Mutilevel Cache Controller

*

Jeeshma t.n
CB.EN.P2VLD22019
Dept. of Electronics and Communication Engineering
Amrita School of Engineering
Amrita Vishwa Vidyapeetham, Coimbatore

***Abstract*—This paper presents the implementation of a Cache controller using FPGA(Field-Programmable Gate Array). There are L1 and L2 caches here. The L1 Cache Memory is directly mapped and the mapping used for L2 Cache is four-way set associative mapping The policies used for L2 cache are writeback,write-no-allocate, and least recently used. The implementation is done using Verilog HDL and This design is synthesized and simulated using Xilinx vivadio.**

***Index Terms*—ache Controller, Cache memory, Mapping,read, write**

I. Introduction

To improve the performance of the system and to reduce the latency of memory access cache memory plays an important role in modern computer architecture. for temporarily storing frequently accessed data and instructions it provides an efficient mechanism. it acts as a bridge between the processor and the memory. The block diagram of cache memory is given below.

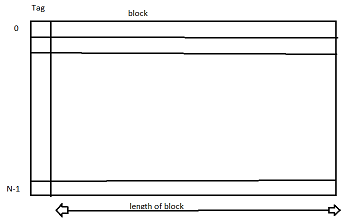

Fig. 1. Cache memory

There are different types of mapping techniques in cache memory [1]. It highlights the importance of performance and simplicity. There are policies for read and write [2]. Also the data movement between policies. There are different levels for a cache memory as per our specifications [3]. Based on requirements we are adapting the size of cache memory. By improving the cache hit rate we can improve the performance [4]. we can also implement these systems on FPGA due to their flexibility [5]

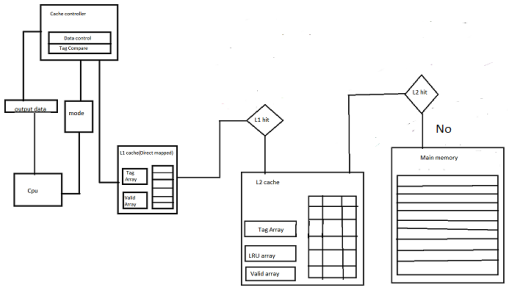

Fig. 2. Architecture .

II. Architecture

A. Cache Controller

The cache controller block determines whether the operation is read or write. There is a wait signal. This block verifies it is not busy executing the previous instruction with the help of a wait signal before accepting input. For a read operation, it will do the following steps. for the requested data the cache controller block first searches in the L1 cache, if found then it will give an L1 hit signal as 1, if not found then it will continue its search in L2 cache, if it not found in L2 cache then it will continue its search in main memory. retrieving data from the L1 cache will cause a 2-cycle delay. For locating the data in the L2 cache, it will promote to the l2 cache. To make space for new data there will be eviction for the existing block. Like this eviction and promotion will be performed in L2 cache and main memory. In a write operation, a similar operation will be performed but there will be no eviction and promotion as per the write no allocate policy.

B. Direct mapping

In this mapping, the memory block is mapped to one of the cache lines In the memory address there are index bits. this mapping is determined from there.

address = 32'b00000000000000000000000000000001;
data = 8'b10110101;
mode = 1'b1;

Identify applicable funding agency here. If none, delete this.

it is a 32-bit address. in this block index is 0 and the block offset is 1. The data is written to the address. This portion of code corresponds to direct mapping since the block index is zero here. this block will not be able to be copied to the l1 or l2 cache because it is indicating write no allocate policy here which is direct mapping typically.

C. Set-associative mapping

in this mapping, since it is a 4-way set associative mapping, each set in the cache can store multiple blocks of memory.in memory address, there are index bits. This mapping can be determined by those bits.

address = 32'b00000000000000000000000100000000;

data = 8'b10110111;

mode = 1'b0;

It is a 32-bit address. in that, there are 64 block index bits and zero block offset bits. since it is a read operation the data doesn't matter here. after this read operation, the data from this block will be promoted to the l1 or l2 cache which clearly explains the write-back policy. It indicates a set associative mapping because the block is present in both l1 and l2 cache.

III. Design

A. Design calculations

The size of the controller blocks the first cache and second cache and the size of the main memory is necessary for design calculations. According to the specifications, using the equations we are calculating the size of each block. The size of l1 is 64. here the indexing bits required are 6.2 offset bits are required here. The tag bits required are 24. For the second cache there are 4 ways in that and the lines in that block are 128 the no of bits required for indexing is 7 here. the no of bits required for offsetting is 2 here. and 23 tag bits are required here. latency is different for l1,l2, and main memory. it will come in the following like 1,3,10.

B. Timing Diagram

when the mode is one that is in a write operation, for a 32-bit address, the block index is 0, block offset is 2, and data is given, then initially block 0 will be in main memory due to the write-no-allocate policy, this block will not be copied to L1 or L2 caches. when a mode is zero, that is a read operation, since it is a read operation the data doesn't matter here, for a 32-bit address block index is 64 and the block offset is zero, the block will be in the main memory initially, because of write back policy it will be promoted to l2 then to l1 cache and this block now will be found in l1 cache. Again write operation is performed, at the block index is 64 block offset is 1, and data is written at the given address, earlier block 64 was promoted to the l1 cache so now this block will be in the main memory. again when a read operation is performed data doesn't matter here as it is a read operation, at block index 0 and block offset 2, block 0 was in main memory, and it will be promoted to the l1 cache after this operation, this block takes the same position as block 64, block 64 will be evicted from l1, it will be now found in l2 cache.

IV. Working

The first mode is selected, after selecting the mode address is given and input data is given, giving both controller blocks to continue their search in the first cache, if the data is in the first cache then it will give an l1 hit as one if data is not there then the controller continues its search in the second cache if the data is there then give hit signal as 1 if not there then continues its search in main memory .when it is searching in first cache 2-sec delay is implemented here. when it is searched in a second cache 3 second delay is implemented here. when it is searching in main memory the delay will be more.

V. Results

A. Vivado Simulation and FPGA implementation

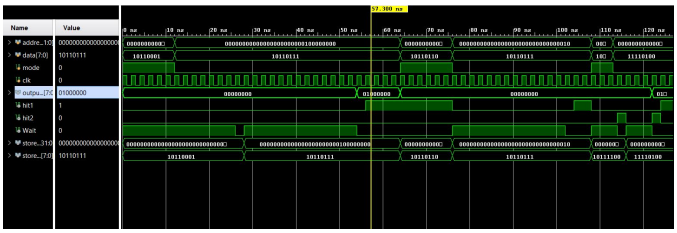

Fig. 3. mode=0,hit1=1

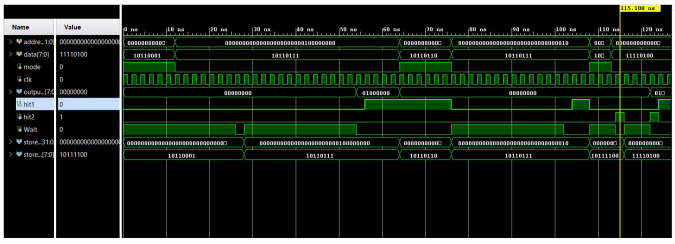

Fig. 4. mode=0,hit2=1

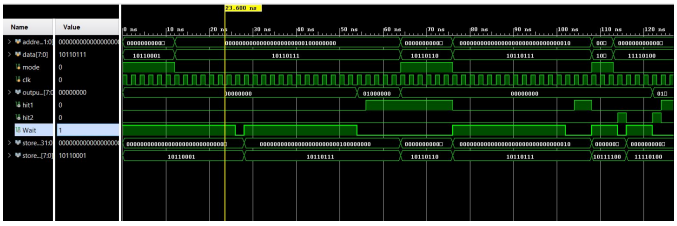

Fig. 5. mode=0,wait=1

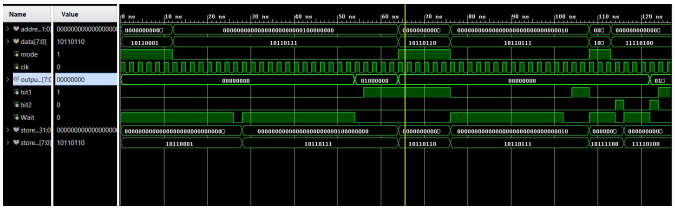

Fig. 6. mode=1,hit1=1

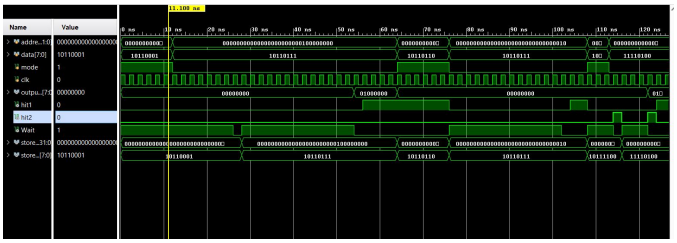

Fig. 7. mode=1,wait=1

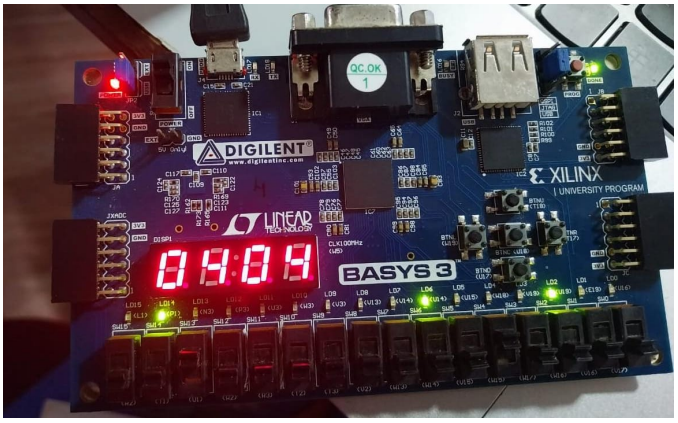

Fig. 8. FPGA implementation

VI. Conclusion

Here I have implemented the design and implementation of a cache controller for l1 and l2 cache using Verilog hdl and simulated and synthesized in Xilinx Vivado. Here I have implemented direct mapping for l1 cache and set associative mapping for l2 cache with write back ,write no allocate and LRU policy.This design was implemented on a Basys3 board.

References

[1] G. Kaur, R. Arora and S. S. Panchal, "Implementation and Comparison of Direct mapped and 4-way Set Associative mapped Cache Controller in VHDL," 2021 8th International Conference on Signal Processing and Integrated Networks (SPIN), Noida, India, 2021, pp. 1018-1023, doi: 10.1109/SPIN52536.2021.9566081.

[2] B. S. Vamsi and M. A. Basiri M, "High Performance Read and Write Policy of Multilevel Cache Memory," 2022 IEEE 6th Conference on Information and Communication Technology (CICT), Gwalior, India, 2022, pp. 1-5, doi: 10.1109/CICT56698.2022.9997938.

[3] A. S. Kushchenko, N. E. Ternovoy, M. G. Popov and A. N. Yakunin, "L1 Cache with Dynamic Coverage of an External Memory," 2022 Conference of Russian Young Researchers in Electrical and Electronic Engineering (ElConRus), Saint Petersburg, Russian Federation, 2022, pp. 365-367, doi: 10.1109/ElConRus54750.2022.9755627.

[4] I. Lokegaonkar, D. Nair and V. Kulkarni, "Enhancement of Cache Memory Performance," 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N), Greater Noida, India, 2021, pp. 1490-1492, doi: 10.1109/ICAC3N53548.2021.9725639.

[5] M. K. N, Kanagasabapati and S. S. Yellampalli, "Design of Reconfigurable Cache Memory Using Verilog HDL," 2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), Msyuru, India, 2018, pp. 1171-1176, doi: 10.1109/ICEECCOT43722.2018.9001646.