Phased Set Associative Cache Design For Reduced Power Consumption

Rajesh Kannan Megalingam, Deepu .K.B, Iype P. Joseph, Vandana Vikram
Department of Electronics and Communication Engineering
Amrita School Of Engineering, Amritapuri
rajeshm@amritapuri.amrita.edu, deepu13589@gmail.com, iype_06ec24@students.amrita.ac.in,
vandnavikram@yahoo.co.in

Abstract— **In this paper, improvised versions of the set associative cache accessing technique have been proposed to reduce the power consumption. In phased cache the cache-access process is divided into two phases. In the first phase all the tag in the set are examined in parallel. In the next phase, if there is a hit, then a data access is performed for the hit way. The average energy consumption is reduced as we are not accessing the data together with tag in each phase. Behavioral implementation of these mechanisms was carried out using Verilog HDL. Synthesis of the design was done in Xilinx 10.1. The Xilinx Xpower analyzer is used to find the power consumption. The results show an average of 41% reduction in power consumption as compared to the conventional sequential set associative cache and an average of 21% power reduction as compared to conventional parallel set associative cache architecture.**

Keywords- set associative, cache accessing technique and HDL.

I. INTRODUCTION

Microprocessors use on chip cache to reduce the access time between processor and memory which in turn results in reduced power consumption. But as the size of on-chip cache started increasing, the energy dissipation per unit area also started increasing proportionately. The various ways for reducing cache energy dissipation includes the use of alternative cache organizations, way predictions, selective turning-off of various parts of cache, feeding the cache with different voltages and low power design based on value frequencies[1][2][3][4][6].

Commonly used cache architectures are direct mapped, fully associative and set associative (N way set associative). Direct mapped caches are easier to search but results in lot of data contention and cache trashing. Fully associative caches are difficult to search but results in very less data contention. N way set associative caches are easier to search than fully associative cache. In the set associative cache algorithm, the processor will search for data or instruction sequentially or in parallel after jumping to vicinity called set. The block replacement algorithm needs to consider only few sets at a time. It can be economically implemented with limited number of choices as compared with fully associative.

978-1-4244-4520-2/09/$25.00 ©2009 IEEE

Here, alternate techniques of cache access have been verified for low power operation. These mechanisms are improvisations over the current set associative cache. The phased cache attempts to reduce the energy dissipation caused by accessing the data along with tag. We use separate data and tag arrays. This phased cache technique can be quite significant as the word lengths and data array sizes become bigger. The designs have been implemented using Verilog HDL and simulation results show a maximum of 50% decrease in power consumption.

In this paper the conventional cache architecture is explained at first and then we move on to the phased cache architecture. In section II the sequential and parallel architectures of the conventional cache is mentioned. In section III the sequential and parallel architectures of the phased cache architecture is mentioned. Section IV contains experimental results and section V describes additional related work.

II. CONVENTIONAL SET ASSOCIATIVE CACHE ARCHITECTURE

In conventional set associative cache the tag checking is done sequentially or in parallel and at the same time the data is fired to the bus for the bus master to access but the valid data bit is set only when a hit occurs. When the CPU generates an address, it first searches the cache for the data or instruction. If it is found in the cache then it is said to be a hit and if not found it is a miss. The address generated by the CPU is split into tag, set and word [7][8]. The corresponding set is selected with the generated set address and the tag of each line in the set is compared with the generated tag and at the same time the data will be available in the bus but the valid bit is not set. If there is a tag match then there is a hit and if none of the tag matches then there is a cache miss and the data or instruction should be fetched form the memory. If there is a hit the valid bit is set and the required data can be accessed from the bus.

A. Conventional Parallel Cache

In conventional parallel cache the tag comparison is done in parallel and hence it will take only one clock cycle to complete the comparison, but it needs a lot of hardware to perform this task. If the cache is four way set associative, we need four comparators each for a line in the set. This will

save clock cycles but will consume much more power by making all the comparators working at the same time and by accessing unwanted data and tags.

Figure 1a shows how the cache is accessed in a conventional parallel cache. The tags from all the ways in a set are taken simultaneously and are compared with the tag from CPU. If there is a tag match then the data that is already driven from the data subarray is sent to the data bus by enabling the buffer.

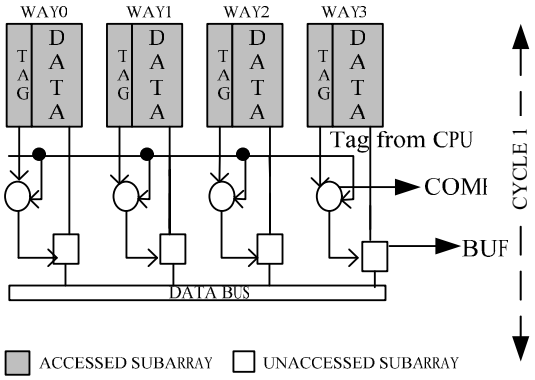

Fig1a: Conventional Parallel Cache

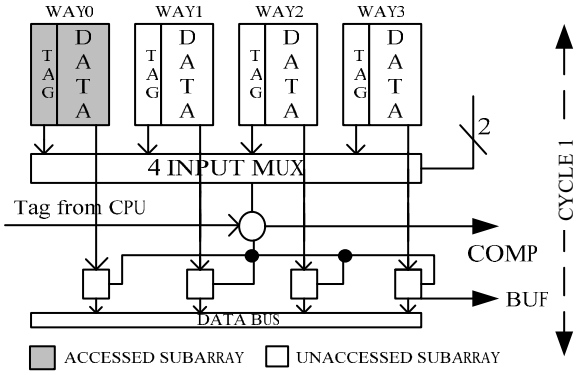

Fig1b: Conventional Sequential Cache

Each cache line is having a tag subarray and data subarray. The separation between the two subarrays is only logical. Each tag subarray is connected to a comparator and each data subarray is connected to a buffer. After set isolation when a tag is generated by the CPU, it is compared with the tags stored in all the tag subarrays in a set. If there is a tag match, the buffer is enabled and the requested data will be available in the data bus. All these operations are done in a single clock cycle. For simulation purpose we have taken a four way set associative cache with data subarray width of 8bits.

B. Conventional Sequential Cache

Consider the case when sequential method of access is employed. In this system after the set isolation, the tag of the first line is compared and the data line will be activated, i.e. it will be fed to the input of the buffer. If there is a hit the buffer is enabled and the data will be available at the output of the buffer. If it is a miss then the tag of the next way is compared and the data will be fed to the input of the buffer. This is repeated until there is a hit or when all the lines in a set are compared. This way the need for redundant hardware is eliminated, thus reducing the power consumed but will result in more number of clock cycles.

Here each way in a set is having a tag subarray and a data subarray. The separation between these two subarrays is only logical. Each tag is connected to the input of a mux and the selection line of the mux is a counter. In the first clock cycle the tag of the way0 is selected and compared with the tag from CPU, at the same time the data will be fed to the input of the buffer irrespective of a hit or miss. The enable of the buffer is the output of the comparator and hence it is enabled only if there is a tag match. The counter is incremented in each clock cycle, i.e. in each clock cycle different tags are compared. The worst case of this type of cache is when the requested data is in the last way. Here for simulation purpose we have taken a four way set associative cache with data subarray width of 8bits. So our cache will take four clock cycles in the worst case to find a hit if the data is stored in way3. The advantage of this cache organization is that if the data is stored in the way0 we can avoid accessing all other ways to find the required data.

Figure 1b shows a conventional sequential cache. The tag of the first way is selected and compared with the tag generated by the CPU, at the same time the data of the corresponding way will be fed to the input of the buffer and the buffer is enabled only if there is a hit, i.e. if there is a tag match. If there is a tag mismatch then the tag of the next way is selected and the process is repeated until a hit occurs or after checking all the ways in the set.

III. PHASED SET ASSOCIATIVE CACHE ARCHITECTURE

The energy consumption of the set-associative cache is much higher than the direct mapped cache, because the data and tag of the unwanted ways are also accessed. In phased cache the tags of all the ways are compared sequentially or in parallel in the first phase and if there is a tag match then the corresponding data is accessed and fed to the input of the buffer, which is the second phase. The corresponding buffer will be enabled in the same clock cycle and the data will be available in the data bus. If none of the tags in a set is matched then there is a cache miss and the data is fetched from the memory. The usage of phased cache will reduce the power consumption by avoid accessing the unwanted data subarrays. The disadvantage of phased cache is that it will use more number of clock cycles to access the data. Parallel phased cache gives more performance when compared to the sequential one but the hardware required for this is more and thus consuming more power.

A. Phased Sequential Cache

In phased sequential cache architecture there are physically separated tag subarray and data subarray. The tag subarray of way0 is accessed at first and then compared with the tag generated by the CPU. If the tag matches only then the data subarray is accessed. If there is no match, then the tag of the next way is accessed and compared. This is repeated until a hit occurs or after comparing all the ways in a particular set. For simulation purpose we have taken a four way set associative cache with data subarray width of 8bits.

Figure 2a shows a phased sequential cache. In this cache the tags of each way in a set is accessed sequentially and compared with the tag generated by the CPU. The tag-subarray is compared sequentially but the data sub-array is not even accessed. If there is a match then the corresponding data subarray is accessed in the next clock cycle and is fed in to the data bus. This will avoid unwanted access of data subarray and will save energy. If there is a mismatch then the next tag is accessed. Each tag in a set is connected to the input of a mux. The selection line of the mux is a counter. The counter is incremented in every clock cycle, so in the first clock cycle the tag of the way0 is compared with the tag generated by the CPU. If there is a tag match only then the corresponding data subarray is accessed, which is done in the next clock cycle. If there is no match then the counter is incremented and hence the tag of the next way will be selected and compared.

In the figure 2a the required data is in way0. The tag of the way0 is accessed and it is fed to the input of the mux. The counter is set to 00 in the beginning and so the tag of way0 is compared with the tag from the CPU. This is done in the first cycle, in the second cycle as there is a tag match the data subarray of way0 is accessed and fed to the input of the buffer. The buffer will be enabled in the same clock cycle and the required data will be available in the data bus in two clock cycles. This is the best case of phased sequential cache. The worst case is when the required data is stored in way3; in this case it will take five clock cycles to get the data in the data bus.

B. Phased Parallel Cache

In phased cache there are physically separated tag and data subarrays. The tag subarrays of all the ways are accessed simultaneously and compared with the tag from CPU. If there is a match then the data subarray of the matched way is accessed and is fed to the input of the buffer. The corresponding buffer is enabled and the requested data will be available in the data bus in two clock cycles provided it is not a miss. Here for simulation purpose we have taken a four way set associative cache with data subarray width of eight bits.

Figure 2b shows a phased parallel cache. In this cache the tags of all the ways in a set are compared simultaneously. If there is a hit then the corresponding data subarray is accessed and the data will be available in the data bus in the next clock cycle. This will avoid unwanted access of the data subarray. If there is a hit, then only the data is accessed. If none of the tags are matched with the tag generated by the CPU then there is a cache miss.

In the figure 2b it is clear that the number of comparators needed is equal to the number of ways in a set. Here we need four comparators as our cache is four way set associative. In the first clock cycle all the tags in a set are compared with the tag from the CPU. If there is a tag match then in the next clock cycle the data is accessed and will be available in the data bus.

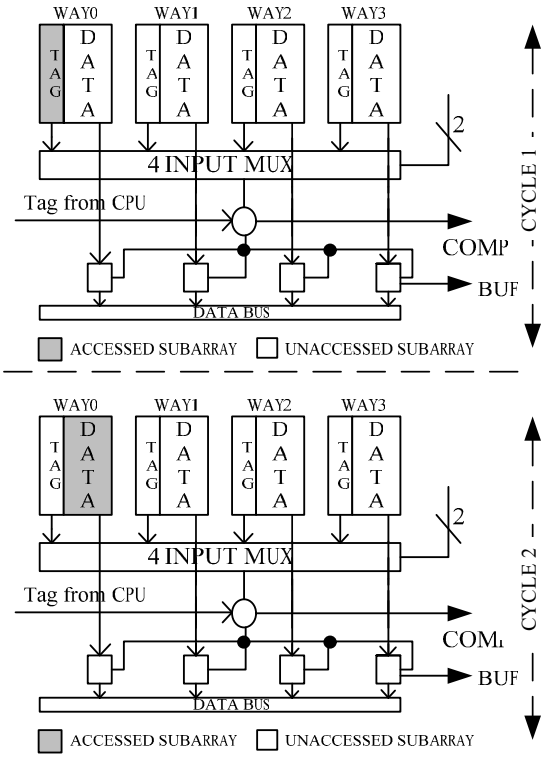

Fig2a: Phased Sequential Cache

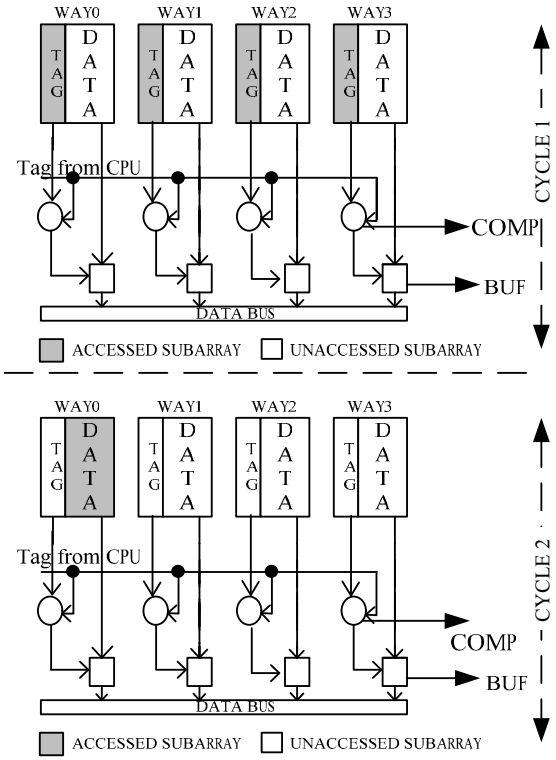

Fig2b: Phased Parallel Cache

IV. EXPERIMENTAL RESULTS

The conventional and phase cache architectures are checked using Xilinx Xpower analyzer. The coding of various cache architectures were done in Verilog HDL. The simulation results shows that the phased sequential cache can save up to 50% power when compared to the conventional sequential cache and phased parallel cache can save up to 23.2% power when compared to conventional parallel cache. This shows that this cache management technique can be employed in cases where the power is given much importance when compared to the performance. In modern computers also people are ready to sacrifice speed for saving power. This cache architecture can be employed in high frequencies, the frequencies at which modern CPUs work. It can be observed that the power reduction is higher at high frequencies.

Figure 3a shows the variation of power with respect to different clock frequencies for sequential cache architectures. From this we can observe that the power consumed by the conventional sequential cache is much higher than the phased sequential cache. At low clock frequencies there is no much difference in the power consumed by the conventional and phased cache. The reduction in power consumption is higher at higher frequencies. In the case of Phased sequential cache it is observed that there can be a reduction 50% of power at 300MHz. The reduction of power will increase further if we increase the frequency.

The conventional parallel, conventional sequential, phased parallel and phased sequential techniques are implemented using Verilog HDL. The results show that when sequential cache is replaced with the parallel cache, the maximum power gain is obtained. The difference in power consumption is much higher when the conventional sequential and phased sequential caches are compared. The graph containing the result is shown in figures 3a and 3b.

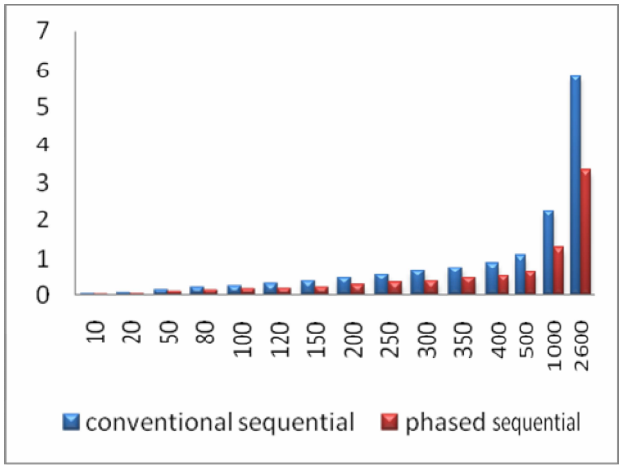

Fig 3a: Comparison of Conventional Sequential and Phased Sequential Caches

The conventional parallel cache consumes less number of clock cycles to deliver the requested data when compared to the conventional sequential cache. The conventional parallel cache will use more hardware to function and hence the power consumed will also be more. This is the same case with the phased parallel and phased sequential cache.

Phased parallel cache consumes more power than phased sequential cache as the requirement for hardware is more in parallel comparison of tags and accessing of data. The data obtained in the power analysis done in Xpower analyzer for four types of caches is given in tables 1 and 2. In that the final clock frequency for which power is measured is the operating frequency of the Intel core processor. Table 1 shows the power consumption and percentage reduction for conventional sequential and phased sequential caches. From the values obtained we can note that a 42.9% power reduction can be obtained if we replace the conventional cache with the phased sequential one. The reduction in power consumption will increase further if we include prediction logic also to the phased cache. The prediction logic is explained in the next section. As we can see from the table 1, if we replace the conventional sequential cache with the phased sequential one, we can get an average power reduction of around 40% and a maximum of 50% reduction at 300MHz.

Fig 3b which was plotted based on the values obtained in the table 2 shows the power consumption and percentage reduction in power consumption for conventional parallel

and phased parallel caches. Maximum power reduction obtained here is for the highest frequency. But we can note that in sequential caches the percentage reduction in power consumption is maximum at 300 MHz (50 % reduction), which is a moderate frequency of operation and is common in embedded systems applications i.e. Phased sequential cache is excellent for the embedded system applications. Phased parallel cache can be used for applications that need power reduction but without compromising the performance.

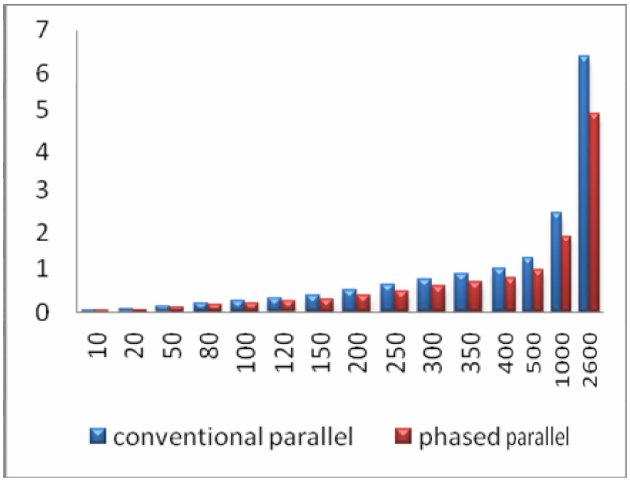

Fig 3b: Comparison of Conventional Parallel and Phased Parallel Caches

Clock Frequency (in MHz)	*Power Consumed (in W)*		% Reduction in Power Consumption
	Conventional Sequential	*Phased Sequential*	
10	0.036	*0.029*	19.4
20	0.058	*0.039*	32.7
50	0.124	*0.077*	37.9
80	0.19	*0.116*	38.9
100	0.235	*0.142*	39.5
120	0.279	*0.167*	40.1
150	0.346	*0.206*	40.5
200	0.457	*0.27*	40.9
250	0.567	*0.334*	41.1
300	0.678	*0.339*	50
350	0.789	*0.463*	41.3
400	0.9	*0.528*	413
500	1.121	*0.655*	41.5
1000	2.229	*1.292*	42
2600	5.835	*3.332*	42.9

1

Table 1: Power analysis done using Xilinx 10.

Clock Frequency (in MHz)	*Power Consumed (in W)*		% Reduction in Power Consumption
	Conventional Parallel	*Phased Parallel*	
10	0.035	*0.033*	5.7
20	0.063	*0.053*	15.8
50	0.136	*0.113*	16.9
80	0.209	*0.171*	18.18
100	0.259	*0.208*	19.69
120	0.307	*0.245*	20.19
150	0.381	*0.302*	20.73
200	0.503	*0.396*	21.27
250	0.626	*0.491*	21.56
300	0.748	*0.585*	21.79
350	0.871	*0.678*	22.15
400	0.993	*0.772*	22.25
500	1.239	*0.96*	22.51
1000	2.462	*1.899*	22.86
2600	6.386	*4.906*	23.21

Table 2: Power analysis done using Xilinx 10.1

V. WAY PREDICTING SET ASSOCIATIVE CACHE ARCHITECTURE

The way-predicting cache speculatively chooses one way before starting the normal cache-access process, and then accesses the predicted way[3] If the prediction is correct then cache access has been completed successfully and the data is fed to the buffer input and the buffer will be enabled in the same clock cycle and the data will be available in the data bus. On a prediction hit the way predicting cache consumes energy for activating the predicted way. If there is a hit the cache access can be completed in one cycle. If there is a prediction miss then the other ways in the set is accessed sequentially or in parallel. The performance or energy efficiency of a way predicting cache largely depends on the accuracy of the way prediction. In this paper we are suggesting the MRU (Most Recently Used) algorithm for the way prediction [1][2][5].The way predicting cache can be very suitable for embedded system applications.

In case of a 16 KB four-way set-associative cache, the MRU region is only 4 KB. The MRU information for each set, which is a two-bit flag, is used to speculatively choose one way from the corresponding set. These two-bit flags are stored in a table accessed by the set-index address. Reading the MRU information before starting the cache access might make cache access time longer. However, it can be hidden by calculating the set-index address at an earlier pipe-line stage [l]. In addition, way prediction helps reduce cache access-time due to eliminating of a delay for way selection.

VI. CONCLUSION

Here we have proposed and compared two techniques which will effectively reduce the logical hardware and thus save a significant amount of power. But this implementation has the disadvantage of slower execution and increase in

access time. This design may be implemented especially in embedded systems where speed is not a major concern but reduction in power consumption is appreciated.

ACKNOWLEDGMENT

We gratefully acknowledge the Almighty GOD who gave us strength and health to successfully complete this venture. The authors wish to thank Amrita Vishwa Vidyapeetham, in particular the Digital library, for access to their research facilities.

REFERENCES

[1] B.Calder, D.Grunwald, and J.Emer, "Predictive Sequential Associative Cache," Proceedings of Second International Symposium on High-Performance Computer Architecture, Feb. 1996, pp.244 -253.

[2] J. H. Chang, H. Chao, and K. So., "Cache design of a sub-micron cmos system/370," in 14th Annual International Symposium on Computer Architecture, SIGARCH Newsletter, pp. 208–213, June 1987.

[3] Koji I, Tohru I and Kazuaki M, "Way Predicting Set Associative Cache for High Performance and Low Energy Consumption," ISLPED99, San Diego, CA, USA.

[4] A. Hasegawa et al., "SH3: High Code Density, Low Power," IEEE Micro, Vol. 15, No. 6, Dec. 1995, pp. 11-19.

[5] Kessler R, Jooss R, Lebeck A, Hill M., "Inexpensive implementation of set-associativity," Proc. of the 16th annual international symposium on Computer Architecture, pp. 131-139,1989.

[6] J. Kin, M. Gupta, and W.H. Mangione-Smith, "The Filter Cache: An Energy Efficient Memory Structure," *IEEE/ACM International Symposium on Microarchitecture* (MICRO-30),pages 184–193, 1997.

[7] M. Morris Mano, "Computer System Architecture 3rd edition,"Prentice Hall, Inc. USA, 1993.

[8] William Stallings, "Computer Organization and Architecture", 4th edition.