Advance in Electronic and Electric Engineering.
ISSN 2231-1297, Volume 3, Number 8 (2013), pp. 977-982

http://www.ripublication.com/aeee.htm

Design of a Two-Way Set-Associative Cache

[1]Karthik Ghanta and [2]Srikanth Parikibandla

Sreenidhi Institute of Science & Technology, Hyderabad, Andhra Pradesh, INDIA.

Abstract

The principle of locality applies at many levels of memory, and taking advantage of this locality improves performance in the hierarchy. Based on the same principle, cache is a small but very fast memory held near to the processor. So a block, multiple-words of memory, of recently used memory values was moved into it for this performance enhancement. In addition, to examine which address the block corresponds to, a tag is also included. Placement of this block in the cache is important and among such the most popular scheme is set associativity. The cache placement referred as n-way set associative if it has n blocks of sets. Absence of required copy of memory, a cache miss, needs to make a transfer from its lower level. Thus, the interface of the cache with its slave memory, a lower level memory, is also critical.

For customized embedded interfaces ARM has released a bus specification AMBA advanced extensible interface, AXI4. This interface is also a technology independent methodology helpful in the design, implementation and test of highly integrated modular interfacing. Features for faster sub-micron interconnect mainly include: only start address issued transactions, byte strobes for unaligned data transfers, dedicated control, address and data interfacing signals.

This work aims in developing such 2-way set associative cache of 32kB size with 64-bit lines. Data for cache, to be read or written, assumed to be given by the master and, read or written to a particular address location of slave using AMBA AXI protocol. In this work cache and slave are modeled in Verilog.

Keywords: Cache, set- associative cache, AMBA, AXI, Verilog.

1. Introduction

With the increase in technological advancements the performance of a processor is steadily increasing. Though similar advancements in memory storage technologies are available, but are still far behind with the performance rates compared to processors. Arrangement of entire memory with such comparable speeds escalates the design cost to a large amount. Also urge for higher bit-width computations is also on demand. This led to deployment of faster but small-scale memory onto the processor core along with the regular optimum speed mammoth-sized main memory [2][3].

This memory, called as Cache, is small but very fast and is held near to the processor. A piece of information in cache is accessed based on its location and this becomes the base for it to hold the data. Recently used bits or a chunk of required amount of bits for further operation are stored here. In addition, to examine which address these chunk of bits corresponds to, a portion of the actual address, known as tag, is also held in the cache. These tag bits are compared with the incoming address of the required content address, if success, known as hit, indicating availability of data with cache and so a faster transaction is made. Otherwise, known as miss, transfer of bits from the main memory is to be made leading to the additional cost of tag-compare time than the usual access time of main memory [4]. To reduce this additional delay or reducing the miss rates associativity of the cache memory can be used. However the higher associativity results in higher cost and large die-space [2], a proper choice is thus to be entertained as per the need.

Not just providing the additional piece of memory, the availability of better interface with the processor is also desired [1]. Considering this ARM has released a bus specification AMBA AXI4, AMBA advanced extensible interface. This interface is also a technology independent methodology helpful in the design, implementation and test of highly integrated modular interfacing. The work done in this paper follows with the introduction to set-associative cache, AMBA AXI, proposed work and its corresponding simulation results.

2. Set Associative Cache

Holding the same index and different tags, Set-associative mapping allows a limited number of blocks in the cache. This division of cache into parts is referred to as "sets" of blocks. The number of blocks in a set is known as the **associativity** or **set** size. Each block in each set has a stored tag which, together with the index, completes the identification of the block. A 2-way set-associative cache is illustrated in Figure 1.

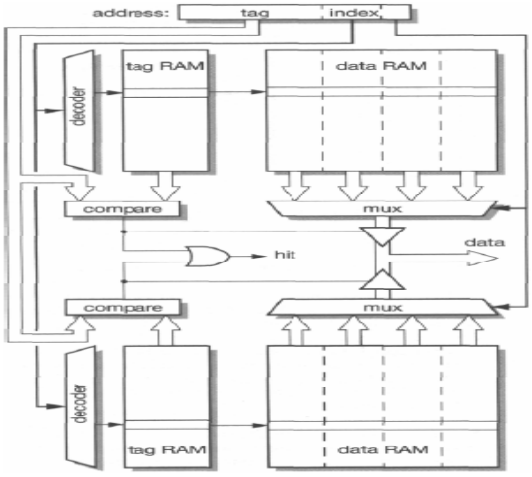

Figure 1: Two-way set-associative cache organization.

The set-associative approach extends beyond 2-way up to any degree of associativity, but in practice the benefits of going beyond 4-way associativity are small and do not warrant the extra complexity incurred [8] [10].

3. AMBA AXI Protocol

This Advanced eXtensible Interface AXI is a technology independent methodology helpful in the design, implementation and test of highly integrated modular interfacing [7]. Features for faster sub-micron interconnect mainly include: only start address issued transactions, byte strobes for unaligned data transfers, dedicated control, address and data interfacing signals [6].

To achieve all these features AMBA makes use of 5 channels [9]. Each of these five independent channels consists of a set of information signals and uses a two-way **VALID** and **READY** handshake mechanism.

The information source uses the **VALID** signal to show when valid data or control information is available on the channel. The destination uses the **READY** signal to show when it can accept the data. Both the read data channel and the write data channel also include a **LAST** signal to indicate when the transfer of the final data item within a transaction takes place.

3.1 Read and write address channels

Read and write transactions each have their own address channel. The appropriate address channel carries all of the required address and control information for a transaction.

3.2 Read data channel

The read data information and any read response information is given by save to the master. The read data channel includes the data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide, the completion status indication of the read transaction.

3.3 Write data channel

The write data channel conveys the write data from the master to the slave and includes the data bus, which can be 8, 16, 32, 64, 128, 256, 512, or 1024 bits wide, indication for the corresponding byte of data bus byte lane strobe for every eight data bits. To perform operation without slave acknowledgement for prior write the write data channel information is always treated as buffered.

3.4 Write response channel

The write transactions response is provided for the slave using this channel. Instead of sending completion signal for every data transfer, signal after each burst is used.

4. Proposed Work

The cache memory that is to be designed under this project is a 32kB size two-way set-associative cache memory. The three portions of the cache address are shown below in Figure 9, namely Tag, Index and Byte Offset.

In this design, as of present day requirement, a 64-bit memory is to be considered, with the processor considered to be 32-bit. This requires accessing of 8 8-bit (Byte) memory locations for an address specified by the 32-bit address. In order to see the functionality of the cache, a slave memory is also designed to interface along with the cache.

The 32-bit address is split in the following way to have the organization of the cache memory:

- To locate one of the 8 Bytes of memory 3 memory bits are to be considered, namely B2-B0.
- And in order to address a 32KB memory a total of 4k locations are needed with each location capable of holding 8 Bytes. These 4k (2^2x2^{10}=2^{12}) locations need 12 bits to hold the value and so using B14-B3 of the address bits for that purpose.
- The remaining 17 bits i.e. B31-B15, along with a valid bit, act as tag lines for the memory locations.

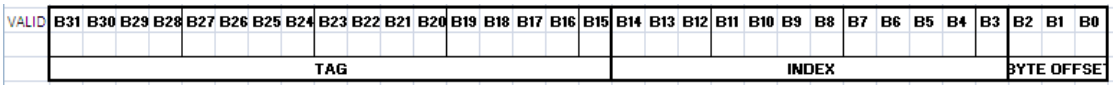

Figure 9: The three portions of an address in the proposed cache.

5. Simulation

In order to see the functionality of this cache, a slave memory is also designed to interface along with the cache. This is interfaced to cache using the above mentioned AXI protocol. This work is done using Verilog [10] HDL language in Xilinx Plan Ahead software [11].

A. Simulation for a cache-miss

An input for the data addressed at a location `0057ea10`h is accessed which is not residing in the cache. This makes the slave to be accessed which can be seen in corresponding signals like araddr, arlen, etc., are obtained accordingly as 0057ea00h, fh respectively.

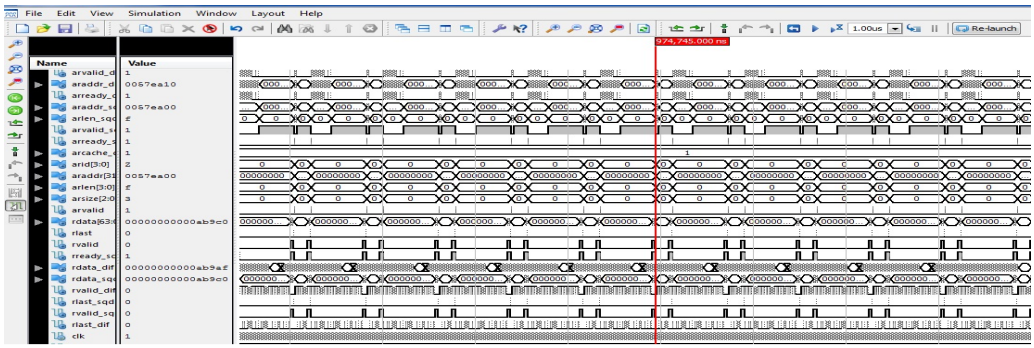

Figure Simulation showing a cache-miss access.

B. Simulation for a cache-hit

An input for the data addressed at a location `00578f60`h is accessed which is already resided in the cache. This makes the slave not to be accessed which can be seen in corresponding signals like araddr, arlen, etc., are made 0's.

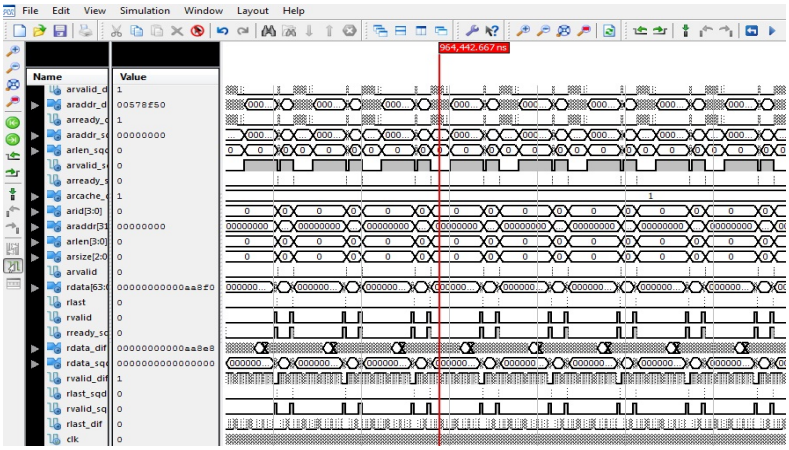

Figure Simulation showing a hit access.

6. Conclusion

This high speed data transfer interfacing helps in better way of transaction. Like reduction in completion signals by sending them at the end of the burst transfer instead of having after every byte. This design can be used to cater the needs of lower level of caches, L2 and subsequent levels having interface with the main memory. However this is achieved by an additional burden of few signals than regular interface signals.

References

[1] Math, Shaila S., and R. B. Manjula, "Survey of system on chip buses based on industry standards," in conference on Evolutionary Trends in Information Technology (CETIT), Bekgaum, Karnataka, India, p. 52. 2011.

[2] M. Powell, A. Agrawal, T. N. Vijaykumar, B. Falsafi, and K. Roy, “Reducing set-associative cache energy via selective direct-mapping and way prediction”, In Proc. of 34th Annual Int’l Symp. on Microarchitecture (MICRO), Dec. 2001, pp. 54-65.

[3] Min, Rui, Wen-Ben Jone, and Yiming Hu. "Location cache: a low-power L2 cache system." In Low Power Electronics and Design, 2004. ISLPED'04. Proceedings of the 2004 International Symposium on, pp. 120-125. IEEE, 2004.

[4] John L Henessey, David A Petterson, “Computer Architecture: A Qualitative Approach”, Fourth Edition, Elsevier, MK Publishers.

[5] Chang, J. H., H. Chao, and Kimming So. "Cache design of a sub-micron CMOS system/370." In Proceedings of the 14th annual international symposium on Computer architecture, pp. 208-213. ACM, 1987.

[6] M. Siva Prasad Reddy, B. Babu Rajesh, TVS Gowtham Prasad, “A Synthesizable Design of AMBA-AXI Protocol for SoC Integration,” IJEI Journal, vol 1 issue 3, pp: 19-26.

[7] Chien-Hung Chen, Jiun-Cheng Ju, and Ing-Jer Huang, “A Synthesizable AXI Protocol Checker for SoC Integration”, IEEE transl, ISOCC, Vol 8, pp.103-106, 2010

[8] C.-L. Su and A. M. Despain, “Cache design trade-offs for power and performance optimization: A case study”, In Proc. of 1995 In’l Symp. on Low Power Electronics and Design (ISLPED), 1995, pp. 63-68.

[9] ARM, AMBA. "AMBA AXI and ACE Protocol Specification." (2013).

[10] Samir Palnitkar, Verilog HDL: A Guide to Digital Design and synthesis, 2nd ed, Prentice Hall PTR Pub, 2003

[11] PlanAhead Design and Analysis Tool -User Guide, www.xilinx.com