Comparative Study of Reconfigurable Cache Memory

Safaa S. Omran
Department of Computer Engineering
College of Electrical and Electronic Engineering Techniques
Baghdad, Iraq
Omran_safaa@ymail.com

Ibrahim A. Amory
Department of Computer Engineering
College of Electrical and Electronic Engineering Techniques
Baghdad, Iraq
Ibrahim_a7@yahoo.com

Abstract

Reconfigurable cache memory is important to improve the cache performance and reduces the energy consumption. In this paper, a review for previous papers related with reconfigurable cache memory were presented and compared it with our work in which we implemented two dimensional reconfigurable cache memory with exploitation of full amount of cache memory size with different organizations and compare the performance of different cache organizations.

Keywords: Reconfigurable Cache; Hit ratio; Energy consumption; SRAM; FPGA.

Introduction

The modern microprocessors performance depends on the processing speed and the energy saving feature. The cache memory is placed between the CPU and main memory and it represents important part because it reduces the speed gap between CPU (Central Processor Unit) and RAM (Random Access Memory). The cache memory prefetching the data from the main memory before the processor request for it, that occurs with phenomenon known as locality of reference and there are two principles of locality: (spatial locality) and (temporal locality). The spatial locality principle states that programs tend to access data and instructions sequentially, and temporal locality refers to the tendency of programs to repeatedly use a small part of code/data over a certain time [1].

Power consumption has become a major design consideration. The cache memory consume about 50% of a microprocessor's energy. The performance of cache memory is approximately determined by specific applications. Better energy performance is achieved by different applications with different configuration of the

DOI: 10.24086/cuesj.si.2017.n1a3

cache memory. Several solutions have been proposed to reduce the energy consumption of the cache memory.

Comprehensive Studies

A selective cache ways proposed by David H.Albonesi [2] exploits the fact that large on-chip caches are often partitioned into multiple sub arrays to reduce the long word delays of a single large array. This is for the data array and for set associative caches for which the word lines can be exceedingly long.

The partitioning required to combine hardware and software elements. A partitioning of the data and tag arrays into one or more subarrays for each cache way. Also a gating hardware and decision logic for disabling the operation of particular ways. Design a cache way select register as a software-visible register that signals the hardware to enable/disable particular ways.

Fig. 1 shows the 4 way set associative cache using selective cache ways.

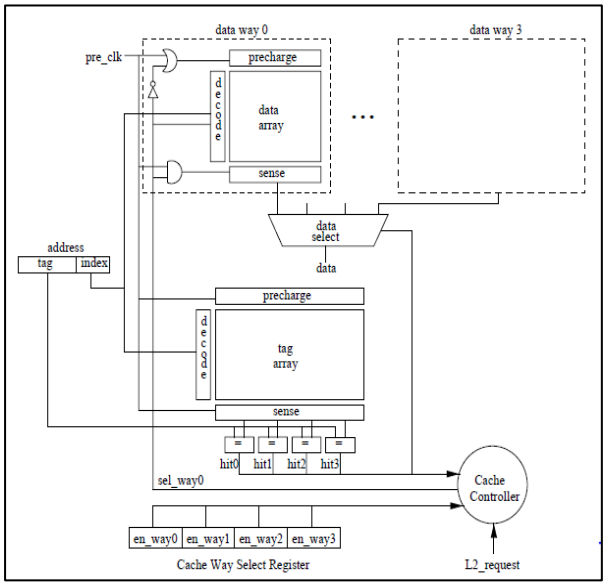

Fig. 1. 4-way Set Associative Cache using Selective Cache ways.

One advantage of selective cache ways over static solutions to energy reduction is that the tradeoff between performance and energy dissipation can be made variable via different Performance Degradation Threshold values.

The results shown that for the 4-way set associative caches, performance degrades by 0-4% when only three ways are enabled, and more sharply thereafter. The degradation is particularly significant (up to 17%) when moving from two ways to only one way enabled.

This is also the case for the 2-way set associative caches, in which performance degrades by 2-11% when only one of the two ways is enabled.

The energy savings for 2-way set associative caches is significantly less than that of 4-way caches of the same size for two reasons. First, a large jump in conflict misses often accompanies a move from two ways to only one way enabled. Second, the higher granularity of partitioning in the 4-way caches allows for the potential of disabling a greater percentage of the cache. Thus, in general, the effectiveness of selective cache ways increases with the cache associativity.

A 40% reduction in overall cache energy dissipation can be achieved for 4-way set associative caches with less than a 2% overall performance degradation.

A reconfigurable cache design proposed by Ranganathan et al. [3] enables the cache SRAM arrays to be dynamically divided into multiple partitions that can be used for different processor activities.

They divide the reconfigurable cache into partitions at the granularity of the ways of the conventional cache, exploiting the conceptual division into ways already present in a conventional cache.

A reconfigurable cache will require an equal or greater number of subarrays than a non-reconfigurable cache.

An increase in the number of subarrays can have two effects: if the number of subarrays is larger, the wire area and delay required to connect them will be larger, and a larger number of subarrays results in a reduction in the number of bits in each subarray, which means the individual subarray access times are reduced.

Their results show that a reconfigurable cache organization can increase the cache access time by anywhere between 1% to 15%. For small numbers of partitions (2-way), reconfigurable caches usually increase the access time of the non-configurable baseline cache by less than 5% (4% for a 128KB cache and 1% for a 1MB cache).

For larger numbers of partitions (4-way and 8-way), reconfigurable caches suffer a relatively larger cache access time penalty, particularly for smaller cache sizes (7-15% for the 128KB cache and 2-6% for the 1MB cache).

Zhang et al. [4] proposed way-concatenation depending on the associativity it can reduce dynamic power by accessing fewer ways. This was performed before an application started execution.

They choose to use a base cache of 8 Kbytes having four way set-associativity and a line size of 32 bytes. The base cache is the cache that they extend to be configurable.

A direct mapped cache has a high miss rate, resulting in higher energy due to longer time as well as high power for accessing the next level memory. Increasing cache associativity can decrease the cache miss rate and hence reduce energy.

Although accessing a four-way set associative cache requires more power per access, that extra power may be compensated for by the reduction in time and power that would have been caused by misses.

Fig. 2 shows the miss rate and normalized energy for 8 Kbyte data cache with different associativity.

They used two single-bit registers reg0 and reg1 that can be set to configure the cache as four, two or one way associative. Those two bits are combined with address bits a11 and a12 in a configuration circuit to generate four signals c0, c1, c2, c3, which are in turn used to control the configuration of the four ways.

Their results show a way- concatenaTable cache results in an average energy savings of 37% compared to a conventional four-way cache, with savings over 60% for several examples. Compared to a conventional direct mapped cache, the average savings are more modest, but the direct mapped cache suffers large penalties for some examples – up to 284% for parser, with degraded performance in several examples.

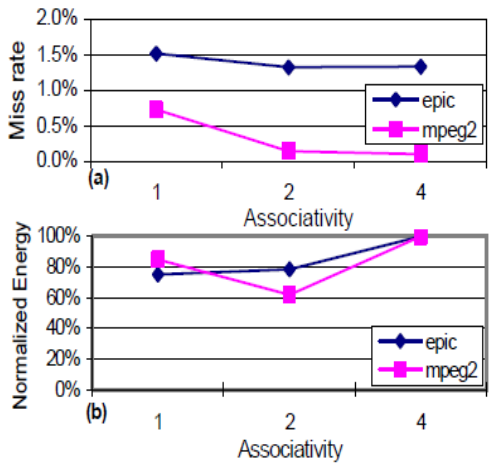

Fig. 2. Miss rate and normalized energy for data cache of different associativity.

Santana Gil et al. [5] proposed a design for reconfigurable cache with fixed size, the cache design can work as direct mapped cache or as 2 way set associative cache, and can select 1, 2, 4 or 8 words per block for each mode.

The core of the reconfiguration mechanism is the Control Block. The Control Block is a pure combinational circuit. Once the working mode is selected, the reconfiguration process with no latency. Then, the cache is ready to work in the new selected mode at the next memory access.

The Control Block indicates if the cache operation resulted in a hit or a miss and proceeds controlling all processor buses, to provide data from cache or system RAM, according to the operation result.

They design a data cache memory of 2 K words of 32 bits (8 KB). Additional amount of memory is required to store the tags, valid bits and LRU bits.

The design was implemented on a Xilinx Spartan 3 starter board with a xc3s200-ft256 FPGA. The cache structure was described using Handel-C as a hardware description language.

Sundararajan et al. [6] presented a Set and Way management cache architecture for Run-Time reconfiguration that allows to change the size and associativity. The associativity can be selected as (Direct Mapped (DM), 2 Way and 4 Way) with sizes (64KB, 128KB, 256KB, 512K Byte, 1M Byte and 2M Byte).

They designed a cache memory with set selection and group the sets in each bank by augmenting the cache with size selection bits that determine the sets that are enabled. And the size selection bits then ANDed with bits from the index to determine the sets to access.

The five size selected bits representing cache size of 128KB, 256KB, 512KB, 1MB and 2MB. The 64KB cache is always enabled.

A way selection circuits control the associativity of the cache and it could be Direct Mapped (one way), 2 way and 4 way set associative.

Fig. 3 shows the accessed, unaccessed and disabled sets for each case of cache associativities.

The differences between this Smart cache and other reconfiguration techniques. The associativity and size are varied in parallel by using the way control signals and the size control registers. The Smart cache organizes ways at set boundaries, which avoids flushing data back to memory when increasing the associativity but keeping the cache size fixed.

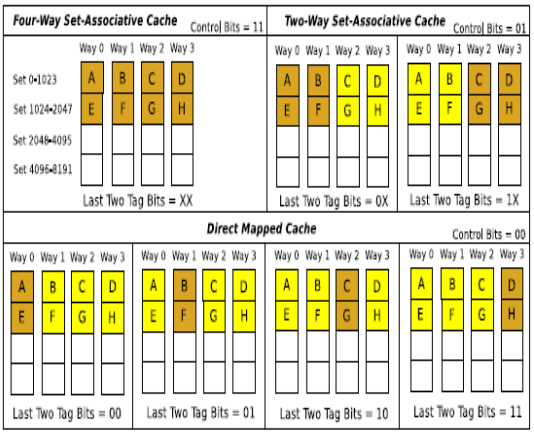

Fig. 3. The smart cache. Brown, yellow and white regions show accessed, unaccessed and disabled sets respectively.

They implemented cache reconfiguration using the HotLeakage simulator.

Their results shows that for some benchmarks the best static configuration is 2MB with eight-way associativity so there are no energy savings achievable for any cache architecture. The set-only, set-and-way and Smart approaches consume around 1.7% more energy compared to way-concatenation.

For other some benchmarks, the set-only and set-and-way approaches do not do well compared to the way-concatenation and Smart caches. The reason for this is that these benchmarks require a 2MB cache with two-way associativity which is only offered by way-concatenation and Smart cache. For these architectures, dynamic energy is reduced by accessing fewer ways, which is not possible in the set-only and set-and-way caches.

The average cache energy delay is 0.28, which is 14% better than set-only and set-and-way approaches and 25% better than the way-concatenation with way-shutdown approach. This clearly demonstrates the benefits of using their smart cache reconfiguration architecture.

Zhenglin Liu et al. [7] presented an efficient reconfiguration management algorithm (RMA) to improve the performance and energy-efficiency of the memory hierarchy.

They have described the differences between the variation in the energy consumption and the variation in the performance for different cache configurations. It is this difference in the variation that motivated to develop the RMA.

They used cache capacities of 4, 8, 16, 32, 64, and 128KB, associativities of 1, 2, 4 and 8-way, and line sizes of 16 and 32 Bytes. The total cache capacity has the biggest average impact on energy and miss rate, line size has little impact for instruction cache, but more impact for data cache because of spatial locality, and associativity has least impact among the three.

They refer to the method of overlapped wide-tag partitioning, which enables the cache partitions to potentially be any size, however, they limit them to be powers of two to enable simpler decoding. Because of 6 choices for the cache capacity. Thus, it is reasonable to physically partition the 128KB cache into 6 subarrays: 8KB, 16KB, 32KB and 64KB for each subarray, and 4KB for two subarrays.

The cache memory have "physical line" 8 Bytes, but to fetch and replace a variable number of words simultaneously as a "virtual line". Besides, the approach allows the line size to be changed dynamically. They added a counter to cache controller to specify how many words to read from the off-chip memory, and determine how many cache physical lines will be filled when there is a miss.

They utilize an 8-way set-associativity cache as baseline architecture and each way is enabled by one of the Cache Way Enable Signals using some decision logic and gating hardware. When a Way is disabled, no data is selected from a disabled way and its data array dissipates essentially no dynamic power.

Their results shows the cache of lowest energy dissipation is 32K8W64B with miss rate 0.0118 and energy dissipation 34.159nJ, however, the "optimal" cache by RMA is 64K8W64B with miss rate 0.0082 and energy dissipation 34.475nJ. By increasing the cache size from 32KB to 64KB, they achieve a significant reduction in miss rate by 30.51% with an increase in energy dissipation by only 0.93%. Thus, the "optimal" cache configuration of 64KB8W64B is reasonable, due to the fact that the reduced miss rate is large enough to overcome the added energy consumed by the cache itself.

C J Janraj, et al. [8] proposed way sharing cache, where two sets share a subset of cache ways apart from having their own cache ways, ways are shared among a pair of sets. The total number of sets in their design is the half of total number of conventional cache have the same degree of associativity.

Their way-sharing cache has 64 pairs of shared sets with 6 ways in each pair, it is denoted as 64×(6, 2)-way sharing DL1 cache. Sharing in their technique enables a flexibility of 2, 3, or 4-way associativity available to each set in the shared pair of sets. Each set of between (0 to 63) of the conventional cache is sharing a set in their way

sharing cache with its friend set of between (64 to 127), such that s and p sets own two ways each and share two ways among them.

They run 20 SPEC2000 CPU Benchmarks on modified simplescalar tool and considered the base L1 data cache configuration as 16KB DL1 cache with 4-way associativity and 32B blocks. Hence, the total number of sets in the base L1 data cache is 128. They denoted the base DL1 cache as 128×4-way DL1 cache. Similarly, the base L2 unified cache configuration is referred as 1024×8-way cache with 64B block.

In their results, the 64×(6, 2)-way sharing DL1 cache takes one-and-half times the area of 8KB 4-way DL1 cache and 512 × (12, 4)-way sharing unified L2 (UL2) cache takes one-and-half times that of 256KB 8-way L2 cache. Since these are more like approximations, they conservatively consider an additional 10% overhead each for access time, energy and area.

The way-sharing DL1 cache has better per-access energy, area values, and almost equal access time to those of conventional cache, respectively. In the case of UL2 cache with fast-type, the way-sharing mechanism gives better access time and energy with less area. The way sharing mechanism may not be effective in UL2 caches with serial-type as serial-type caches are already optimized for energy consumption.

They consider 512 × (12, 4)-way sharing UL2 cache and compare it with 1024×8-way base UL2 cache to know the effectiveness of their way-sharing mechanism when it is applied to unified L2 cache alone. And they consider both serial type and fast-type modes, the way-sharing mechanism is not effective when it is applied to UL2 caches with serial-type and it incurs an average energy overhead of 0.7% (for "bzip" overhead is as the way-sharing mechanism reduces the overall memory energy by 6%. Fast-type L2 caches are generally used in systems when performance is more critical (example, Intel Core family processors). In such systems can apply the way-sharing mechanism in L2 caches also (we observe that it incurs negligible performance penalty).

Jungwoo Park, et al. [9] proposed a cache architecture that can logically increase cache associativity of way-powered-down LLCs. Their proposed scheme is designed to be dynamic in activating an appropriate number of cache ways in order to eliminate the need for static profiling to determine an energy-optimized cache configuration.

They performed experiments using the Sniper simulator and SPEC CPU2000 benchmarks.

They proposed a cache architecture that can be applied to set-associative LLCs. They call this cache architecture "logical–associative cache."

Their first idea to design the "logical–associative cache" architecture is to activate all tag ways to access them in parallel and not to activate all data ways to reduce their leakage energy consumption. Increasing cache associativity to support parallel access is a very effective solution to increase performance. They compared the execution times, energy consumptions, EDPs, and LLC misses of a two-way 1024-set LLC configuration and a 16-way 128-set LLC configuration. These two cache configurations have the same capacity (256 kbytes). The results shows on average, the 16-way 128-set cache shows 7.4% better performance, 5.4% less energy, and 10.9% less EDP compared with the two-way 1024-set cache. If the capacity is the same, increasing the cache associativity is a good solution for increasing the system performance. Their idea to increase parallelism is shown in Fig. 4, where the tag ways are physically set-associative, whereas the data ways are logically associative within the activated data ways. In this example, cache associativity is four.

They apply logical association to LLCs in which the tag ways are accessed first and then a matching data way is accessed to reduce their dynamic energy consumption

They also proposed the Way-Filtering-based logical–associative LLC architecture to reduce the energy consumption in the tag ways of LLCs, they apply a partial tag matching scheme. It extracts a few bits from the tag bits to early identify a cache miss. A partial tag consists of a few least significant bits from the original tag bits and a few most significant bits from the index bits. Because the cache lines in a cache way are logically divided, the most significant bits (3 bits for eight logical ways) from the index bits are used as part of a partial tag. Its because the number of logical cache sets becomes smaller than that of cache sets when logical cache ways is applied. A partial tag-based way filter is allocated to each cache way, and it is powered down when its corresponding cache way is turned OFF. Experimentally, they find that a 4-bit partial tag and eight logical ways show optimized results.

They evaluated their proposed architecture and observed a 3.4% total system energy and 34% LLC energy savings over the selective-cache-ways scheme with a relatively small overhead on a single-core system. Their proposed scheme shows more energy reduction results on multicore systems with more capacity pressure on the LLC. They observed that the total energy consumption and EDP are reduced by 9.2% and 11.8%, respectively, over the selective-cache-ways scheme on a quad-core system.

Bhargavi R.Upadhyay and Sudarshan TSB [10] surveyed different techniques to find the efficient design space aimed at reducing the design space time and provide good insight to researchers to explore further.

Simulation based design exploration leverages software to find the exact cache parameters. Simulators can be classifying as a functional simulator, Full system

simulator and trace driven simulator. Functional simulator provides the functional correctness with hit-rate and miss-rate. Functional simulators can be used to trace the memory accesses for the application.

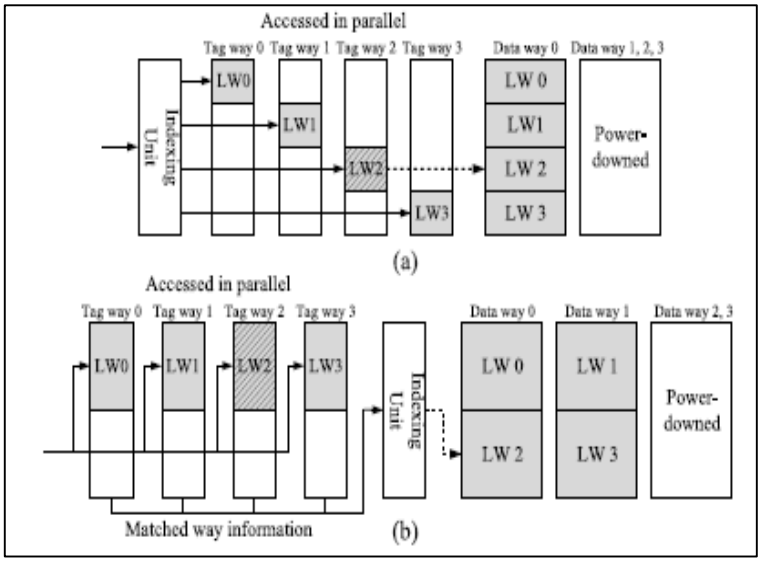

Fig. 4. Non filtering parallel logical–associative cache for a four-way cache. (a) and (b) All tag ways are accessed in parallel. The solid arrows indicate tag way accesses and the dotted arrows indicate data way accesses when a tag hit occurs. LW stands for logical way.

They studied several simulators like Simple scalar, SimOS, Simics, Gem5 and trace driven simulators.

Trace driven simulators takes a collected memory access pattern for an application as input which gives the output as performance matrix like hit rate. Simulator traces millions of memory reference for the application for the different cache configuration for the application without building the costly hardware. Single pass can take only one configuration, for another configuration we need to run the simulator again. Repeatedly running Simulation process for different configuration of the application can take hours or days.

They mentioned several previously work of reconfigurable cache memory and the hardware units which required for dynamic cache reconfigurations, and also the evolutionary technique which widely used to optimize software and hardware design techniques. Techniques mentioned there are based on offline traces collection and runtime behavior of applications. These approaches used the cache hit ratio, miss ratio, CPI as a performance matrix to evaluate the cache performance.

Safaa S. Omran and Ibrahim A. Amory [11] Presented Two Dimensional Reconfigurable Cache memory implemented on FPGA. The design allows reconfiguration in both associativity and size of the cache memory, the memory size can be (64K Byte, 128K Byte, 256K Byte and 512K byte) and the associativity can be (Direct mapped (DM), 2 way, 4 way and 8 way set associative).

We designed eight sets of cache memory each with size 64K byte, for DM the maximum size is 512K byte and the CPU is dealing with the eight sets as one set with size 512K byte.

For set associative organization the maximum cache size is 512K byte too, hence, the maximum cache memory size for each organization is equal to the already occupied cache memory size, then there is no waste in cache memory size.

The cache design consists of cache size controller unit, cache set controller unit, and cache memory (cache data memory and cache controller which consists of cache tag memory, LRU controller unit, General Mux and finite state machine) as shown in Fig. 5.

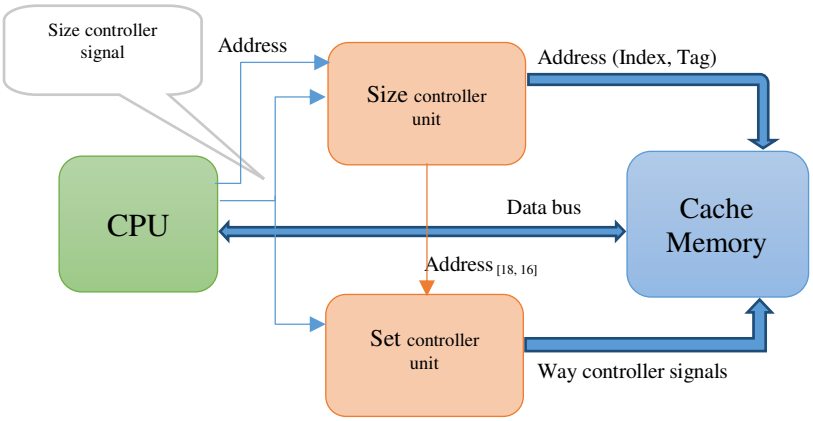

Fig. 5. Cache memory design.

Cache size controller unit responsible for managing the cache size and mapping function, when the processor request an address, this address will be send for the cache size controller unit, the address consists of 32 bits as shown in Fig. 6.

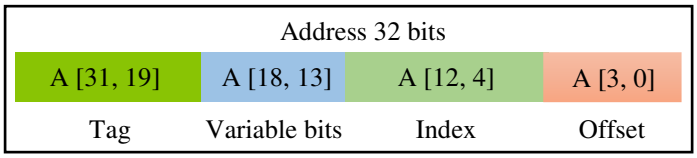

Fig. 6. 32 bits Address

The required cache size and mapping function are chosen by CPU by sending Size controller signals (4 bits) at run time to the cache size controller unit.

Cache size controller unit produces 6 bits called (I bits) which ANDed with the 6 bits of A [18, 13] produces the index bits which will send to the cache memory to indicate which line is requested to access. At the same time A [18, 13] will be ANDed with another 6 bits called (T bits) to produce the tag bits.

Cache memory size of 512K bytes is divided into 8 sets of cache memory each with size of 64K byte.

- cache size 64K byte: with DM the CPU dealing with the first set only with size 64K byte while the other 7 sets is unpowered, with 2 way set associative the CPU dealing with two sets with size 32K byte from each one, with 4 way the CPU dealing with four sets with size 16K byte from each one and with 8 way the CPU dealing with all 8 sets with size 8K byte from each one.

- cache size 128K byte: with DM the CPU dealing with two sets as one set with size 128K byte while the other 6 sets is unpowered, with 2 way set associative the CPU dealing with two sets with size 64K byte for each one, with 4 way the CPU dealing with four sets with size 32K byte from each one and with 8 way the CPU dealing with all 8 sets with size 16K byte from each one.

- cache size 256K byte: with DM the CPU dealing with four sets as one set with size 256K byte while the other 4 sets is unpowered, with 2 way set associative the CPU dealing with four sets as two set with size 128K byte for each one, with 4 way the CPU dealing with four sets with size 64K byte for each one and with 8 way the CPU dealing with all 8 sets with size 32K byte from each one.

- cache size 512K byte: with DM the CPU dealing with all 8 sets as one set with size 512K byte, with 2 way set associative the CPU dealing with 8 sets as two set with size 256K byte for each one, with 4 way the CPU dealing with four sets with size 128K byte for each one and with 8 way the CPU dealing with all 8 sets with size 64K byte for each one.

The dealing of CPU with several sets of cache as one set is controlled by Cache way controller unit, depending on the selected cache size and associativity and on the location of requested line, each set covered 64K byte (4096 Line each with size 16 byte).

Fig. 7 shows the all 8 sets of cache memory each with size 64KB. While a 256KB cache size selected with Direct Mapped organization. The CPU now dealing with the first 4 sets as one set of cache memory with size 256KB. While another 4 sets with red color unaccessed.

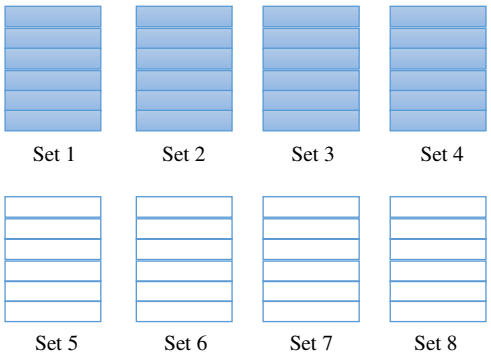

Fig. 7. Cache Sets with DM and size 256KB.

Fig. 8 shows a 256KB cache size selected with two way set associative. The CPU now dealing with the first 2 sets (Set1 and Set2) with blue color as Way0 with size 128KB and with Set3 and Set4 with green color as Way1. While another 4 sets with red color unaccessed.

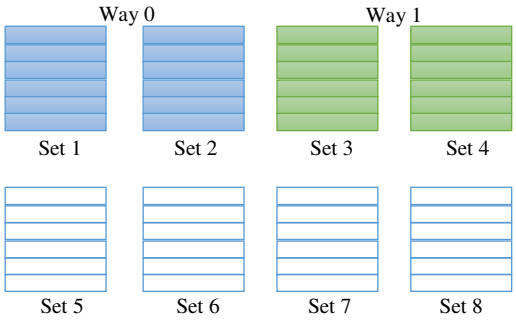

Fig. 8. Cache Sets with 2 Way and size 256KB.

In case of Cache full size 512KB and 8 Way set associative. Each set individually representing way of 8 ways of cache memory.

Way Controller Unit is controlling the dealing of CPU with several sets as one cache memory set.

General Mux responsible for connecting the input and output data buses to and from cache sets relying on the signal (Dwith) which sent from FSM. LRU controller Unit indicates which way least recently used in case of set associative is selected.

The two dimensional reconfigurable cache memory implemented using FPGA (Field-Programmable Gate Array).

Results

A complete design of a 32-bit MIPS (Microprocessor without Interlocked Pipeline Stages) is used to test the reconfigurable cache memory by execute several test programs.

A direct mapped cache less time access because it accesses only one tag and data array per access, while in set associative cache memory the cache accesses many tag and data arrays per access.

For some applications a DM cache memory exhibits a very poor hit rate and this causes poor performance. Adding set associativity increases the hit rate, but for many applications, the additional associativity is unnecessary and thus results in wasted energy and longer access time.

In this paper a model of a maximum cache size up to 512KB for any mapping function was designed. A 512KB cache size for direct mapped, 2-way, 4-way or 8-way can be selected and the design is programmed using VHDL and implemented on FPGA.

The technique were used in this design is different from other papers were they need more amount of cache size from the used size. For example in smart cache design they need 8MB cache size to implement a 2MB cache of reconfigurable 4-way, 2-way or Direct Mapped cache memory.

Conclusions

From the comprehensive studies for several papers of reconfigurable cache memory. It is clear that the reconfigurable cache memory is important to improve the cache performance and reduces the energy consumption.

The performance of the cache memory is largely determined by the specific application. Different applications achieve better energy-performance with different configurations of the cache memory hierarchy.

The direct mapped have better access time and energy saving feature but it have higher miss rate compare with the set associative mapping which have longer access time and more energy consumption.

References

[1]. Sivarama P. Dandamudi, *Fundamentals of Computer Organization and Design.* 2nd ed. New York, USA: Springer, 2002.

[2]. D. H. Albonesi, "Selective Cache ways: on demand cache resource allocation," in Proc. 32nd International Symposium on Microarchitecture, 1999, pp. 248-259.

[3]. P. Ranganathan, S. Adve, and N. Jouppi. "Reconfigurable Caches and their Application to Media Processing". In 27th International Symposium on Computer Architecture, June 2000.

[4]. C. Zhang, F. Vahid, and W. Najjar, "A highly configurable cache architecture for embedded systems," in *ISCA*, 2003.

[5]. Santana Gil, A.D., Benavides, Hernandez, Herruzo, "Reconfigurable Cache implemented on an FPGA" International Conference on Reconfigurable Computing and FPGAs, IEEE. 2010.

[6]. Karthik T. Sundararajan, Timothy M. Jones and Nigel Topham, "Smart Cache: A Self Adaptive Cache Architecture for Energy Efficiency". In International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, IEEE, 2011.

[7]. Liming Chen, Xuecheng Zou, Jianming Lei and Zhenglin Liu, "Dynamically Reconfigurable Cache for Low-Power Embedded System". Third International Conference on Natural Computation, IEEE, 2007.

[8]. C.J. Janraj, T. Kalyan, T. Warrier, M. Mutyam, "Way sharing set associative cache architecture", in: 25th International Conference on VLSI Design (VLSID), 2012, pp. 251–256.

[9]. Jungwoo Park, Jongmin Lee and Soontae Kim, "A Way-Filtering-Based Dynamic Logical–Associative Cache Architecture for Low-Energy Consumption". In IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 25, 2017, pp. 793 – 805.

[10]. Bhargavi R. Upadhyay and T S B Sudarshan, "Design Space Exploration of Cache Memory –A Survey". In ICEEOT, IEEE, 2016.

[11]. Safaa S. Omran and Ibrahim A. Amory, "Design of Two Dimensional Reconfigurable Cache memory using FPGA".In ICEDSA IEEE, UAE, 2016.