Advanced Research in Electrical and Electronic Engineering
Print ISSN: 2349-5804; Online ISSN: 2349-5812 Volume 2, Number 6 April-June (2015) pp. 21-25

http://www.krishisanskriti.org/areee.html

Comparison of Various Energy Efficient L2 Cache Implementing Way-Tagging using different Datum Storage Technique

Anurag Gupta[1], Himanshu Sharma[2] and Preeti Shakya[3]

[1]PG Research Scholar SRGI, JHANSI
[2,3]SRI, Datia
E-mail: [1]anurag.jalaun18@rediffmail.com, [2]himanshusharma1003@gmail.com, [3]shakyapreeti@gmail.com

Abstract—*Cache memories employ two main policies namely write-through and write back. The write- through policy is much more energy efficient than write-back. However write-through policy also incurs large energy overhead. Considering the energy efficiency comparison of the various way-tagged cache architectures is made here which employs D-FF, ETL, GDI technique as well as D-latch as the datum storage elements.*

1. CACHE MEMORY INTRODUCTION

In computer science, Cache is small high speed memory usually Static RAM (SRAM) that contains the most recently accessed data of main memory. Now the question arises, why is this high speed memory necessary or beneficial? The basic purpose of cache memory is to store program instructions that are repeatedly used by software during operation. Fast access to these instructions increases the overall speed of the software program[1]. As the microprocessor processes data, it first refers the cache memory; if it finds the instructions within the cache, it does not have to do a more time-consuming reading of data from larger memory i.e main memory or other data storage devices. An entry can be found with a tag matching that of the desired datum, the datum in the entry is used instead. This situation is known as a cache hit. So, for example, a web browser program might check its local cache on disk to see if it has a local copy of the contents of a web page at a particular URL. In this example, the URL is the tag. During a cache miss, the Central Processing Unit (CPU) usually ejects some other entry in order to make room for the previously un-cached datum. The heuristic used to select the entry to eject is known is the replacement policy. One popular replacement policy, "least recently used", replaces the least recently used entry. More efficient caches compute use frequency against the size of the stored contents, as well as the latencies and throughputs for both the cache and the backing store. This works well for larger amounts of data, longer latencies and slower throughputs, such as experienced with a hard drive and the Internet, but is not efficient for use with a CPU cache[6]. New cache architecture, referred to as way-tagged cache, can be implemented so as to improve the energy efficiency of write-through cache systems with minimal area overhead and no performance degradation. Consider a two-level cache hierarchy, where the L1 data cache is write-through and the L2 cache is inclusive for high performance. It is observed that all the data residing in the L1 cache will have copies in the L2 cache. In addition, the locations of these copies in the L2 cache will not change until they are expelled from the L2 cache. Thus, it is possible to attach a tag to each way in the L2 cache and send this tag information to the L1 cache when the data is loaded to the L1 cache. By doing so, for all the data in the L1 cache, it is possible to know exactly the locations of their copies in the L2 cache. During the subsequent accesses when there is a write hit in the L1 cache, the L2 cache can be accessed in an equivalent direct-mapping manner because the way tag of the data copy in the L2 cache is available. As this operation accounts for the majority of L2 cache accesses in most applications, the energy consumption of L2 cache can be reduced significantly. In Section 2, a glance has been put at the architecture. In Section 3, detailed VLSI architecture of the way-tagged cache is presented. In Section 4 effectiveness of the proposed technique is presented. Simulation results are given in Section 5.

2. ARCHITECTURE

A way-tagged cache that exploits the way information in L2 cache to improve energy efficiency is an important technique in association with data consistency and cache energy consideration. A conventional set-associative cache system is considered, when the L1 data cache loads/writes data from/into the L2 cache, all ways in the L2 cache are activated simultaneously for performance consideration at the cost of energy overhead. Fig. 1 and 2 illustrates the architecture of the two-level cache[5]. Under the write-through policy, the L2 cache always maintains the most recent copy of the data. Thus, whenever a data is updated in the L1 cache, the L2 cache is updated with the same data as well. This results in an

increase in the write accesses to the L2 cache and thus more energy consumption.

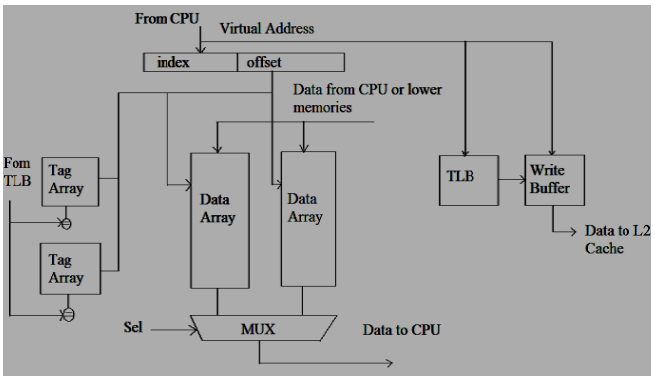

Fig. 1: Illustration of the conventional L1 cache architecture.

So, reducing the energy consumption of L2 write accesses is an effective way as far as memory power management is considered. The locations of L1 data copies in the L2 cache will not change until the data are expelled from the L2 cache. The proposed way-tagged cache takes advantage of this fact to reduce the number of ways accessed during L2 cache accesses. When the L1 data cache loads a data from the L2 cache, the way tag of the data in the L2 cache is also sent to the L1 cache and stored in a new set of arrays referred to as way-tag arrays. These way tags provide the key information for the subsequent write accesses to the L2 cache. In general, both write and read accesses in the L1 cache may need to access the L2 cache. These accesses lead to different operations in the proposed way-tagged cache. Under the write-through policy, all write operations of the L1 cache need to access the L2 cache. In the case of a write hit in the L1 cache, only one way in the L2 cache will be activated because the way tag information of the L2 cache is available, i.e., from the way-tag arrays we can obtain the L2 way of the accessed data.

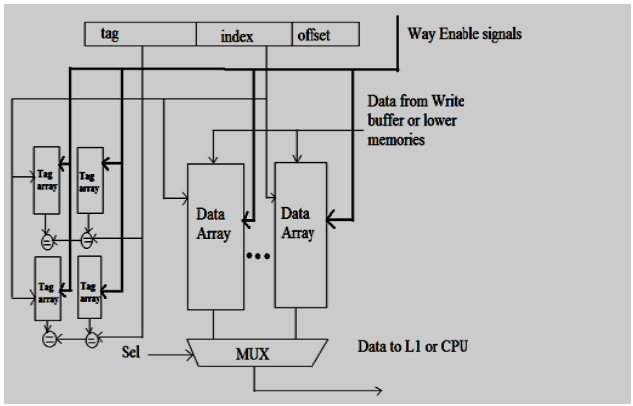

Fig. 2: Illustration of the conventional L2 cache architecture.

While for a write miss in the L1 cache, the requested data is not stored in the L1 cache. As a result, its corresponding L2 way information is not available in the way-tag arrays. Therefore, all ways in the L2 cache need to be activated simultaneously. Since write hit/miss is not known a priori, the way-tag arrays need to be accessed simultaneously with all L1 write operations in order to avoid performance degradation.

3. IMPLEMENTATION OF THE WAY-TAGGED CACHE

3.1 Way tag arrays

In the way-tagged cache, each cache line in the L1 cache keeps its L2 way tag information in the corresponding entry of the array. When a data is loaded from the L2 cache to the L1 cache, the way tag of this data is also written into the array[3]. At a later time when updating this data in the L1 data cache, the respective copy in the L2 cache also needs to be updated as well. This is based over the write-through policy. The way tag stored in the way-tag array is read and passed to the way-tag buffer together with the data from the L1 data cache. The write/read signal of way-tag arrays is generated from the write/read signal of the data arrays in the L1 data cache. A control signal UPDATE is obtained from the cache controller. When the write access to the L2 data cache is caused by a L1 cache miss, UPDATE will be asserted and allows enabling the write operation to the way-tag arrays. If a STORE instruction accesses the L1 data cache, UPDATE keeps invalid and write signal indicates a read operation to the way-tag arrays. During the read operations of the L1 cache, the way-tag arrays do not need to be accessed and thus are deactivated to reduce energy overhead.

3.2 Way-Tag Buffer

Way-tag buffer is basically for temporary storage of the way tags read from the arrays. The implementation of the way-tag buffer is shown in Fig. 3. It has the same number of entries as the write buffer of the L2 cache and shares the control signals with the L2 cache. Each entry of the way tag Buffer has (n+1) bits, where n is the line size of array of the way tag. An additional status bit indicates whether the operation in the current entry is a write miss or not on the L1 data cache. In case of a write miss, all the ways in the L2 cache need to be activated since the way information is not available. In the opposite case, only the desired way is activated[3].

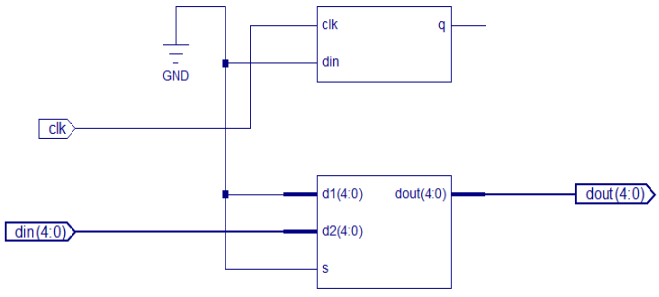

Fig. 3: Way-Tag Buffer

The status bit is updated in accordance with the read operations of way-tag arrays at the same clock cycle. Similar to the write buffer of the L2 cache, the way-tag buffer has separate write and read logic which helps in supporting parallel write as well as read operations. The write operations in the way-tag buffer occur one clock cycle far ahead of the respective write operations in the write buffer.

3.3 Way-Decoder

Way decoder is usually employed to decode way tags and activate only the looked-for ways in the L2 cache as depicted in Fig. 4. Here the binary codes are utilized; thus the line size of way-tag arrays is bits. This curtails the energy overhead from the additional wires and the impact on chip area can be neglected. For a L2 write access instigated by a write hit in the L1 cache, the way decoder runs as a -to- decoder that selects just one way-enable signal. The way-decoder operates concurrently with the decoders of the tag as well as data arrays in the L2 cache. For a write miss or a read miss in the L1 cache, it is required to affirm all way-enable signals so that all L2 cache ways are initiated. There are two signals read as well as write miss, which govern the functioning mode of the way decoder. The read signal will be high in case a read access is directed to the L2 cache. Signal write miss will be low if the write operation accessing the L2 cache is triggered by a write miss in the respective L1 cache.

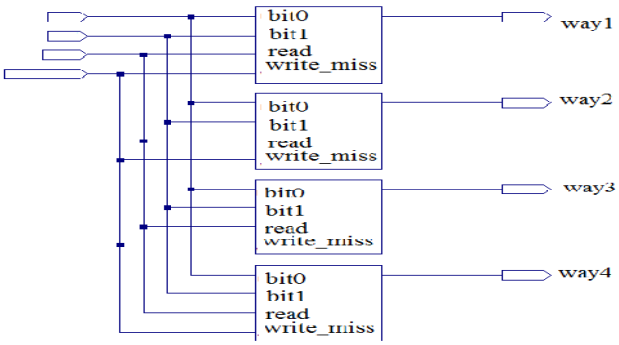

Fig. 4: Way Decoder

3.4 Way Register

The way register is the resource for providing way tags for the respective way-tag array. For a 4-way L2 cache, the markers ―00‖, ―01‖, ―10‖, and ―11‖ are deposited in the relevant way register. When the L1 cache loads a data as of from the L2 data cache, the respective way tag in the way register is sent to the arrays.

4. DATA STORAGE TECHNIQUE

(i) D FF using gates

To design a D latch we use universal NAND and NOR gates. The D latch is simple gated S R latch with a NAND inverter connected between its S and R inputs. Due to inverter S and R is always be the complement of each other. Hence S= R =0 or S = R= 1, these input condition will never appear. And this will

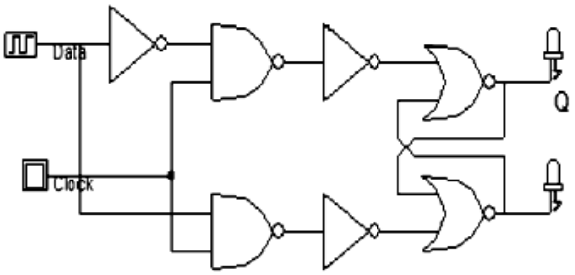

Fig. 5: D-latch using gate

avoid the problems associated with SR= 00 and SR = 11 conditions. To design a D latch we saw a working first, with Dsch software that when the clock is low, output is not affected. When the clock is high, the Q output is equals to input D and the not Q output is the inverse of Q. The previous implementation needs 22 transistors. In order to optimize our design, we decide to use complex gates, which only need 14 transistors. Moreover, we have better propagation delay with its implementation. This is CMOS D Latch design by using one CMOS inverter and two complex gates. The one input is D and other is clock, and the output is Q and not Q. The Q output is same as D input. The not Q is just opposite to output Q.

(ii)DFF using GDI

An enactment of efficient D-Flip-Flop (DFF) using Gate-Diffusion-Input (GDI) method is depicted in Fig. 6[11]. This DFF design allows reduction of power-delay product and as such the area of the circuit, while maintaining low complication in logic design. It is based on the Master-Slave linking of two individual GDI D-Latches. Each latch consists of four basic GDI cells, causing in an unsophisticated eight-transistor model. The sections of the circuit can be classified as:

- Body gates
- Inverters

Body gates are accountable for the state-owned by the circuit[11]. These gates are measured by the clock signal and create two alternative paths: one for transparent state of the latch when the clock is low and the signals are propagating through P-channel Metal Oxide Semiconductor (PMOS) transistors, and another for the holding state of the latch when the clock is high and internal values are maintained due to conduction of the N-channel Metal Oxide semiconductor (NMOS) transistors. While the inverters responsible for maintaining the complementary values of the internal signals and the circuit outputs. An additional important role of

inverters is buffering of the internal signals for swing restoration and improved driving abilities of the outputs.

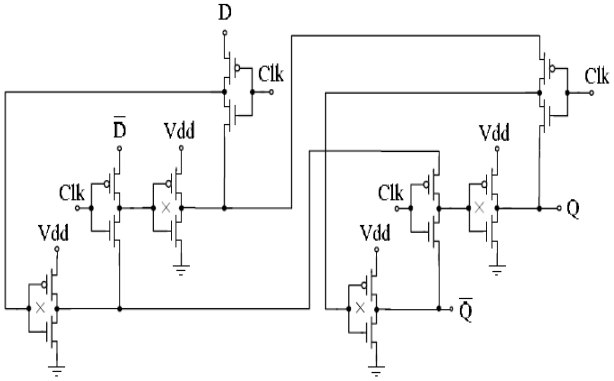

Fig. 6: GDI D-Flip flop implementation

(iii)ETL DFF

As the feature size of CMOS technology process shrinks according to Moore's Law, designers are able to integrate more transistors onto the same die. The more transistors there are the more switching and the more power dissipated in the form of heat or radiation. Heat is one of the most important packaging challenges in this era; it is one of the main drivers of low power design methodologies and practices. Another mover of low power research is the reliability of the integrated circuit. More switching implies higher average current is flowing and therefore the probability of reliability issues occurring rises. The most important prime mover of low power research and design is our convergence to a mobile society. We are moving from desktops to laptops to handhelds and smaller computing systems. With this profound trend continuing, and without a matching trend in battery life expectancy, the more low power issues will have to be addressed. This entails that low power tools and methodologies have to be developed and adhered to. The current trends will eventually mandate low power design automation on a very large scale to match the trends of power consumption of today's integrated chips.as shown in Fig. 7.

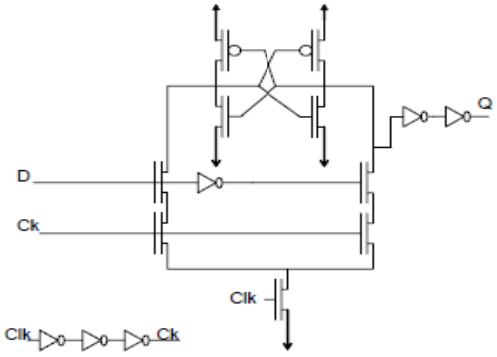

Fig. 7: ETL D FLIP FLOP

5. SIMULATION RESULTS

With the above said components, the proposed way-tagged as depicted in Fig. 8, cache operates under different modes of read and write operations. Just the way involving the desired data is activated in the L2 cache for a write hit in the L1 cache, making the L2 cache equivalently a direct-mapping cache to reduce energy consumption without introducing performance overhead. Multiplexer is employed to generate the enable signal for the tag arrays of the L2 cache.

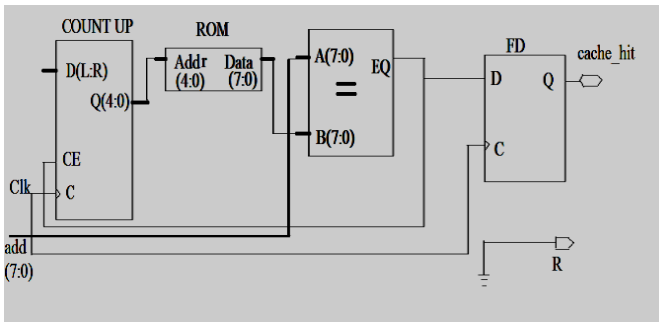

Fig. 8: Complete L2 cache depiction

When the status bit indicates a write hit M1 outputs low to disable all the ways in the tag arrays. Multiplexer chooses the output from the way decoder as the selection signal for the data arrays. If on the other hand the access is caused by a write miss or a read miss from the L1 cache, all ways are enabled by the tag array decoder, and the result of tag comparison is selected by M2 as the selection signal for the data arrays. Overall, fewer ways in the tag arrays are activated, thereby reducing the energy consumption. Finally with the help of the energy efficient technique the overall energy consumption of the architecture is maintained and reduced as compared to the earlier cache architectures.

Power Consumption

A variety of circuits have been implemented in 0.35micrometre technology to compare the proposed D-Latch structure with Other Technique reduction in power as illustrated.

Table I: Power Analysis

Technique Used	Parameter-Power Consumption (in mw)
Earlier cache architecture (DFF)	176
ETL	63
GDI	46
D Latch	25

B. Power Analysis of entire Architecture

By the implementation of the Way-Tagged Architecture in conjunction with the efficient D-Latch technique the overall power consumption undergoes a reduction of about 86.03%

with less performance degradation as such. This reduction is shown in Fig. 9. as such.

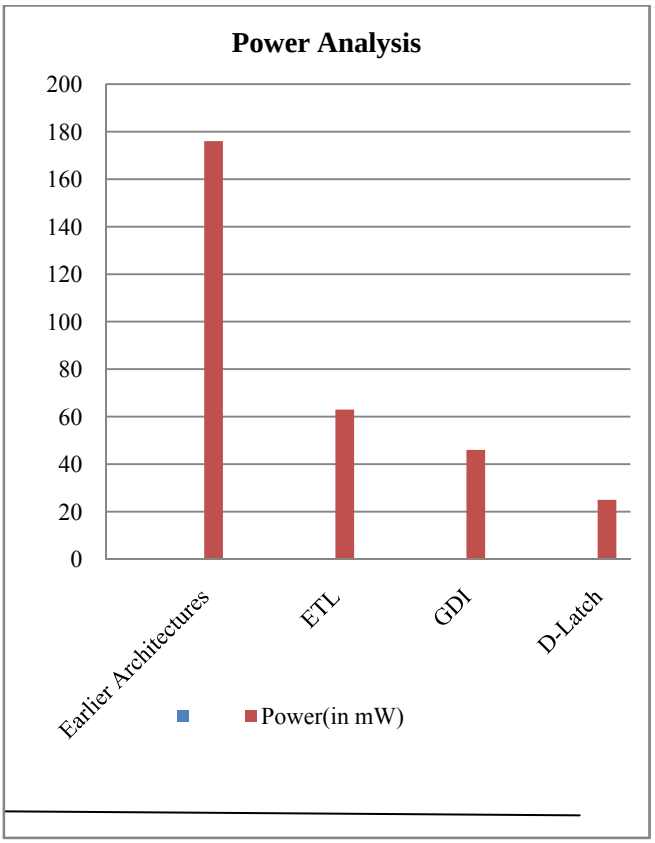

Fig. 9: Power comparisons

6. CONCLUSION

The Way-Tagged L2 Cache architecture presents a new energy-efficient cache technique for high-performance microprocessors employing the write-through policy. The proposed technique attaches a tag to each way in the L2 cache. This way tag is sent to the way-tag arrays in the L1 cache when the data is loaded from the L2 cache to the L1 cache. Utilizing the way tags stored in the way-tag arrays, the L2 cache can be accessed as a direct-mapping cache during the subsequent write hits, thereby reducing cache energy consumption. Simulation results demonstrate significantly reduction in cache energy consumption with minimal area overhead and no performance degradation.

REFERENCE

[1] ArkadiyMorgenshtein, Alexander Fish and Israel A.Wagner, ―An efficient implementation of D-Flip-Flop using the GDI technique.

[2] V.Stojanovic and V.G.Oklobdzija, "Comparative Analysis of Master–Slave Latches and Flip-Flops for High-Performance and Low-Power Systems", IEEE J. Solid-State Circuits, vol. 34, no. 4, April 1999.

[3] K. Inoue, T. Ishihara, and K. Murakami, ―Way-predicting set-associative cache for high performance and low energy consumption,‖ in Proc. Int. Symp. Low Power Electron. Design, 1999, pp. 273–275.

[4] J.M. Rabaey, A. Chandrakasan, B. Nikolic, ―Digital Integrated Circuits‖, 2nd edition, Prentice Hall, 2002.

[5] C. Zhang, F. Vahid, and W. Najjar, ―A highly-configurable cache architecture for embedded systems,‖ in Proc. Int. Symp. Comput. Arch., 2003, pp. 136–146.

[6] R. Min, W. Jone, and Y. Hu, ―Location cache: A low-power L2 cache system,‖ in Proc. Int. Symp. Low Power Electron. Design, 2004, pp. 120–125.

[7] T. Ishihara and F. Fallah, ―A way memoization technique for reducing power consumption of caches in application specific integrated processors,‖ in Proc. Design Autom. Test Euro. Conf., 2005, pp. 358–363.

[8] T. N.Vijaykumar,-Reactive-associative caches,‖ in Proc. Int. Conf. Parallel Arch. Compiler Tech., 2011.

[9] Simmy Hirkaney, Sandip Nemade, Vikash Guptapower efficient design of counter on .12 micron technology -2011.

[10] Way-Tagged L2 Cache Architecture in Conjunction with Energy Efficient Datum Storage VineetaVasudevan Nair ECE Department, ANNA University Chennai Sri. Eshwar College Of Engineering Coimbatore, India-2014