ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

Power Reduction in L2 Cache System Using Way-Tag Information Under Write Through Policy

A.Sudhakar
PG Scholar (MTech)
Department of ECE
Vaagdevi Engineering College
Bollikunta, Warangal, India.

Mrs.M.Swetha
Assistant Professor
Department of ECE
Vaagdevi Engineering College
Bollikunta, Warangal, India.

***Abstract*:**
In this paper we are going to increase the performance of the processor by a new policy called write-through and a new cache architecture referred to as way-tagged. In this way-tagged process we are having L1 cache and L2 cache and the address at which the data have to be stored is divided into three parts tag, index and offset address and the data which is going to be stored in the L1 & L2 caches are stored with reference with the tag address and the copy of the tag address is stored in the way-tag array. Way-tag array is an array where the way-tag address of the data is stored. When the processor required the data to perform the required operations first it check the L1 cache and if the data is present in the L1 cache it fetches the data otherwise it check the L2 cache for the data and similarly if the data is not present in L2 cache the processor checks the data in the main memory .while processor fetching the data from the main memory it stores the data in the L2 & L1 cache respectively and stores the way-tag address in the respectively L1 and L2 way-tag arrays. Simulation results on the Model-Sims and synthesis results on Xilinx demonstrate that the proposed technique achieves power savings over conventional L2cache on average with small area overhead and no performance degradation. Furthermore, the idea of way tagging can be applied to existing low-power cache design techniques to further improve energy efficiency.

***Index Terms*— Cache, power, write-through policy.**

1.Introduction

A general-purpose processor is a finite-state automaton that executes instructions held in a memory. The state of the system is defined by the values held in the memory locations together with the values held in certain registers within the processor itself (see Fig.1). Each instruction defines a particular way the total state should change and it also defines which instruction should be executed next. If we want to make a processor go fast, we must first understand what it spends its time doing. It is a common misconception that computers spend their time computing, that is, carrying out arithmetic operations on user data. In practice they spend very little time 'computing' in this sense. Although they do a fair amount of arithmetic, most of this is with addresses in order to locate the relevant data items and program routines. Then, having found the user's data, most of the work is in moving it around rather than processing it in any transformational sense.

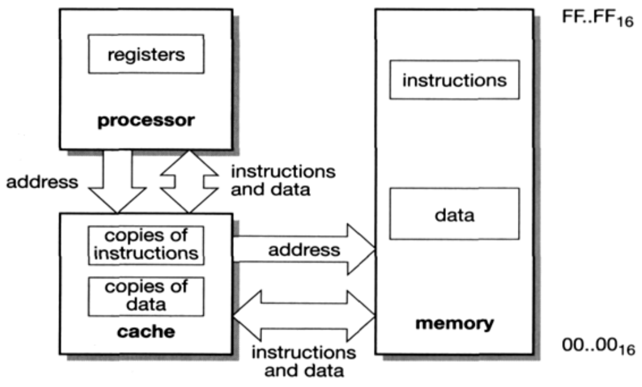


Figure.1: A unified instruction and data cache

In this paper we are concerned mainly about the Data movement by keeping the data very near to the processor by using the L1 & L2 Cache and by employing Write-through and Write-back polices. Under the write-back policy, a modified cache block is copied back to its corresponding lower level cache only when the block is about to be replaced. While under the write-through policy, all copies of a cache block are updated immediately after the cache block is modified at the current cache, even though the block might not be evicted. As a result, the write-through policy maintains identical data copies at all levels of the cache hierarchy throughout most of their life time of execution.

It has been reported that single-event multi bit upsets are getting worse in on-chip memories. Currently, this problem has been addressed at different levels of the design abstraction. At the architecture level, an effective solution is to keep data consistent among different levels of the memory hierarchy to prevent the system from collapse due to soft errors. Benefited from immediate update, cache write-through policy is inherently tolerant to soft errors because the data at all related levels of the cache hierarchy are always kept consistent. Due to this feature, many high-performance microprocessor designs have adopted the write-through policy.

Write strategies

Write-through policy:

All write operations are passed to main memory; if the addressed location is currently held in the cache, the cache is updated to hold the new value. The processor must slow down to main memory speed while the write takes place.

Write-through with buffered write:

Here all write operations are still passed to main memory and the cache updatedas appropriate, but instead of slowing the processor down to main memory speedthe write address and data are stored in a write buffer which can accept the writeinformation at high speed. The write buffer then transfers the data to main memory,at main memory speed, while the processor continues with its next task.

Copy-back (also known as write-back):

A copy-back cache is not kept coherent with main memory. Write operations update only the cache, so cache lines must remember when they have been modified (usually using a dirty bit on each line or block). If a dirty cache line is allocated to new data it must be copied back to memory before the line is reused.

The write-through cache is the simplest to implement and has the merit that the memory is kept up to date; the drawback is that the processor must slow to memory speeds on every write transfer. The addition of a write buffer allows the processor to continue until the write traffic exceeds the external write bandwidth. The copy-back cache reduces the external write bandwidth requirement since a location may be written many times before the final value gets written back to memory, but the implementation is considerably more complex and the loss of coherency is hard to manage.

2.Conventional Two Level Cache

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 consequently more energy consumption.

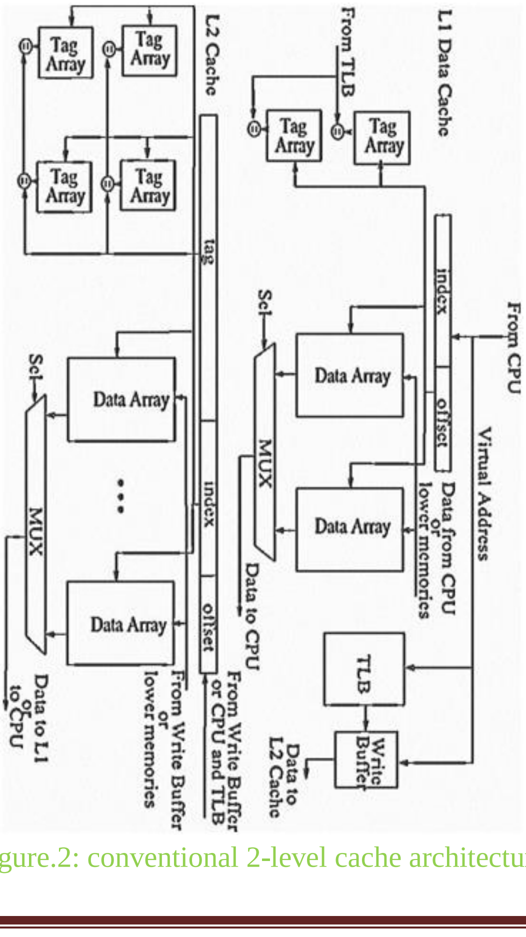


Figure.2: conventional 2-level cache architecture

	Read hit			Write hit	
L1 cache	**YES**	**NO**		**Yes If same data is not present**	**No**
				Need to update	**Need to copy**
L2 cache	**No Need to check**	**Need to check Set-associative**		**Need to update (direct mapping)**	**Need to copy (set-associative)**
		YES	**NO**		
Main memory	**No Need to check**	**No need to check**	**Need to check**	**Need to update**	**Need to copy**

Table.1: Different Operations in L1,L2 and Main memory

For read process initially the processor check the L1 cache memory for the data, if the required data is present read hit signal is set to yes other wise No. If read hit signal in L1 is Yes then no-need to check the L2 and Main memory. If it is No it has to check L2 cache in that if the required data is found in L2 cache then the read hit signal is set to YES otherwise No. If the data found in the L2 Cache no need to go for Main memory otherwise it has to checks the main memory.

For write process the processor initially checks the address location in the L1 cache if the address location is found then it compare the data. If the same data is present then no need write otherwise it has to update. Same update operation is carried out in L2 and main Memory. If the required address is not Found in the L1 cache then it has to copy the data in the L1, L2 and Main Memory.

3. Way-Tagged Cache

Fig.3 shows the system diagram of proposed way-tagged cache. We introduce several new components: way-tag arrays, way-tag buffer, way decoder, and way register, all shown in the dotted line. The way tags of each cache line in the L2 cache are maintained in the way-tag arrays, located with the L1 data cache. Note that write buffers are commonly employed in write-through caches (and even in many write-back caches) to improve the performance. With a write buffer, the data to be written into the L1 cache is also sent to the write buffer.

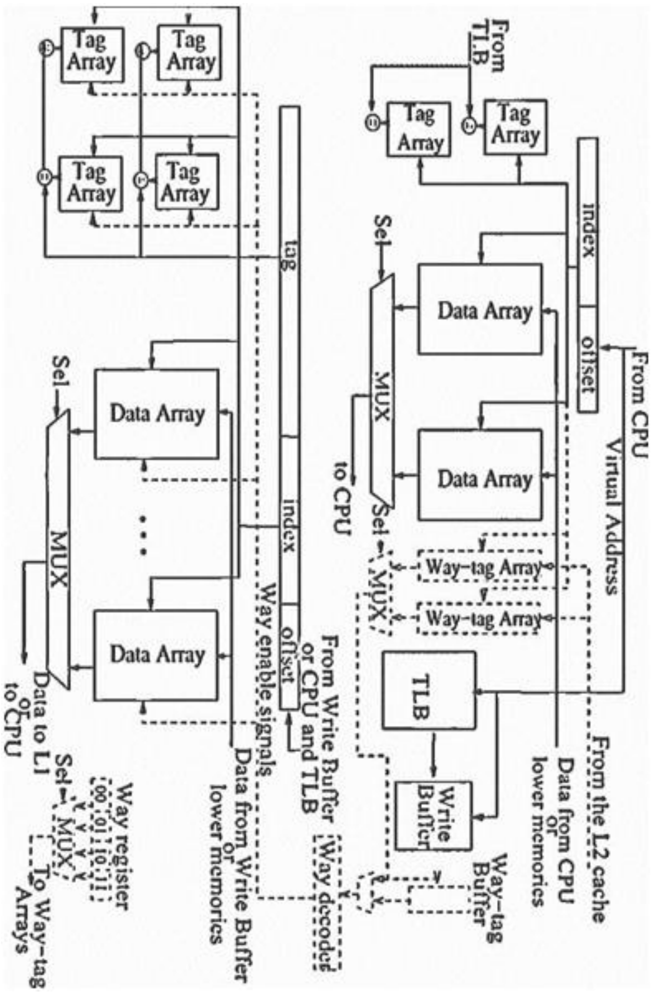


Figure.3: Proposed way-tag cache

The operations stored in the write buffer are then sent to the L2 cache in sequence. This avoids write stalls when the processor waits for write operations to be completed in the L2 cache. In the pro- posed technique, we also need to send the way tags stored in the way-tag arrays to the L2 cache along with the operations in the write buffer. Thus, a small way-tag buffer is introduced to buffer the way tags read from the way-tag arrays. A way decoder is employed to decode way tags and generate the enable signals for the L2 cache, which activate only the desired ways in the L2 cache. Each way in the L2 cache is encoded into a way tag. A way register stores way tags and provides this information to the way-tag arrays.

4. Implementation of Way-Tag Cache

A. Way-Tag Array: Way-tag array stores the tag information of the data in different location depending on the starting address location. Whenever the data is needed by the processor, the processor check address in the way-tag and if the tag address is found then the data is fetched from the corresponding address location. If not then there is no need in check the memory location.

When a data is loaded from the L2 cache to the L1 cache, the way tag of the data is written into the way-tag array. At a later time when updating this data in the L1 data cache, the corresponding copy in the L2 cache needs to be updated as well under the write-through policy. The way tag stored in the way-tag array is read out and forwarded to the way-tag buffer together with the data from the L1 data cache.

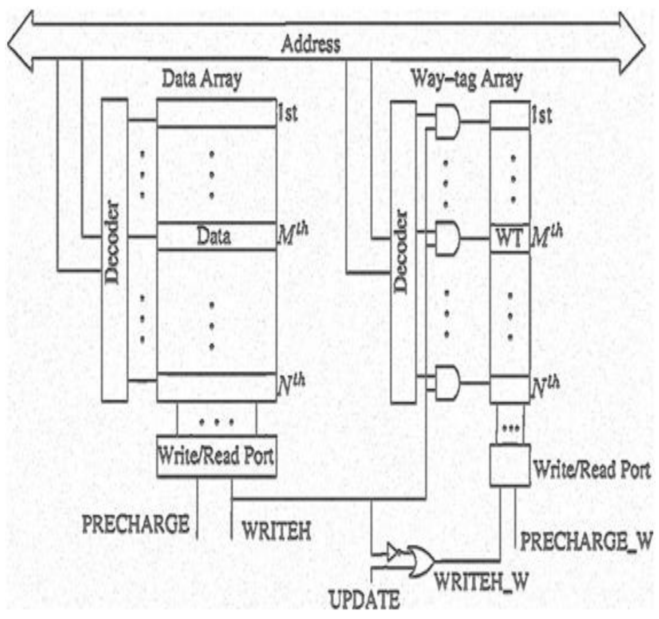


Figure.4: way-tag array

WRITEH	UPDATE	OPERATION
1	1	Write way tag arrays
1	0	Read way tag arrays
0	0	No access
0	1	No access

Table 2: Operations of way-tag array

B. Way-Tag buffer: Way-Tag buffer is used to mirroring the information from one place to another place. Way-tag buffer temporarily stores the way tags read from the way-tag arrays. Implementation is shown in below fig.5. It has the same number of entries as the write buffer of the L2 cache and shares the control signals with it. Each entry of the way-tag buffer has n+1 bits, where n is the line size of way-tag arrays. An additional status bit indicates whether the operation in the current entry is a write miss on the L1 data cache. When a write miss occurs, all the ways in the L2 cache need to be activated as the way information is not available. Otherwise, only the desired way is activated. The status bit is updated with the read operations of way-tag arrays at the same clock cycle.

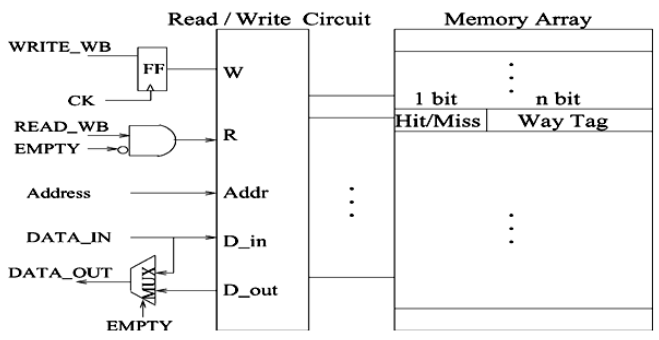


Figure 5: Way-tag buffer

C.Way-Decoder: Way decoders are used to select the way and activate only the desired ways in the L2 cache. Below fig. 6 shows the block diagram of the Way-Decoder. It operates simultaneously with the decoders of the tag and data arrays in the L2 cache. For a write miss or a read miss in the L1cache, we need to assert all way-enable signals so that all ways in the L2

ISSN No: 2348-4845
International Conference on Electronics, Communications and VLSI Circuits (ICECV-2015)
July 10, 2015 - Hyderabad, India
Papers Published in IJMETMR, A Peer Reviewed Open Access International Journal.

cache are activated. Two signals *read* and *write miss*, determine the operation mode of the way decoder. Signal *read* will be "1" when a read access is sent to the L2 cache. Signal *write miss* will be "1" if the write operation accessing the L2 cache is caused by a write miss in the L1 cache.

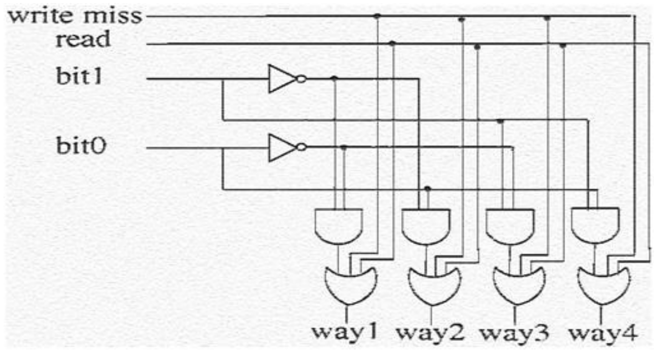


Figure 6: way decoder

D. Way Register: The way register are used to provide way tags for the way-tag arrays. When the data is carried from main memory to L2 cache and also form L2 to L1 cache i.e., when the data is carried the way-tag address is carried by way-register.

5. Simulation Results

A. simulation results way-tag cache

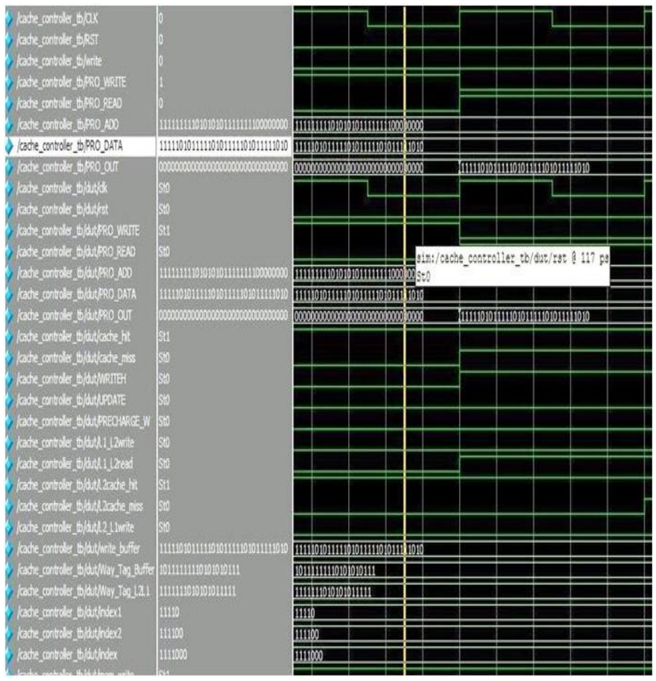


Figure 7: simulation result of way-tag cache top module (capture-1)

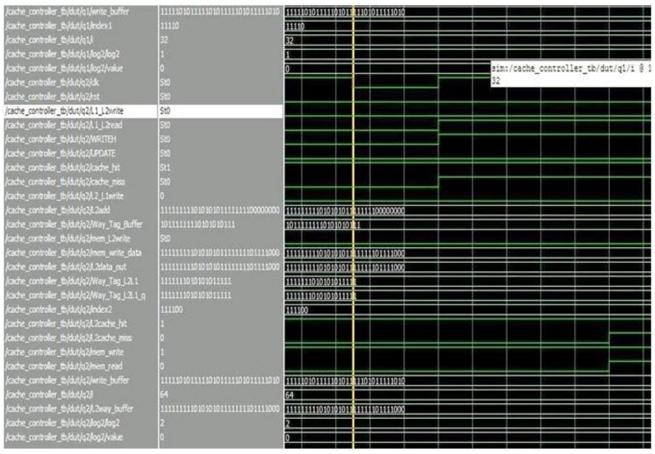


Figure 8: Simulation result of way tag cache top module(capture-2)

B. Simulation result of conventional two-level cache

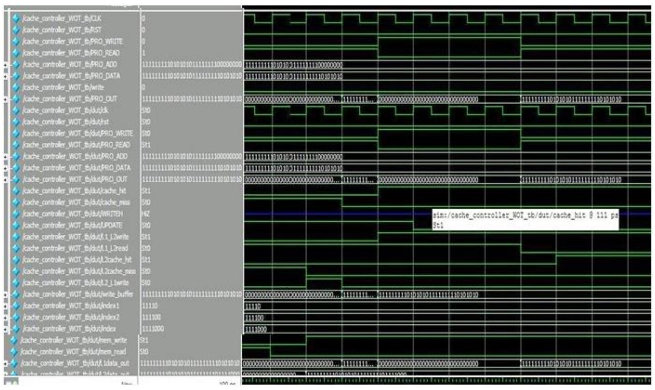
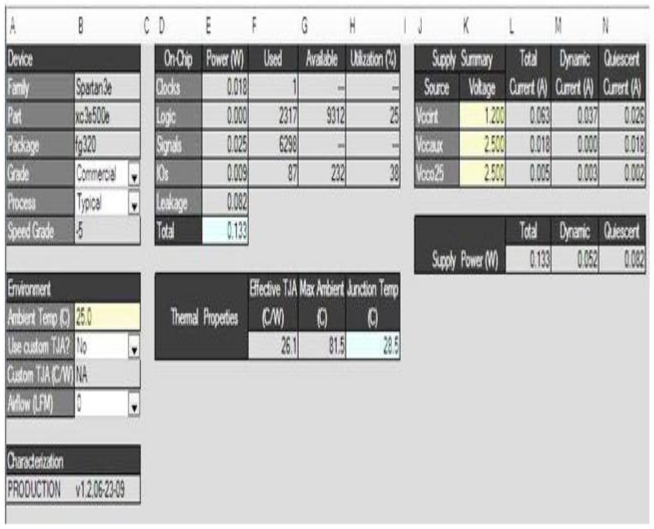


Figure 9: Simulation result of conventional 2-Level cache top module

6. Synthesis Report

A. Synthesis report without way-tag array



Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Grade	Commercial
Process	Typical
Speed Grade	-5

On-Chip	Power (W)	Used	Available	Utilization (%)
Clocks	0.018	1	--	--
Logic	0.000	2317	9312	25
Signals	0.025	6298	--	--
IOs	0.009	87	232	38
Leakage	0.082			
Total	0.133			

Supply Summary		Total	Dynamic	Quiescent
Source	Voltage	Current (A)	Current (A)	Current (A)
Vccint	1.200	0.063	0.037	0.026
Vccaux	2.500	0.018	0.000	0.018
Vcco25	2.500	0.005	0.003	0.002

	Total	Dynamic	Quiescent
Supply Power (W)	0.133	0.052	0.082

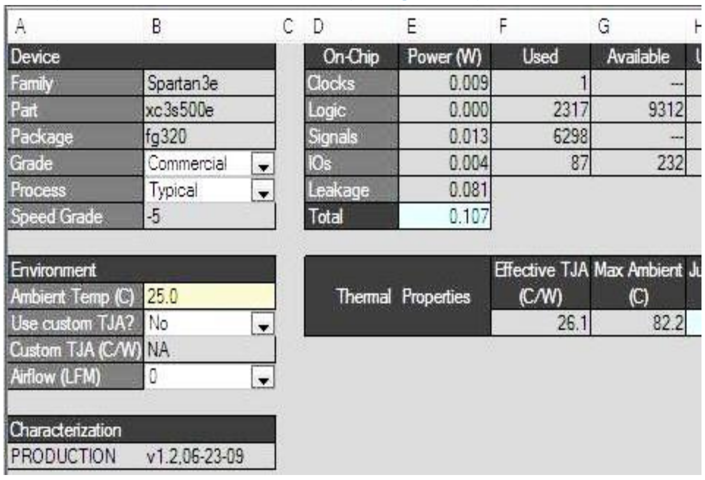
Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)	Junction Temp (C)
	26.1	81.5	28.5

Characterization	
PRODUCTION	v1.2,06-23-09

Figure10: Conventional 2-level cache synthesis report

B. Synthesis report with way-tag array



Device	
Family	Spartan3e
Part	xc3s500e
Package	fg320
Grade	Commercial
Process	Typical
Speed Grade	-5

On-Chip	Power (W)	Used	Available
Clocks	0.009	1	---
Logic	0.000	2317	9312
Signals	0.013	6298	---
IOs	0.004	87	232
Leakage	0.081		
Total	0.107		

Environment	
Ambient Temp (C)	25.0
Use custom TJA?	No
Custom TJA (C/W)	NA
Airflow (LFM)	0

Thermal Properties	Effective TJA (C/W)	Max Ambient (C)
	26.1	82.2

Characterization	
PRODUCTION	v1.2,06-23-09

Figure11: Synthesis report with way-tag array

7.Further Extension

We can extend this 2-way tag L2 cache architecture to 4-way tag L2 architecture for more high performance. The below figure shows the architecture of the 4-way tag L2Cache.

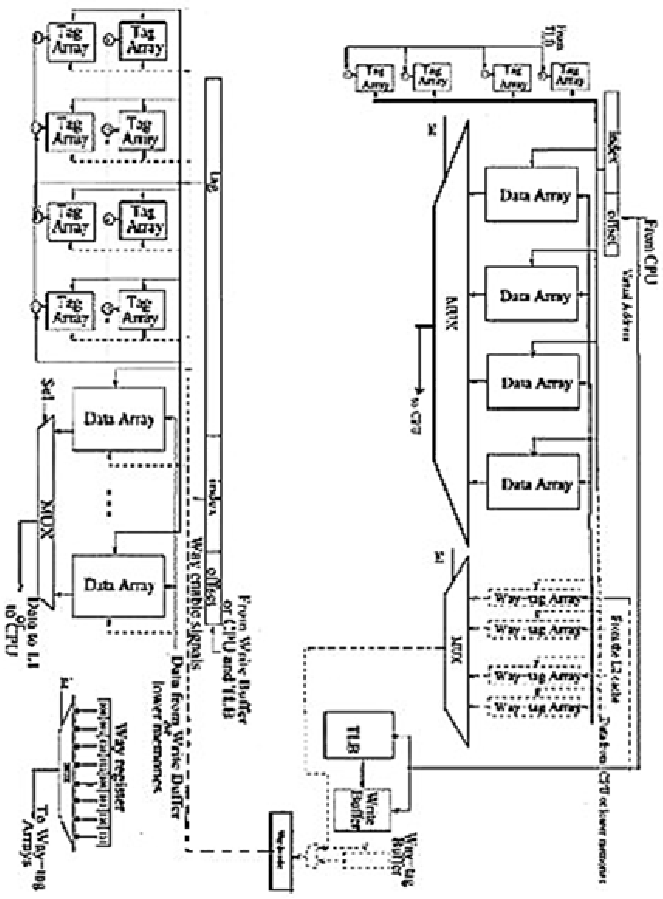


Figure 12:The Architecture of the 4-way tag L2 Cache

8. Conclusion

This project 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. Furthermore, the idea of way tagging can be applied to many existing low-power cache techniques such as the phased access cache to further reduce cache energy consumption.

References

1. G. H.Asadi,V. Sridharan, M. B. Tahoori, Andd.Kaeli, “Balancing Performance And Reliability In The Memory Hierarchy,” In Proc. Int. Symp.Perform. Anal. Syst. Softw., 2005, Pp. 269–279.

2. C. Su And A. Despain, “Cache Design Tradeoffs For Power And Performance Optimization: A Case Study,” In Proc. Int. Symp. Low Power Electron. Design, 1997, Pp. 63–68.

3. 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.

4. B. Brock And M. Exerman, “Cache Latencies Of The Powerpc Mpc7451,” Freescale Semiconductor, Austin, Tx, 2006. [Online]. Available: Cache.Freescale.Com

5. A.Ma, M. Zhang, And K.Asanovi, “Way Memoization To Reduce Fetch Energy In Instruction Caches,” In Proc. Isca Workshop Complexity Effective Design, 2001, Pp. 1–9.

6. 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.

7. 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.

8. B. Calder, D. Grunwald, And J Emer, "Predictive Sequential Associative Cache," In Proc. 2nd IeeeSymp. High-Perform.Comput. Arch., 1996, Pp. 244–254.

9. T. N. Vijaykumar, "Reactive-Associative Caches," In Proc. Int. Conf. Parallel Arch. Compiler Tech., 2011, P. 4961.

10. J. Dai And L. Wang, "Way-Tagged Cache: An Energy Efficient L2 Cache Architecture Under Write Through Policy," In Proc. Int. Symp. Low Power Electron.Design, 2009, Pp. 159 164.

11. R.Min,W. Jone, And Y. Hu, "Phased Tag Cache: An Efficient Low Power Cache System," In Proc. Int. Symp. Circuits Syst., 2004, Pp. 23–26.

Mr.A.Sudhakar received B.Tech degree in Electronics and Communication Engineering from Balaji Institute of Technology and Sciences(Affiliated to JNTUH) in 2013, pursuing M.Tech (2013-2015) in the stream of VLSI System Design at Vaagdevi Engineering College (Affiliated to JNTUH).

Mrs.M.Swetha received M.Tech degree in VLSI System Design from Vaagdevi College of Engineering(Affiliated to JNTUH), Presently working as Assistance professor in Vaagdevi Engineering College.