2018 Third International Conference on Electrical, Electronics, Communication, Computer Technologies and Optimization Techniques (ICEECCOT) 14-15, December 2018

Design of Reconfigurable Cache Memory Using Verilog HDL

Manjunatha K N,
VLSI design and Embedded Systems
VTU Extension Centre,
UTL Technologies Ltd
Bangalore, India
vijaymanjunatha9.kn@gmail.com

Kanagasabapati
VLSI design and Embedded Systems
VTU Extension Centre,
UTL Technologies Ltd
Bangalore, India

Siva S Yellampalli
VLSI design and Embedded Systems
VTU Extension Centre,
UTL Technologies Ltd
Bangalore, India

***Abstract*—Verilog Hardware Description Language is used to design cache memory which involves direct mapping and set associative cache. Further set associative cache involves two-way, four-way and eight-way. In this design of cache memory architecture, the mapping technique can be varied using controller unit. To increase accessing speed and optimize power by disable unused cache memory set blocks.**

***Keywords*—Cache memory, Mapping Controller unit, hit.**

I. INTRODUCTION

In Today's current computers speed of the memory is very important and need for it is in demand. Cache memory plays an important role in increasing the speed. The block diagram of Cache Memory is shown in the Fig.1. Cache is placed in between processor and main memory. It will also contribute to the performance of CPU.

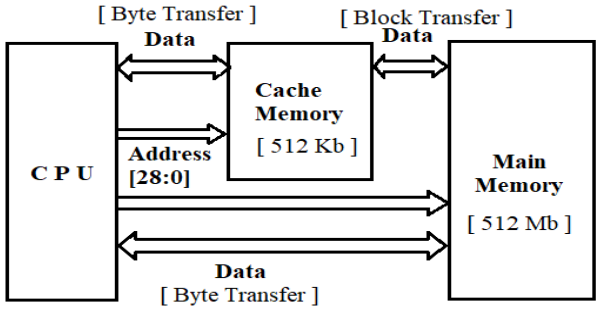

Fig.1: Basic Cache Memory Block Diagram.

To reduce speed gap between CPU and main memory cache memory was introduced, because main memory data transfer speed is less compared with CPU processing speed [3]. For preload data from main memory to cache memory address before accessing data from cache is called locality of reference [1].

Different ways of mapping techniques is used to access data from cache to speed up the performance of CPU [4] & [5]. In set associative mapping, cache set selection bit is used to access particular cache memory set [6].

Random Cache Replacement policy is used in this design [3]. Whenever, cache miss happens cache memory respective line is replaced with new main memory corresponding address block data.

Main Memory Design:

To map main memory of 512 MB with block size of 64 byte to cache memory.

- Total main memory lines is given by Main memory size / block size = $\frac{2^{29}}{2^{6}}$ = 8388608 lines.
- To represent main memory lines from 0 to 8388607 lines requires 23 main memory index bits i.e., CPU address bits [28:6].
- Remaining 6 address bits [5:0] is used as offset bits.

This paper explains cache memory architecture with the varied mapping technique using cache way controller unit. The introduction part of Section I explains the basic working of the cache memory. In Section II describes the working of different cache mapping technique. Section III explains implementation of cache way controller unit. Simulation results of each mapping technique is showed in Section IV.

II. CACHE MAPPING TECHNIQUES

The different ways of mapping of main memory to cache memory is as follows:

A. Direct mapping
B. Two way mapping
C. Four way mapping
D. Eight way mapping.

A. Direct Mapping

In direct mapping technique, to map main memory of 512 MB with block size of 64 byte to 512 Kb cache memory requires 29 bits of CPU address.

Fig.2: Direct Cache Memory Address bits.

The address bits of the Direct Cache Memory is shown in the Fig.2. Offset address [5:0] bits is used to represent byte address of respective cache line index address [18:6]. Tag and cache index line is used to represent mapped main memory address line.

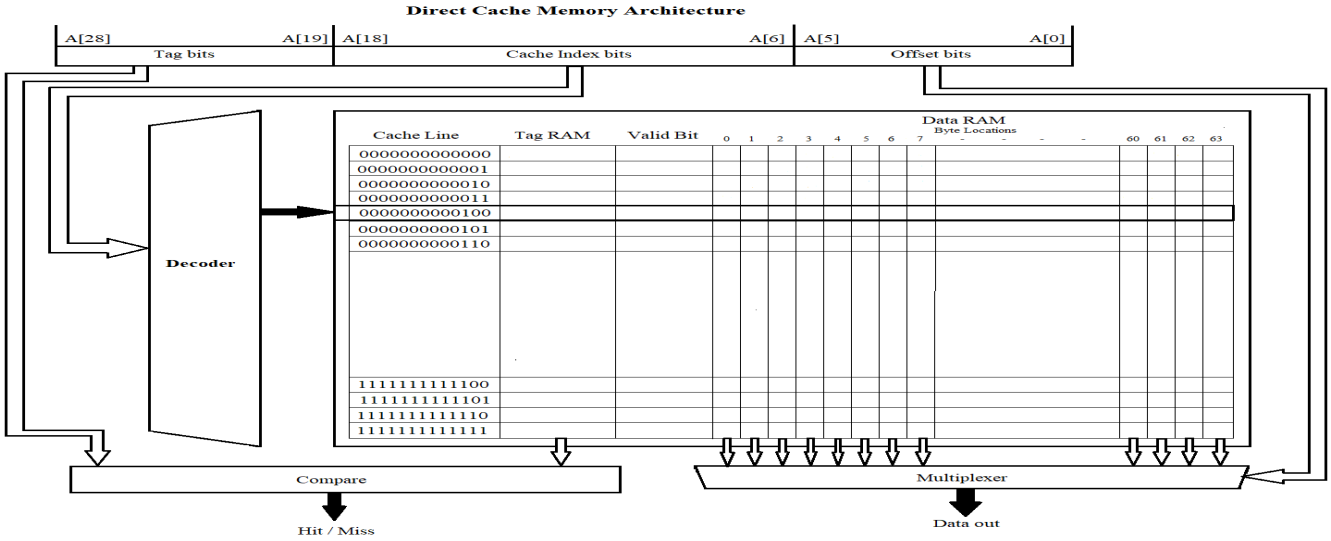

Fig.3: Direct Cache Memory Architecture.

The Fig.3 shows architecture of direct cache memory of 512 Kbyte with line size of 64 byte. Decoder is used to select each cache line and in each cache line is used to store address of respective tag bit. Valid bit is enabled whenever hit occurs so that CPU can access data stored in respective cache line. Offset address [5:0] is used to indicate byte data address.

- Total cache memory lines is Cache memory size by block size = $\frac{2^{19}}{2^{6}}$ = 8192 lines
- To represent cache memory lines from 0 to 8191 lines requires 13 cache memory index bits. (I.e. from 0000000000000 to 1111111111111).

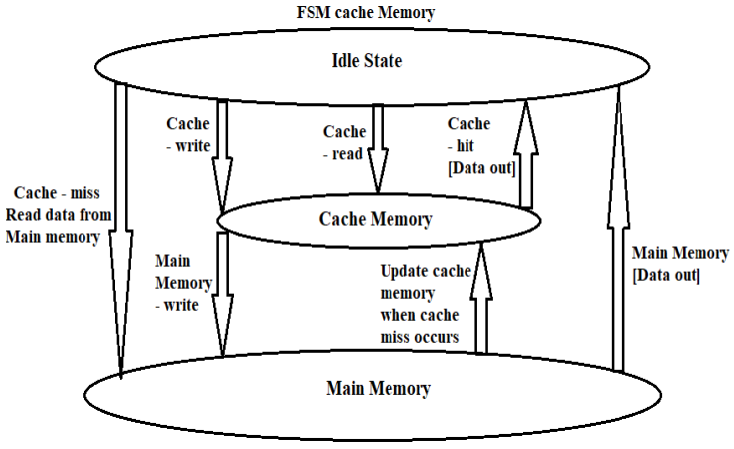

Fig.4: FSM of Direct Mapping.

The Fig.4 shows FSM of Direct Mapping. Initially, 8-bit data is written into each cache address byte location, then each cache line data is copied to respective main memory block location.

While read operation, initially CPU address bits [28:19] is compared with respective cache line tag bits. If it matches then cache hit and valid bit of respective cache line is set, then respective cache offset byte addressed data is send it to CPU. If it doesn't matches then cache hit and valid bit of respective cache line is reset and dirty bit is set, CPU access main memory block addressed offset byte data and meanwhile update tag bits and copy main memory addressed block data to respective cache line.

B. Two way mapping

In two way mapping, 512 Kbyte Cache memory is divided into two equal size of 256 Kbyte each memory. It is similar to direct mapping with two cache memory associative sets.

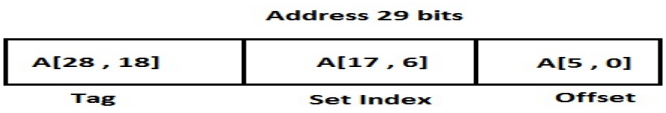

Fig.5: Two way Cache Memory Address bits.

The address bits of the Two- way Cache Memory is shown in the Fig.5. Main memory can be mapped to any of the two cache sets depending upon Cache Set selection CPU address bit [18] is shown in Table 1.

Table 1. Two-Way Mapping Cache Set selection bit.

2 - way Cache Memory Set Selection CPU Address bit [18]	Cache memory Bank Selection	Tag Bank
0	Cache memory - 0 Memory bank [256 Kb]	Tag - 0
1	Cache memory - 1 Memory bank [256 Kb]	Tag - 1

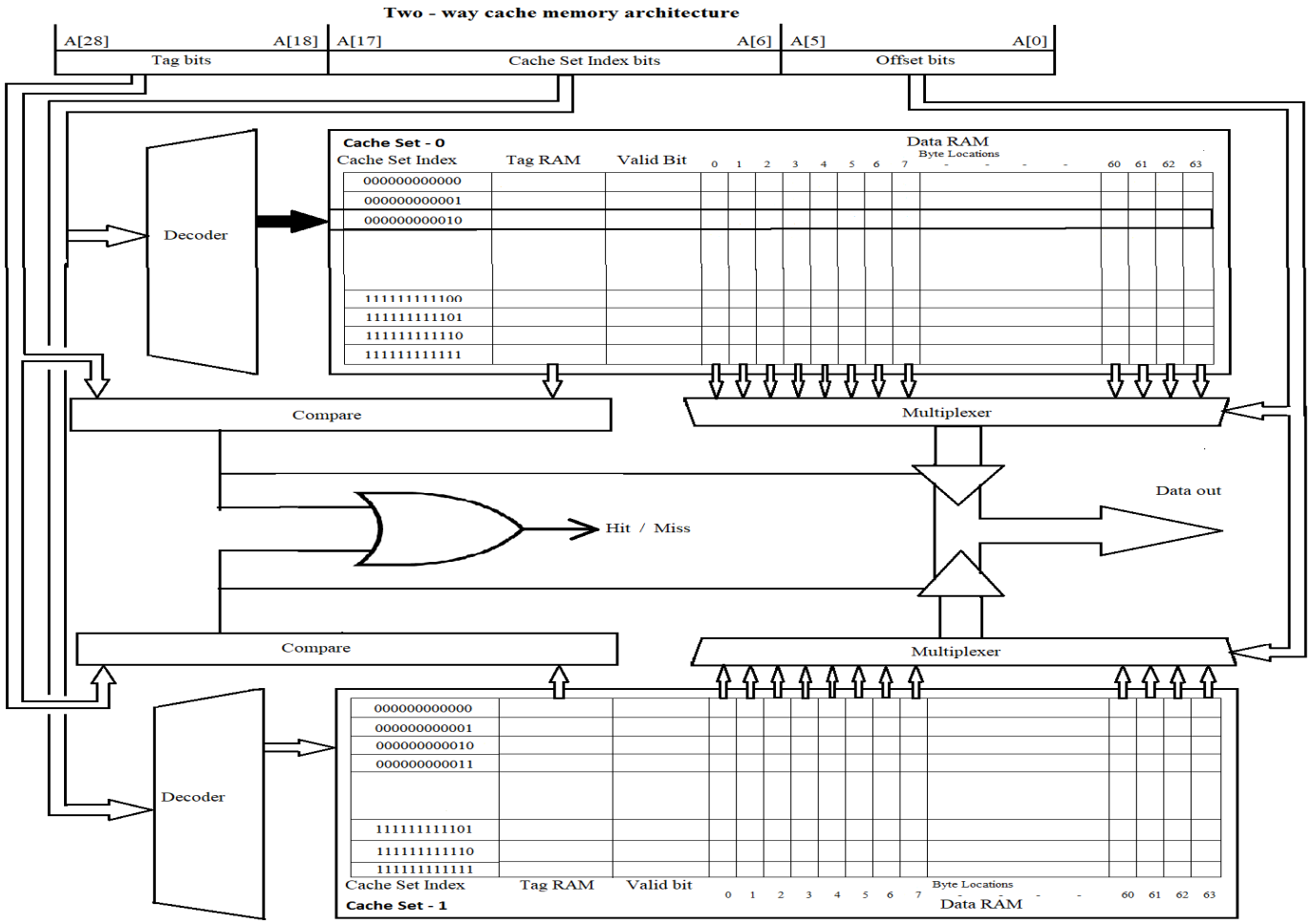

Fig.6: Two way Cache Memory Architecture.

The Fig.6 shows architecture of two-way mapping technique, to map main memory of 512 MB with block size of 64 byte to two sets of 256 Kb cache memory, requires 29 bits CPU address.

- Total cache memory lines in each set is Cache memory set size by block size = $\frac{2^{18}}{2^{6}}$ = 4096 lines
- To represent cache memory lines from 0 to 4095 lines requires 12 cache memory set index bits. (I.e. from 000000000000 to 111111111111).

Similar to Direct mapping, data is written to two cache set memory depending upon cache select bit. Here tag bits of cache lines is reduced when compared to direct mapping because cache is divided into two equal 256 Kbyte.

Tag bits is compared with address bits [28:18], if it matches hit else miss similar to direct mapping. Here, in each CPU address only one Cache Set size of 256 Kbyte is enabled and Searching time is less compared to direct mapping because of less cache set size.

C. Four way mapping

In Four way mapping, 512 Kbyte Cache memory is divided into four equal Cache set size of 128 Kbyte. The address bits of the four way Cache Memory is shown in the Fig.7 and is similar to two way mapping, but instead of two sets having four sets. Each Cache Set memory is selected using CPU address bits [18:17] shown in Table 2.

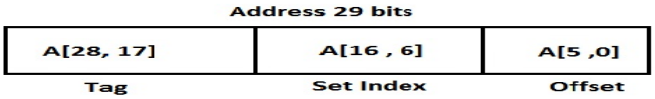

Fig.7: Four way Cache Memory Address bits.

Write and Read operation is similar to two way mapping. If CPU address bits [28:17] matches with stored cache tag bits, then hit and data is read from cache memory. Else miss then data is read from main memory and simultaneously update cache memory with missed memory block.

Each Cache Set memory is selected using CPU address bits [18:17] shown in Table 2.

Table 2. Four-Way Mapping Cache Set selection bits.

4 - way Cache Memory Set Selection CPU Address bits [18:17]	**Cache memory Bank Selection**	**Tag Bank**
00	Cache memory - 0 Memory bank [128 Kb]	Tag - 0
01	Cache memory - 1 Memory bank [128 Kb]	Tag - 1
10	Cache memory - 2 Memory bank [128 Kb]	Tag - 2
11	Cache memory - 3 Memory bank [128 Kb]	Tag - 3

Here, in each CPU address only one Cache Set size of 128 Kbyte is enabled and Searching speed is fast compared to two mapping because of less cache set size compared to two way.

D. Eight Way Mapping

In eight way mapping, 512 Kbyte Cache memory is divided into eight equal cache set size of 64 Kbyte each. The address bits of the eight way Cache Memory is shown in the Fig.8.

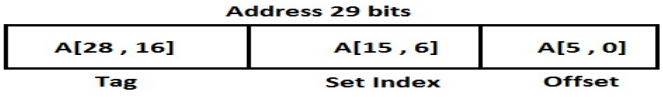

Fig.8: Eight way Cache Memory Address bits.

In eight way mapping, each Cache memory is selected using address bits [18:16] is shown in Table 3.

Table 3. Eight Way Mapping Cache Selection bits.

8 - way Cache Memory Set Selection CPU Address bits [18:16]	**Cache memory Bank Selection**	**Tag Bank**
000	Cache memory - 0 Memory bank [64 Kb]	Tag - 0
001	Cache memory - 1 Memory bank [64 Kb]	Tag - 1
010	Cache memory - 2 Memory bank [64 Kb]	Tag - 2
011	Cache memory - 3 Memory bank [64 Kb]	Tag - 3
100	Cache memory - 4 Memory bank [64 Kb]	Tag - 4
101	Cache memory - 5 Memory bank [64 Kb]	Tag - 5
110	Cache memory - 6 Memory bank [64 Kb]	Tag - 6
111	Cache memory - 7 Memory bank [64 Kb]	Tag - 7

Here, in each CPU address only one Cache Set size of 64 Kbyte is enabled and Searching speed is fast compared to two and four way mapping because of less Cache Set size.

III. IMPLEMENTATION

The Fig.9 shows Cache Way controller unit. Way controller unit is having two control bits is used to change mapping techniques according to designer requirement.

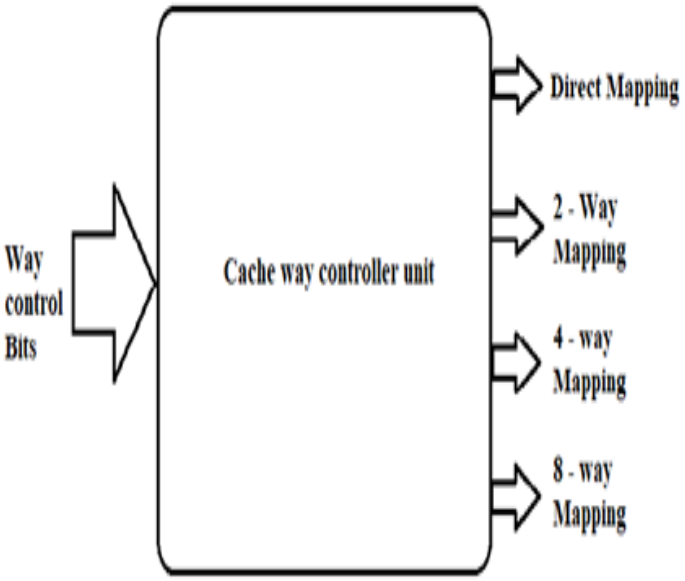

Fig.9: Cache Way controller unit.

Mapping is controlled using Way Controller Unit is shown in Table 4.

Table 4. Cache Mapping Selection bits.

Way Control Bits	Way of Mapping
00	Direct
01	Two way
10	Four way
11	Eight way

- If way control bit = 00, then cache acts direct mapping. Entire 512 Kbyte is enabled and working of direct mapping is explained in Section II [A].
- If way control bit = 01, then cache behaves as two way mapping. CPU access anyone cache sets out of two 256 Kbyte depending upon CPU address bit [18] is described in Section II [B].
- If way control bit = 10, then cache behaves as four way mapping. CPU access anyone cache sets out of four 128 Kbyte depending upon CPU address bits [18:17] is explained in Section II [C].
- If way control bit = 11, then cache behaves as eight way mapping. CPU access anyone cache sets out of eight 64 Kbyte depending upon CPU address bits [18:16] is described in Section II [D].

IV. Results and Simulation

The implemented design of cache way controller unit is simulated using Cadence NC-Sim. The simulation of different mapping technique for the cache way controller unit is presented in this section.

Direct Mapping Cache: In Fig.10, direct mapping single cache memory is used and to know that the data is present in cache which has to search all tag fields.

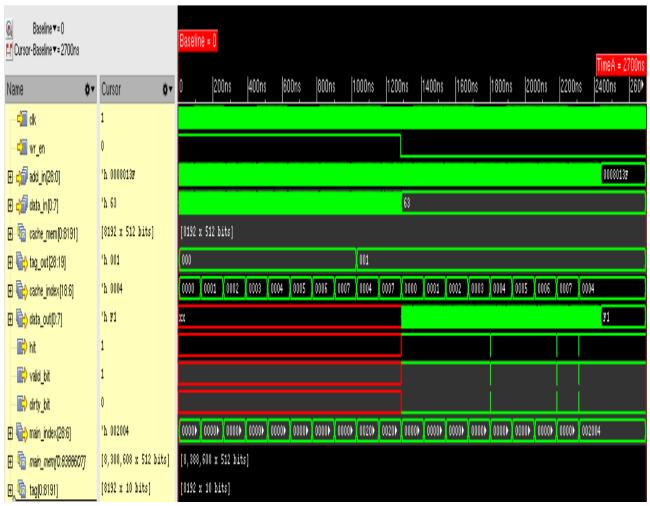

Fig.10: Direct Mapping Cache.

Two Way Cache Mapping: The two separate cache memory is used each size of 256Kbyte in Fig.11.

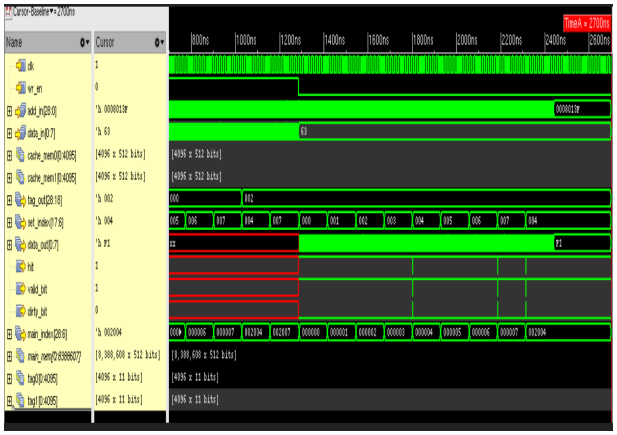

Fig.11: Two Way Cache Mapping.

In two way mapping technique, when cache miss occurs CPU accesses data from main memory shown in Fig.12.

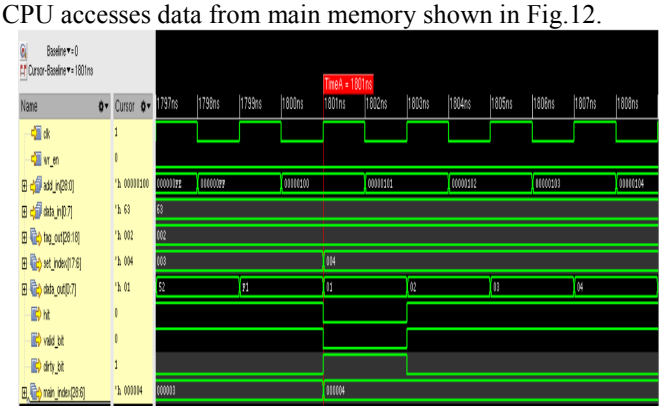

Fig.12: 2 way Mapping Cache Miss.

In miss condition cache memory will update with new main memory block as shown in Fig.13.

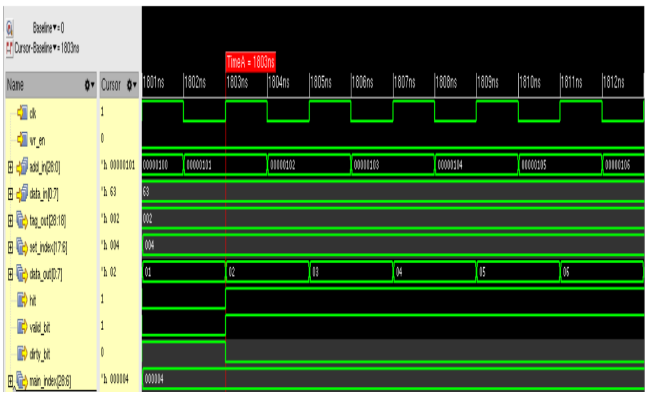

Fig.13: 2 way Mapping Cache Hit.

Four Way Cache Mapping: In Fig.14 the four separate cache memory uses of each cache set of size 128Kbyte is simulated.

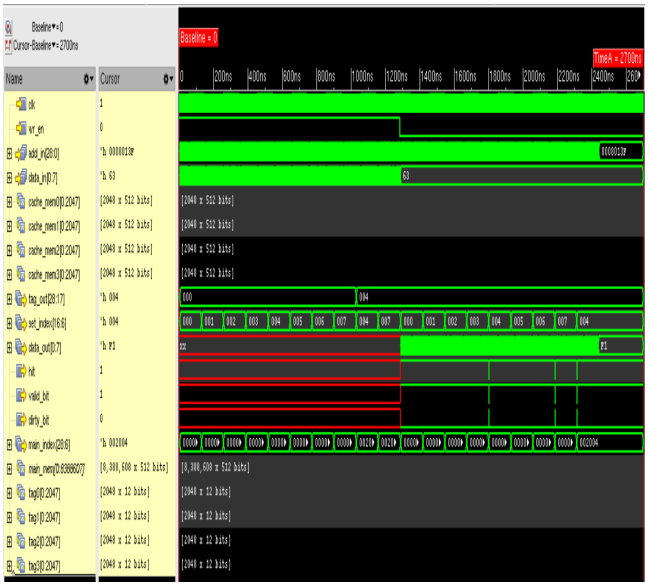

Fig.14: Four Way Cache Mapping.

Eight Way Cache Mapping: Fig.15 shows eight separate cache memory having each cache set size of 64 Kbyte.

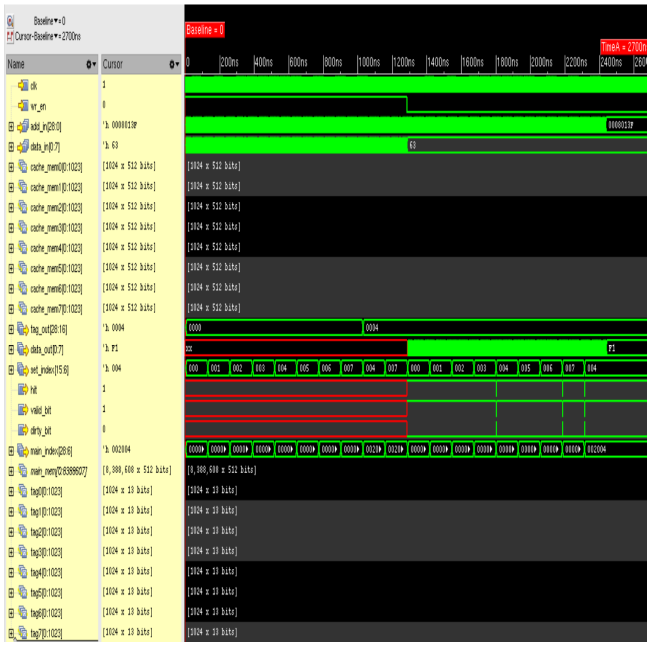

Fig.15: Eight Way Cache Mapping.

V. CONCLUSION

In this paper, we have reconfigured Cache memory according to user requirement. The 512 Mb Main memory is mapped to Cache memory in direct mapping [512-Kbyte], in two way set associative mapping 512 Kbyte is divided into two equal cache sets of 256 Kbyte [2*256-Kbyte]. Similarly, 4-way [4*128-Kbyte], and 8-way [8*64-Kbyte] mapping. Main advantage of set associative is that only one Cache-set is enabled and other Cache-sets are disabled. So that, the accessing speed of the data is fast and keep the unused Cache memory set blocks in stand-by mode to optimize power.

VI. FUTURE SCOPE

In this paper designed 8-bit (1-byte) data accessing. In future implement 64-bit (8-byte) data accessing along with reconfigurable 512-Kbyte cache memory using Verilog HDL. So that it can access more data at each clock cycle.

REFERENCES

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

[2] Mostafa Abd-El-Barr and Hesham El-Rewini, Fundamentals of Computer organization and architecture. New Jersey, USA: John Wiley & Sons, 2005.

[3] Steve Furber ‘’ARM System –On-Chip Architecture”, Addison Wesley, 2nd edition. [Text book page no.272-279].

[4] Safaa S. Omran and Ibrahim A. Amory, "Implementation of Reconfigurable Cache Memory using FPGA". First Engineering Conference for Graduate Research, Technical College, Baghdad, Iraq, 2016.

[5] Sudha, S. “An architecture for victim cache”, 2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), 2016.

[6] M. Tariq Banday, Munis Khan. “A study of Recent Advances in Cache Memories”, 2014 International Conference on Contemporary Computing and Informatics (IC3I), IEEE, 2014.

[7] Swadhesh Kumar, “An Overview of Modern Cache Memory and Performance Analysis of Replacement Policies”, 2nd IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, TN, India, March 2016.

[8] David A. Patterson, John L. Hennessy. "Computer organization and design: the hardware/software interface". 2009.

[9] N. Beckmann and D. Sanchez, “Jigsaw: scalable software-defined caches”, Proc. 22nd Conf. Parallel architectures and compilation techniques (PACT 2013), 2013.

[10] B.M. Lee and G.H. Park, "Performance and energy-efficiency analysis of hybrid cache memory based on SRAM," Proc. Int’l Conf. SoC Design (ISOCC 2012), pp. 247-250, Nov. 2012.