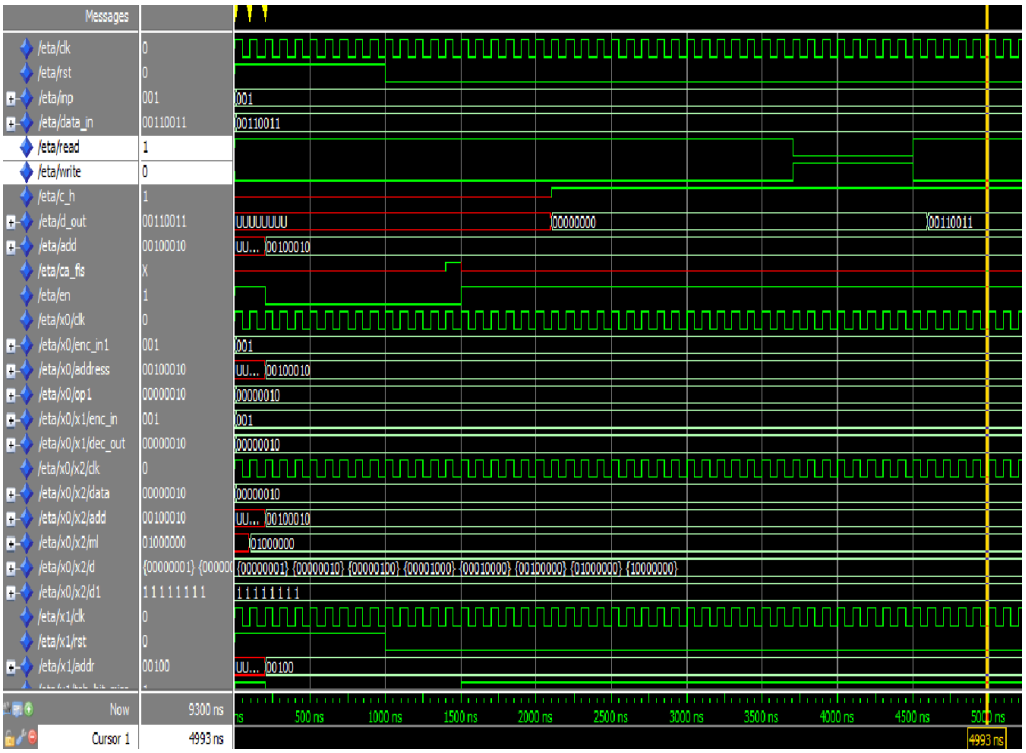

Figure 7: Simulation result of modified cache

Figure 4.2 shows the simulation of cache architecture with ETA and TOB .The simulation displays clock ,input, datain, eta read, eta write, eta data output along with other signals of sub-modules.

Timing Summary

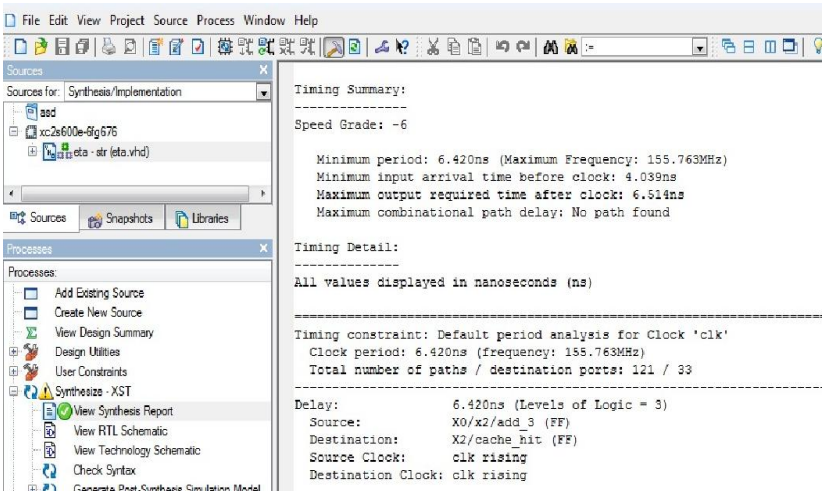

Figure 8: Timing summary of modified ETA

Design Summary

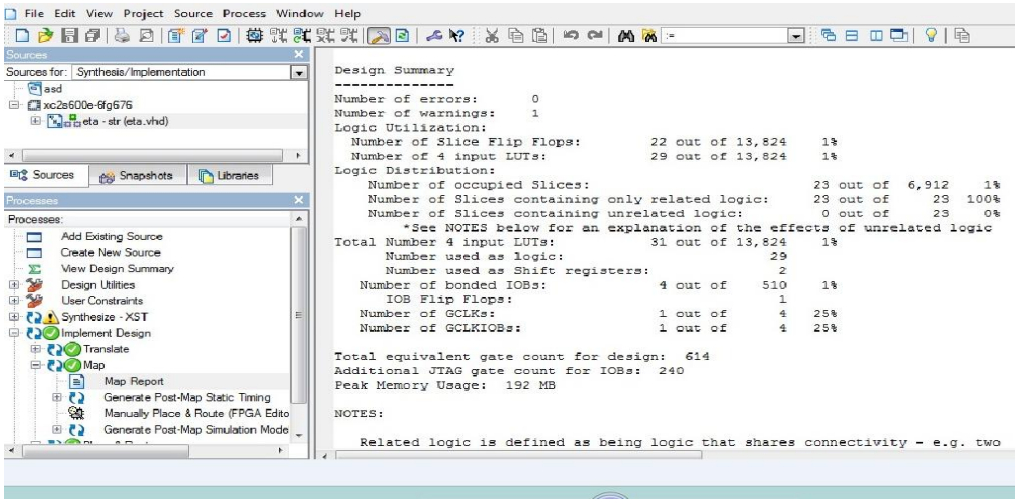

Figure 9: Design summary of modified ETA

Power Summary

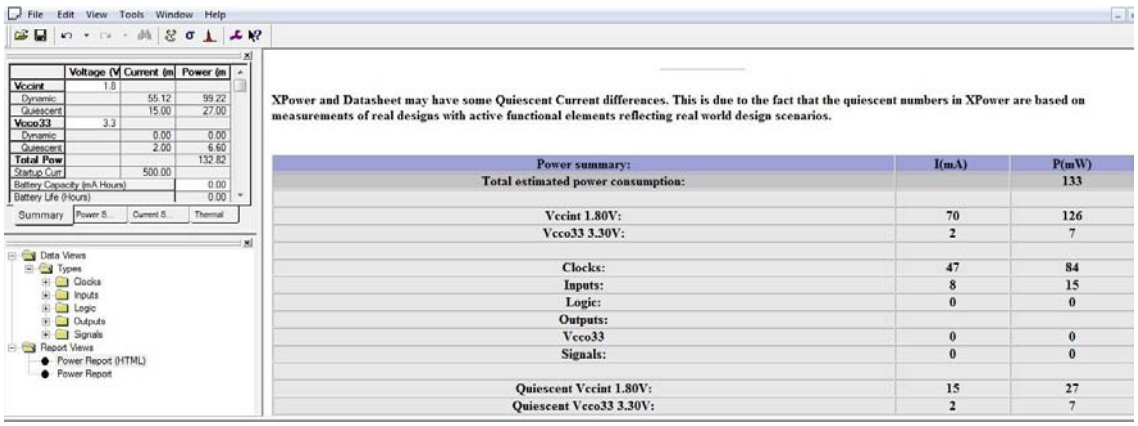

Figure 10: Power summary of modified ETA

4. Inferences

Power is calculated in milli Watt (mW) and the comparison is calculated. Table below shows how time, frequency and power has improved in cache with ETA and TOB comparison to the ETA method. It shows that modified technique requires minimal power. Here power is reduced by several mW using proposed architecture when compared to existing architecture which is based on simple ETA.

Table 2: Comparison on Power consumption, Time and Frequency

	ETA	Cache with ETA andTOB
Time(ns)/Freq(MHz)	7.505	6.420
Freq(MHz)	133.245	155.763
Power(mW)	147	133

Graph below shows bar representation of delay:

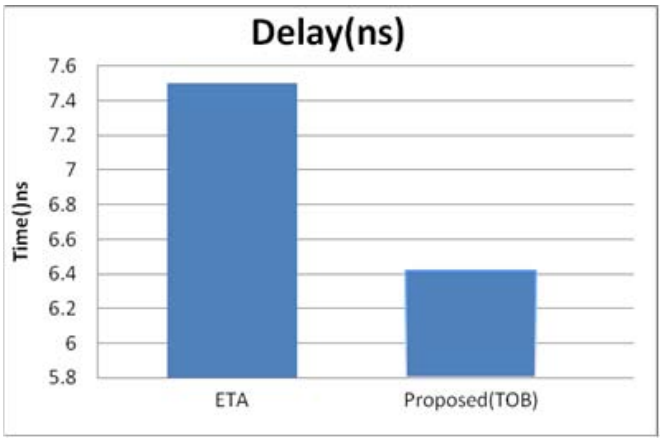

Figure 11: Delay Comparison

Graph below shows bar representation of power:

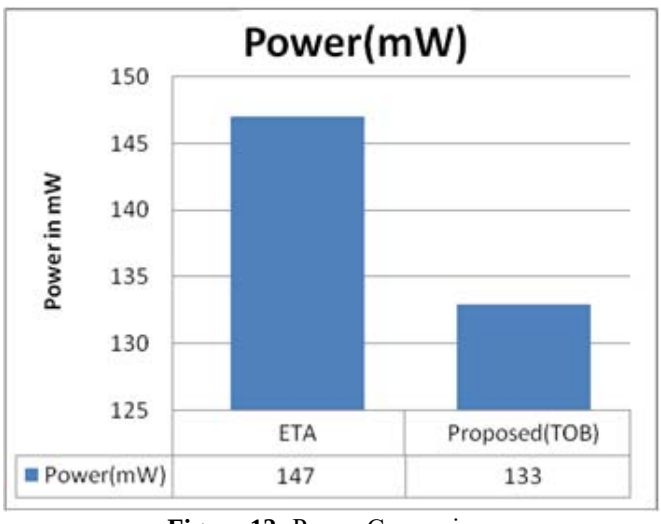

Figure 12: Power Comparison

Graph below shows bar graph representation of frequency:

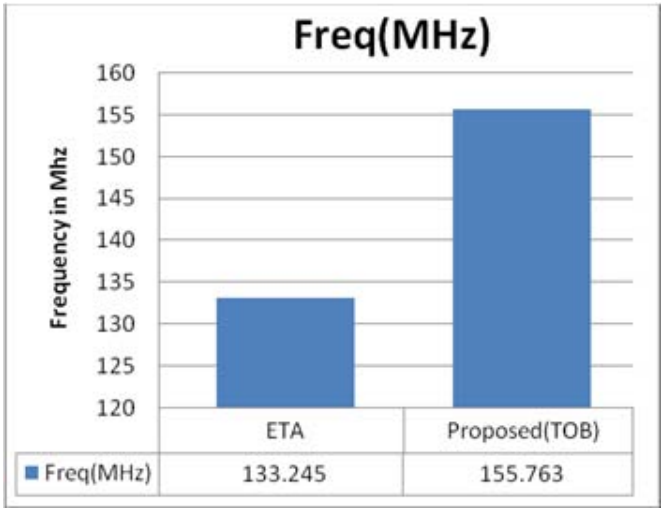

Figure 13: Graph representation of frequency

5. Future Scope

In this paper both Early Tag Access (ETA) and Tag Overflow Buffer (TOB) techniques are used for L1 data cache alone. As a future scope we can extend the same algorithm to other levels of cache too so that we can further reduce the energy consumption effectively.

6. Conclusion

This paper presented a new energy-efficient cache design technique for low-power embedded processors. The proposed design uses two techniques called Early Tag Access and Tag Overflow Buffer to predicts the destination way of a memory instruction at the early LSQ stage. Thus, only one way needed to be accessed during the cache access stage if the prediction is correct, thereby reducing the energy consumption significantly. By applying the idea of phased access to the memory instructions whose early destination ways cannot be determined at the LSQ stage, the energy consumption can be further reduced with negligible performance degradation. Tag overflow buffer helps to predict the cache miss or cache hit before we make use of the second technique i.e Early Tag Access. Thus only if TOB hit occurs the destination way is determined. Thus helping to reduce the power consumption to a great extend.Simulation results demonstrated the effectiveness of the proposed technique as well as the performance impact and design overhead. While our technique was demonstrated by a L1 data cache design, future work is being directed toward extending this technique to other levels of the cache hierarchy and to deal with multithreaded workloads.

References

[1] Ms.T.Nathiya, Mr.R.Kandasamy,M.E., „Design Of Low Power L1 Cache For Embedded Processor"International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 4 Issue 4, April 2015.

[2] Jianwei Dai,Menglong Guan, and Lei Wang,senior Member,IEEE ,"Exploring Early Tag Access for Reducing L1 Data cache Energy in Embedded

Processors," IEEE transactions on very large scale intergration system,VOL.22,NO.2,Feb 2014.
[3] J. Dai and L. Wang, "An energy-efficient L2 cache architecture using way tag information under write-through policy," IEEE Trans. VeryLarge Scale Integr. (VLSI) Syst., vol. 21, no. 1, pp. 102–112, Jan. 2013.
[4] Elham Safi, Andreas Moshovos, and Andreas Veneris,"L-CBF: A Low-Power, Fast Counting Bloom Filter Architecture," TVLSI-00034-2007.R1

Author Profile

Neethu P Joseph received the B.Tech degree in Applied Electronics And Instrumentation Engineering from Mahatma Gandhi University, Kerala at Rajagiri College Of Engineering And Technology 2013, after that she joined for M.Tech and currently she is pursuing her M.Tech degree in VLSI and Embedded systems under the same university in SCMS School of Engineering and Technology, Cochin.