Way Halted Prediction Cache : An Energy Efficient Cache Architecture for Embedded Processors

Neethu Bal Mallya, Geeta Patil and Biju Raveendran
BITS Pilani K.K. Birla Goa Campus, Goa
{h2009191,geetapatil,biju}@goa.bits-pilani.ac.in

Abstract— **This paper proposes a novel cache architecture – Way Halted Prediction – to reduce energy consumption and effective access time of set associative caches. This is achieved with the help of halt tag array and prediction circuit. Experimental evaluation of various SPEC benchmark programs on CACTI 5.3 and CASIM simulators reveal that the proposed architecture offers 33%, 6% and 3% savings in dynamic energy consumption and 1.80%, 6.13% and -1.95% saving in effective access time over conventional, way predicting and way halting cache architectures respectively.**

Keywords—cache architecture; energy efficient cache design; way predicting; way halting;

I. INTRODUCTION

The growing speed mismatch between processor and memory can be alleviated by the introduction of high speed multilevel cache memories. The advancements in fabrication technologies made the modern processors more powerful with more transistors embedded in it. The reduced transistor size offers lesser dynamic energy for switching at the cost of higher static energy because of leakage. The on chip cache plays an important role in performance and energy consumption of the modern processors. As the cache memory consumes significantly large portion of the processor energy - approximately 42% in StrongARM processor [1] and 23% in PowerPC [2, 3] - it is not practically feasible to increase the cache size indefinitely.

The increase in cache size results in reduced dynamic energy consumption because of high cache hit rate at the cost of static energy consumption. Designers has to find the right configuration of cache size, cache line size and associativity for the applications and then apply additional energy optimization techniques to reduce static and dynamic energy consumption without much performance degradation.

Though direct mapped cache offers least per access energy consumption and hardware complexity, most of the embedded architectures prefer set associative (SA) cache because of less overall dynamic energy consumption and effective access time. Analysis reveals that data arrays, tag arrays, decoder, sense amplifiers, comparators, pre-charge circuit and output drivers are the major sources of energy consumption in conventional cache architecture [4]. For a SA cache, a cache hit with optimal energy and time can be achieved by accessing a tag array, a data array, decoder, sense-amplifier, comparator, pre-charge circuit and output driver. A cache miss with optimal energy and time can be achieved by accessing only the decoder.

Researchers proposed various energy optimization strategies to offer performance closer to the ideal scenario with the minimal overhead. The cache architectures like way predicting (WP) cache [5] and way halting (WH) cache [6] tries to reduce the number of active ways during cache hit. WP cache reduces the number of active ways to one when the prediction is correct. Otherwise it compares all the N ways with a penalty of an additional cycle time. WH cache reduces the number of active ways to k where $1 \leq k \leq N$ for hit and $0 \leq k \leq N$ for miss without overhead.

The proposed work – Way Halted Prediction (WHP) cache – aims at reducing the number of active ways to one in case of prediction hit and active ways to k, where k is the number of ways halted in case of prediction miss. The remainder of the paper is organized as follows. Section 2 reviews some of the recent approaches in energy-efficient cache architecture. Section 3 describes in detail about WHP cache architecture. Section 4 presents time and energy models of various cache architectures. Experimental setup and performance comparison of various cache architectures is done in section 5. Section 6 discusses the conclusion and the possible applications of the proposed work.

II. RELATED WORK

Various techniques were proposed in recent past to improve the energy efficiency of cache architecture. Though recent technology advancements made static energy an equally important component as dynamic energy, most of the existing techniques addresses dynamic energy consumption.

Cache memory related dynamic energy consumption can be reduced either by increasing the cache hit rate or by reducing the per access energy consumption. The most common approach to improve the cache hit rate with minimum energy consumption is to find the correct configuration of cache size, cache line size and associativity for the application. Zhang et al. [7] proposed a reconfigurable cache architecture – way concatenation and way shutdown – which can configure the cache size, cache line size and associativity dynamically. Way concatenation cache uses two bit configurable register to select associativity between direct map, 2-way and 4-way. Way concatenation cache configures cache line size as 16, 32 and 64 bytes. Way shutdown cache saves static and dynamic energy consumption by shutting down unused ways based on application's memory usage, there by reconfiguring the associativity and cache size. Bournoutian et al. [8] uses statically calculated miss rate to decide on associativities. The reconfigurable cache architectures reduce the static and dynamic energy consumption at the cost of additional hardware complexity and reconfiguration time.

1063-9667/15 $31.00 © 2015 IEEE
DOI 10.1109/VLSID.2015.16

Cache memory offers very high spatial locality as majority of cache accesses are for the frequently accessed instructions and data. In [9], Jones et al. proposed way placement (WPL) cache which uses static profiling to determine the most frequently used instructions. WPL cache uses way hint bit to decide whether to use the profiling information or not. When the way hint bit is set, the way corresponding to the least significant $\log_2$(associativity) bits of the tag is accessed which results in saving dynamic energy consumption.

Tag less cache (TLC) proposed by Sembrant et al. [10] reduces the dynamic energy in virtually indexed physically tagged cache. TLC eliminates tag comparison in data cache by using extended TLB (eTLB), which stores the way location and valid bit of each cache line of the page. eTLB eliminates tag comparisons and detects early cache misses. TLC accesses only one data array during cache hit and does not access data array during cache miss.

The per access energy consumption can be minimized by partitioning the cache vertically or horizontally. Horizontal cache partitioning schemes reduces per access cache hit energy by introducing a small but powerful cache between the processor and L1 cache. The cache architectures like loop cache and buffer cache [11] falls in this category. Though cache hit energy consumption is low, the energy and time required for cache miss is high. Vertical partitioning [12] schemes try to achieve the ideal cache access scenario, i.e., accesses only one tag and one data array for a cache hit and accesses no tag and data array for a cache miss. This is achieved by dividing the cache into banks and providing bank-wise access control. Hasega et al. [5] proposed phased cache (PC) which accesses cache in two phases. During first phase, it compares all the tags of the given set index. If there exist a match, it accesses the data in next phase. PC achieves energy saving at the cost of additional time and performance.

Inoue et al. [5] proposed way predicting (WP) cache architecture. WP enables the predicted way's tag and data array in first cycle. It compares the input tag bits with predicted way's tag bits in indexed set. If the tag matches, it accesses the data in the same cycle. If the tag is not matching in predicted way, then it enables all the other (N-1) tag and data arrays. If any of the (N-1) tags matches with input tag, the data is accessed, otherwise cache miss is executed. The energy saving of this architecture is around 60% for a 4-way set associative cache with a slight performance degradation because of prediction miss penalty. WP cache achieves ideal scenario during prediction hit and the energy saving depends on the prediction hit rate.

Batson et al.[13] proposed reactive associative (RA) cache to improve the prediction hit rate of WP cache. In first cycle, all the tag and data arrays of either direct mapped way or the predicted way is accessed. The data is accessed in the same cycle if the tag matches in direct mapped way or predicted way. If the tag match is found in any other way, then the data access happens in second cycle. RA cache do not incur additional overhead during cache miss as it detects cache miss in first cycle itself. The prediction accuracy of RA is better than the WP cache.

To improve the energy efficiency further by avoiding unnecessary tag comparison of WP and RA caches, Zhang [6] proposed way halting (WH) cache architecture. WH cache uses a small fully associative halt tag array per way to store the least significant 4 tag bits of each set in that way. The halt tag array is compared with 4 LSB's of address to reduce unnecessary accesses. As the halt tag comparison is happening in parallel with decoding, there is no performance degradation. All the ways where halt tag is hit, is compared with remaining tag bits to find whether there exists a match. WH cache performs better than WP and RA caches.

Though WP cache saves dynamic energy during prediction hit, it consumes extra energy and time during prediction miss. WH cache saves dynamic energy during halt miss but does not offer optimal energy during halt hit as it may enable more than one way for comparison and access. To improve the energy efficiency and access time further, a way halted prediction (WHP) cache architecture is proposed in this paper.

III. PROPOSED WORK

This paper proposes an energy efficient cache architecture named Way Halted Prediction (WHP) cache which exploits the advantages of WH and WP cache architectures. WP, WH and WHP cache architectures activate only the matched ways thereby achieving dynamic energy savings over conventional set-associative cache architecture. The WHP cache architecture is shown in Fig. 1.

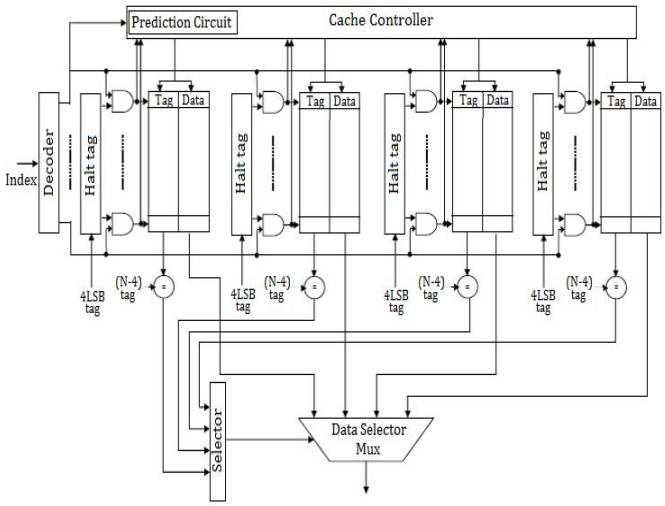

Fig. 1. Way Halted Prediction Architecture

WHP uses halt tags for the early detection of probable hit ways in decode cycle. The detection of miss in decode cycle is achieved when no halt tag hits exist. When more than one halt tag hits exist, WHP uses prediction circuit to predict the most recently accessed way. If the predicted way is a halt tag miss, then all the ways with halt tag hit are accessed in parallel to reduce cycle time. When predicted way is a hit, the energy consumption of WHP will be much lesser than WH cache. The detailed algorithm which illustrates the working of WHP cache architecture is given below.

Algorithm Way Halted Prediction

Input
Physical address P

Output
Hit / Miss and the requested data

BEGIN
Decode P's index bits to locate the desired set and
Compare Halt tag array with low order 4 bits of P's tag.
Count number of Halt-tag hit ways, C in the desired set
If C is 0, then goto C3
If C is 1, then goto C1
Else, find Predicted Way, W_P and goto C1
C1: If C is 1, then
Enable the tag and data array of the halt tag hit way
Compare tag with P's remaining tag bits
If tag matches, PHIT=1 and goto C2
Else goto C3
Else
If W_P is halt tag hit way, then
Enable the tag and data array of W_P
Compare W_P's tag with P's remaining tag bits
If tag matches, PHIT =1 and goto C2
Else PHIT=0 and goto C2
Else
Enable the tag and data arrays of all halt tag hit ways
Compare tags with P's remaining tag bits
If tag matches, PHIT =1 and goto C2
Else goto C3
C2: If PHIT = 1
Transfer requested data from / to Processor
return
Else
Enable tag and data arrays of all C-1 halt tag hit ways
Compare tags with P's remaining tag bits
If tag matches, then
Transfer requested data from / to Processor
return
Else goto C3
C3: Transfer data from Lower level cache to Cache and transfer data from / to Processor
return
END

The eight cache access scenarios of WHP cache architecture are categorized as WHP with only Halt tag in use and WHP with Halt tag and Way prediction in use. The three scenarios where only Halt tag is used are: (a) Halt hit in only one way and cache hit, (b) Halt hit in only one way and cache miss and (c) Halt miss. In case (a) and (b), only a tag array and a data array are accessed. The case (a) offers the ideal cache hit scenario of set associative cache. Both these cases offer energy saving without increasing the effective cache access time. The case (c) takes only decoding time to decide on cache miss. This result in zero tag and data array access which is the ideal cache miss scenario of set associative cache. The early detection of cache misses result in energy saving with improved effective cache access time.

The way prediction circuit in WHP cache is used only when the halt tags are hit in more than one way. The two further categorizations of this are: (i) Predicted way is a Halt hit way (ii) Predicted way is a Halt miss way. The three scenarios in former case are: (a) Prediction hit, Cache hit, (b) Prediction miss, Cache hit and (c) Prediction miss, Cache miss. The case (a) offers ideal cache hit scenario by accessing only a tag array and a data array. The prediction miss scenarios, i.e., case (b) and (c) consumes additional time like in the case of WP cache. The energy consumption of WHP cache in these cases will be much lesser than WP cache and is equal to WH caches as it accesses only k tag arrays and data arrays where $k < N$. If the predicted way is a halt miss way, then the WHP cache checks all the k halt hit ways directly for cache hit /miss. This results in saving energy without increasing the effective access time.

In each of the eight scenarios, WHP cache offers lesser, in the worst case equal dynamic energy consumption in comparison with conventional, WP and WH caches. In all the eight scenarios, WHP cache offers lesser in the worst case equal effective cache access time in comparison with conventional and WP cache. Two of the eight scenarios, i.e., prediction miss and cache hit, prediction miss cache miss, takes more effective access time than the WH cache. The experimental results substantiate these facts.

IV. Energy and Time Model

A. Energy Modeling

The cache components used in the evaluation of energy consumption of various cache architectures are shown in Table 1. E_{halt}, E_{pred} and E_{Tx} represents the energy overhead of the halt tag array, energy overhead of the prediction circuit and penalty for a cache miss respectively.

TABLE I. Cache Components for Energy and Time Modeling

Cache Component	Energy Components		Total Energy	Time Components	
	Data Side	Tag Side		Data Side	Tag Side
Decoder	E_{ds_dec}	E_{ts_dec}	$E_{dec} = E_{ds\_dec} + E_{ts\_dec}$	T_{ds_dec}	T_{ts_dec}
Bit lines	$E_{ds\_bline}^*$	$E_{ts\_bline}^*$	$E_x = E_{ds\_bline} + E_{ts\_bline} + E_{ds\_sa} + E_{ts\_sa} + E_{ds\_pre} + E_{ts\_pre} + E_{ts\_cmp}$	T_{ds_bline}	T_{ts_bline}
Sense Amplifier	$E_{ds\_sa}^*$	$E_{ts\_sa}^*$		T_{ds_sa}	T_{ts_sa}
Pre-charge Circuit	$E_{ds\_pre}^*$	$E_{ts\_pre}^*$		T_{ds_pre}	T_{ts_pre}
Comparator	-	$E_{ts\_cmp}^*$		-	T_{ts_cmp}
Output Driver	$E_{ds\_op\_drvr}$	$E_{ts\_op\_drvr}$	$E_{op\_drvr} = E_{ds\_op\_drvr} + E_{ts\_op\_drvr}$	$T_{ds\_op\_drvr}$	$T_{ts\_op\_drvr}$

* - *Energy dissipation for the components is for 1-way*

Conventional Cache:

Cache Hit Energy:

$E_{Conv\_CHit} = E_{dec} + N * E_x + E_{op\_drvr}$

where N is the Associativity

Cache Miss Energy:

$E_{Conv\_CMiss} = E_{dec} + N * E_x + E_{op\_drvr} + E_{Tx}$

Total Dynamic Energy:

$E_{CONV} = C_{Hit} * E_{Conv\_CHit} + (1 - C_{Hit}) * E_{Conv\_CMiss}$

where C_{Hit} is the Cache Hit Rate

Way Predicting Cache:

Prediction Hit, Cache Hit Energy:

$E_{PHit\_CHit} = (E_{dec} + E_{pred}) + E_x + E_{op\_drvr}$

Prediction Miss, Cache Hit Energy:

$E_{PMiss\_CHit} = (E_{dec} + E_{pred}) + N * E_x + E_{ts\_op\_drvr} + E_{op\_drvr}$

Prediction Miss, Cache Miss Energy:

$E_{PMiss\_CMiss} = (E_{dec} + E_{pred}) + N * E_x + E_{ts\_op\_drvr} + E_{op\_drvr} + E_{Tx}$

Total Dynamic Energy:

$E_{WP} = C_{PHit} * E_{PHit\_CHit} + (C_{Hit} - C_{PHit}) * E_{PMiss\_CHit} + (1 - C_{Hit}) * E_{PMiss\_CMiss}$

where C_{PHit} is the Prediction Hit Rate

Way Halting Cache:

Halt Hit, Cache Hit Energy:

$E_{HHit\_CHit} = (E_{dec} + E_{halt}) + k*E_x + E_{op\_drvr}$

where k is the number of halt-tag hit ways activated

Halt Hit, Cache Miss Energy:

$E_{HHit\_CMiss} = (E_{dec} + E_{halt}) + k*E_x + E_{op\_drvr} + E_{Tx}$

Halt Miss, Cache Miss Energy:

$E_{HMiss\_CMiss} = (E_{dec} + E_{halt}) + E_{Tx} + E_{ds\_op\_drvr}$

Total Dynamic Energy:

$E_{WH} = C_{Hit} * E_{HHit\_CHit} + (C_{HHit} - C_{Hit}) * E_{HHit\_CMiss} + (1 - C_{HHit}) * E_{HMiss\_CMiss}$

where C_{HHit} is the Halt Hit Rate

Since k can be different for different scenarios, the energy is calculated separately for the access and summed up as E_{WH}.

Way Halted Prediction Cache:

Halt Hit in 1 way, Cache Hit Energy:

$E_{HHit1\_CHit} = (E_{dec} + E_{halt}) + E_x + E_{op\_drvr}$

Halt Hit in 1 way, Cache Miss Energy:

$E_{HHit1\_CMiss} = (E_{dec} + E_{halt}) + E_x + E_{op\_drvr} + E_{Tx}$

Halt Hit in k ways, W_P is in Halt Hit ways, Prediction Hit, Cache Hit Energy:

$E_{HHit\text{-}k\_Wp\_PHit\_CHit} = (E_{dec} + E_{halt} + E_{pred}) + E_x + E_{op\_drvr}$

Halt Hit in k ways, W_P is in Halt Hit ways, Prediction Miss, Cache Hit Energy:

$E_{HHit\text{-}k\_Wp\_PMiss\_CHit} = (E_{dec} + E_{halt} + E_{pred}) + k*E_x + E_{ts\_op\_drvr} + E_{op\_drvr}$

Halt Hit in k ways, W_P is in Halt Hit ways, Prediction Miss, Cache Miss Energy:

$E_{HHit\text{-}k\_Wp\_PMiss\_CMiss} = (E_{dec} + E_{halt} + E_{pred}) + k*E_x + E_{ts\_op\_drvr} + E_{op\_drvr} + E_{Tx}$

Halt Hit in k ways, W_P not in Halt Hit ways, Cache Hit Energy:

$E_{HHit\text{-}k\_CHit} = (E_{dec} + E_{halt} + E_{pred}) + k*E_x + E_{ts\_op\_drvr} + E_{op\_drvr}$

Halt Hit in k ways, W_P not in Halt Hit ways, Cache Miss Energy:

$E_{HHit\text{-}k\_CMiss} = (E_{dec} + E_{halt} + E_{pred}) + k*E_x + E_{ts\_op\_drvr} + E_{op\_drvr} + E_{Tx}$

Halt Miss, Cache Miss Energy:

$E_{HMiss\_CMiss} = (E_{dec} + E_{halt}) + E_{Tx} + E_{ds\_op\_drvr}$

Total Dynamic Energy:

$E_{WHP} = C_{HHit1\_CHit} * E_{HHit1\_CHit} + C_{HHit1\_CMiss} * E_{HHit1\_CMiss} + C_{HHit\text{-}k\_Wp\_PHit\_CHit} * E_{HHit\text{-}k\_Wp\_PHit\_CHit} + C_{HHit\text{-}k\_Wp\_PMiss\_CHit} * E_{HHit\text{-}k\_Wp\_PMiss\_CHit} + C_{HHit\text{-}k\_Wp\_PMiss\_CMiss} * E_{HHit\text{-}k\_Wp\_PMiss\_CMiss} + C_{HHit\text{-}k\_CHit} * E_{HHit\text{-}k\_CHit} + C_{HHit\text{-}k\_CMiss} * E_{HHit\text{-}k\_CMiss} + C_{HMiss\_CMiss} * E_{HMiss\_CMiss}$

Since k can be different for different scenarios, the energy is calculated separately for the access and summed up as E_{WHP}.

B. Time Modeling

The access time representations of the cache components are shown in Table 1. T_{Halt}, T_{Pred}, T_{Tx} and T_{1Cycle} are the time for accessing halt tag array, time for accessing the prediction circuit, transfer time from/to processor and effective access time for a single HIT respectively.

The cycle time for the set-associative cache is calculated as:

$T_{1Cycle} = \max \{(\max(T_{ds\_dec}, T_{ts\_dec}, T_{Halt}) + T_{Pred}), \max (T_{ds\_bline} + T_{ds\_sa}), (T_{ts\_bline} + T_{ts\_sa} + T_{ts\_cmp} + T_{ts\_opdrvr}), (T_{ds\_opdrvr} + T_{Tx})\}$

Total access time for the different cache architectures are:

Conventional Cache:

$T_{CONV} = (3 * C_{Hit} + 23 * C_{Miss}) * T_{1Cycle}$

Way Predicting Cache:

$T_{WP} = (3 * C_{PHit} + 4 * (C_{Hit} - C_{PHit}) + 24 * (1 - C_{Hit})) * T_{1Cycle}$

Way Halting Cache:

$T_{WH} = (3 * C_{HHit} + 23 * (C_{HHit} - C_{Hit}) + 21 * (1 - C_{HHit})) * T_{1Cycle}$

Way Halted Prediction Cache:

$T_{WPH} = (3*C_{HHit1\_CHit} + 23*C_{HHit1\_CMiss} + 3*C_{HHit\text{-}k\_Wp\_PHit\_CHit} + 4*C_{HHit\text{-}k\_Wp\_PMiss\_CHit} + 24*C_{HHit\text{-}k\_Wp\_PMiss\_CMiss} + 3*C_{HHit\text{-}k\_CHit} + 23*C_{HHit\text{-}k\_CMiss} + 21*C_{HMiss\_CMiss}) * T_{1Cycle}$

V. Experimental Setup and Evaluation

A. Experimental Setup

This work uses CASIM, a single cycle simulation framework to model the L1 cache of different set-associative cache architectures. The simulation framework implemented in C estimates the access time, hit rate, prediction hit rate, tag comparisons per access, dynamic energy and static energy for various configurations of set-associative caches like conventional (CONV), WP, WH and WHP. This framework takes the address size as 32 bits. The replacement policy used is Least Recently Used (LRU). Since the same replacement policy is used for all cache architectures, the cache-hit rate (CHR) is same for all the caches. MRU is used for way prediction in prediction based architectures.

The framework accepts different cache configurations – associativity, cache size and cache line size – as its inputs. CASIM uses SPEC95 benchmark programs to evaluate the performance of various cache architectures. The cache configurations like associativity, cache size and cache line size varies from 2-way to 16-way, 4KB to 32KB and 8B to 32B respectively. CACTI 5.3 time and power estimations for the given cache configuration in 32nm technology is imported to the framework.

B. Experimental Analysis

1) Dynamic Energy per Access

Fig. 2 shows the dynamic energy consumption per cache access for a 4-way set-associative cache of 32KB size and 16B line size using CONV, WP, WH and WHP cache architectures.

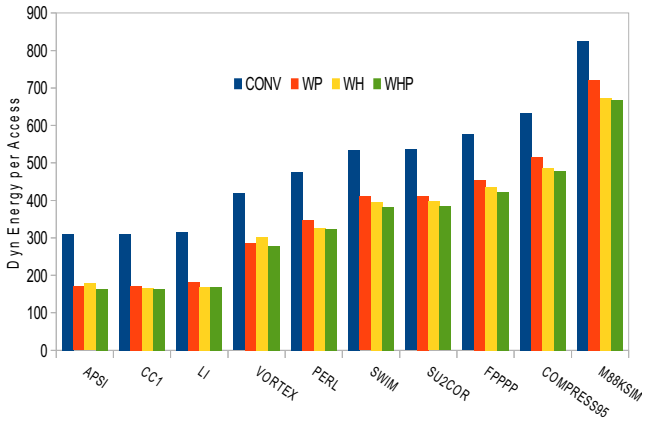

Fig. 2. Dynamic Energy Per Access for different SPEC95 benchmark programs

Results show that all the energy efficient architectures consume less dynamic energy in comparison with CONV cache. For the various SPEC benchmark programs tested, the average dynamic energy saving of WP, WH and WHP caches over CONV cache are 28.93%, 31.28% and 33.31% respectively.

The prediction accuracy of WHP is always higher than the WP cache because of the halting circuit. For the SPEC benchmark program suite, the average prediction accuracy is measured as 85.25% and 89.29% for WP and WHP cache architectures respectively. It is also observed that whenever prediction accuracy is greater than 90%, WP cache offers better energy saving than WH cache. For instance, the benchmark programs like VORTEX and APSI offers very high prediction accuracy which results in better energy saving for WP cache over WH cache. However, WHP cache offers better energy saving in comparison with all the other architectures. Experimental evaluation reveals that WHP cache architecture achieves on an average 6% and 3% of dynamic energy saving over WP and WH caches respectively.

Fig. 3 shows the per access dynamic energy consumption of CONV, WP, WH and WHP cache architectures over various associativities for a 32KB cache of 16B line size. It is observed that per access dynamic energy consumption increases with increase in associativity for all the cache architectures because of the number of active ways. The increase in number of active ways is substantially less for energy efficient caches over CONV cache. Among energy efficient architectures, WHP cache offers the least per access energy consumption followed by WH and WP caches. When prediction accuracy is very high, WP cache offers better energy efficiency over WH cache. On an average, the per access dynamic energy saving for WP, WH and WHP caches over CONV cache are 31.20%, 34.21% and 37.16% respectively.

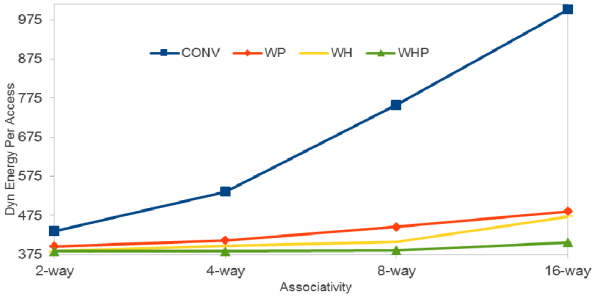

Fig. 3. Dynamic Energy Per Access for different Associativities

Fig. 4 shows the per access dynamic energy consumption of CONV, WP, WH and WHP cache architectures over different cache sizes for a 4-way SA cache of 16B line size. For all the cache configurations, WHP cache offers the least per access energy consumption over CONV, WP and WH caches. It is also observed that WH cache architecture performs better than CONV and WP cache architectures when the prediction accuracy is less than 90%.

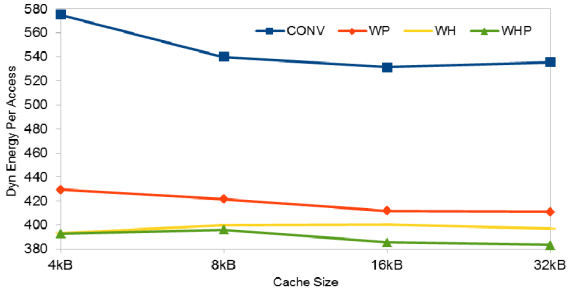

Fig. 4. Dynamic Energy Per Access for different Cache Size

2) Effective Per Access Time

Fig. 5 shows the per access time for a 4-way SA cache of 32KB cache size and 16B line size using CONV, WP, WH and WHP cache architectures. The early detection of cache miss because of halt tag miss without any additional overhead makes WH cache offer the least per access time. Though WHP cache detects all the halt tag misses as in WH cache, per access time of WHP cache is slightly higher than WH cache due to prediction miss penalty. However, per access time of WHP cache is lesser than CONV and WP caches. Per access

time of WP cache is high because of the prediction circuit. The experimental results shows that on an average WH takes 1.99%, 3.82% and 8.24% less effective access time as compared to WHP, CONV and WP respectively.

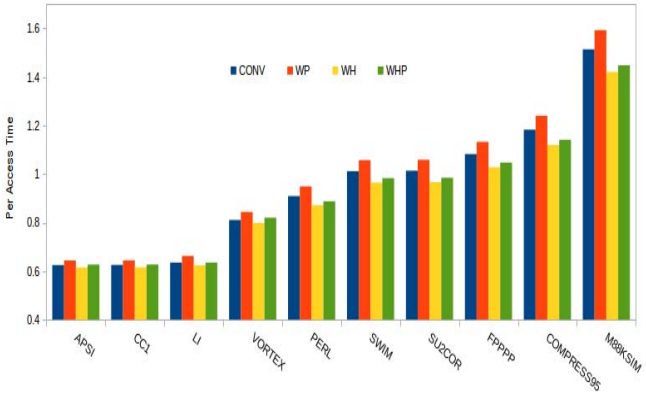

Fig. 5. Per Access Time for different SPEC95 benchmark programs

Fig. 6 shows the effective access time of CONV, WP, WH and WHP cache architectures over various associativities for a 32KB cache of 16B line size. It is observed that effective access time increases with increase in associativity for all the cache architectures.

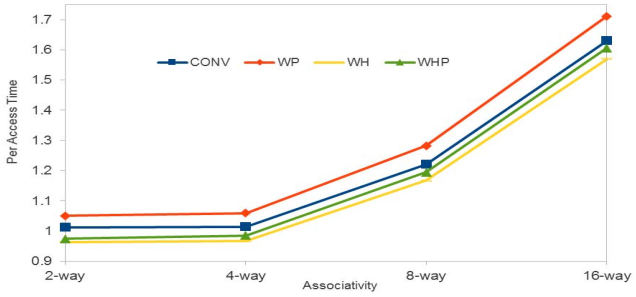

Fig. 6. Per Access Time for different associativities

Fig. 7 shows the effective access time of CONV, WP, WH and WHP cache architectures over various cache sizes for 4-way SA cache of 16B line size. It is observed that effective access time increases with increase in cache size for all the cache architectures.

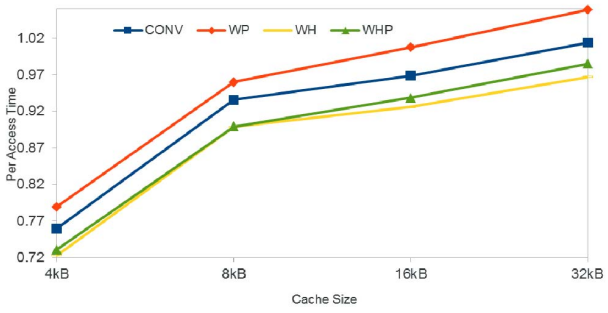

Fig. 7. Per Access Time for different cache sizes

VI. CONCLUSION

The paper proposed an energy efficient cache architecture - Way Halted Prediction. WHP cache uses halt tag array and prediction circuit to achieve reduced energy consumption and effective cache access time. The combination of halt tag and prediction circuit reduces the number of ways to be activated for cache access. In WHP cache, the number of active ways are reduced from k ways in case of WH cache to one way in most of the accesses with the help of prediction circuit. As the prediction circuit is enabled only when k>1, the performance of WHP cache is improved with respect to energy and time. The simulation results shows that WHP offers better energy efficiency over the other architectures analyzed. WHP offers 33%, 6% and 3% dynamic energy saving and 1.80%, 6.13% and -1.95% saving in effective access time with an area overhead of 0.7%, 0.35% and 0.35% over the conventional, way predicting and way halting cache architectures.

REFERENCES

[1] J. Montanaro, R. T. Witek, et al., "A 160MHz 32b 0.5W CMOS RISC microprocessor," IEEE International Solid- State Circuits Conference, vol. 39, pp. 214 - 215, Feb 1996.

[2] R. Bechade et al., "A 32b 66MHz 1.8W Microprocessor," Proceedings of the International Solid-State Circuits Conference, 1994

[3] O. S. Unsal, I. Koren, C. M. Krishna and C. A. Moritz , "The Minimax Cache: An Energy-Efficient Framework for Media Processors," In proceedings 8th International Symposium on High-Performance Computer Architecture, pp. 131 – 140, 2002

[4] S. J. E. Wilton and N. P. Jouppi, "CACTI: an enhanced cache access and cycle time model," In IEEE Journal of Solid-State Circuits, vol. 31, Issue: 5, pp.677 –688, May 1996

[5] K. Inoue, T. Ishihara and K. Murakami, "Way-Predicting Set-Associative Cache for High Performance and Low Energy Consumption," In International Symposium of Low Power Electronics and Design, pp. 273 – 275, 1999

[6] C. Zhang, F. Vahid, J. Yang and W. Najjar, "A Way-Halting Cache for Low-Energy High-Performance Systems," In ACM Transactions on Architecture and Code Optimization (TACO), vol. 2, Issue 1, pp. 34 – 54, 2005

[7] C. Zhang, F. Vahid and W. Najjar, "A Highly Configurable Cache for Low Energy Embedded Systems," In ACM Transactions on Embedded Computing Systems(TECS), vol. 4, Issue 2,pp. 363 – 387, 2005

[8] G. Bournoutian and A. Orailoglu, **"**Application-aware adaptive cache architecture for power-sensitive mobile processors," In Journal of ACM Transactions on Embedded Computing Systems, vol. 13 Issue 3, Dec 2013 Article No. 41

[9] T. M. Jones, S. Bartolini, B. D. Bus, J. Cavazos and M. F. P. O'Boyle, "Instruction Cache Energy Saving Through Compiler Way-Placement," In Proceedings of the conference on Design, automation and test in Europe, pp. 1196-1201, 2008

[10] A. Sembrant, E. Hagersten and D. Black-Shaffer, "TLC a tag-less cache for reducing dynamic first level cache energy," In Proceedings of the 46th Annual IEEE International Symposium on Microarchitecture, pp. 49-61, 2013

[11] L. H. Lee, B. Moyer, J. Arends, "Instruction Fetch Energy Reduction Using Loop Caches For Embedded Applications with Small Tight Loops," In Proceedings of the 1999 international symposium on Low power electronics and design, pp. 267-269, 1999

[12] S. Ching-Long and M. Alvin, "Cache Design Trade-offs for Power and Performance Optimization: a case study," In proceedings of the 1995 international symposium on Low power design, pp. 63-68, 1995

[13] B. Batson and T. N. Vijaykumar, "Reactive-Associative Caches," In Proceedings of International Conference of Parallel Architectures and Compilation Techniques, pp. 49 - 60, 2001.