Design and characterization of a Fully Associative Cache Controller IP core

Deepa C, Nandakumar R.

Abstract— Cache memory is a small, but fast memory that is used to store data that have been accessed recently and are likely to be referenced again soon in the future. Cache controller responds to the read and write requests issued by the processor. In this paper an architecture design of Cache controller core for fully set associative cache with write-through policy is described. This paper proposes to address detailed investigations about the different categories of cache memory, deriving a suitable behavioral model for custom controller, for targeting a cache memory.

Index Terms— Cache controller, CAM, Finite State Machine, Fully set associative cache, IP core, RTL, Verilog.

1 INTRODUCTION

PRESENTLY the speed of the memories is not able to cope up with the speed of the processors. As a result of this a cache memory becomes an integral part of the memory hierarchy. Cache is a small, fast array of memory placed between the processor and main memory. Cache consists of a fast memory that is typically made from SRAM and a controller. This controller section of the cache is responsible for performing all the logical operations of the cache. Cache controller is the brain behind cache. Internally it has a finite state machine that generates the control signals needed for the operation of cache memory and main memory.

The proposed controller is targeted for a fully associative cache memory having a cache size of 512 KB. Both the address and data widths are 32 bit. The address issued by the processor is of 32 bits. The write policy used is write through, i.e., data is written synchronously to the cache memory as well as to the main memory. On write misses, write no allocate policy is used, i.e., data is written to the main memory only. On read misses data is written to the cache. Cyclic replacement policy is used. Section 2 deals with the architecture of the proposed controller; the principle of operation of the controller is covered in section 3; simulation results are shown in section 4 and synthesis results are given in section 5. Hardware test results are given in section 6. Section 7 gives the conclusion.

2 PROPOSED CONTROLLER ARCHITECTURE

2.1 Introduction

The proposed cache controller is designed to work with custom fully set associative cache memory. It has a host interface on one side and cache memory and main memory on the other side. The controller consists of control logic, built in replacement block, CAM and an encoder.

2.2 Block Diagram

The block diagram of proposed Cache controller is shown in Figure 1.

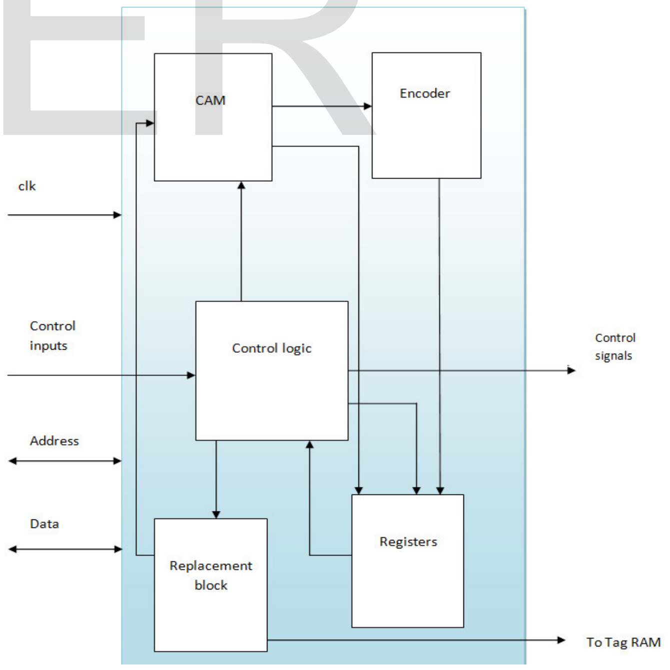

Figure 1: Block diagram of cache controller

2.3 Input-Output Description

The input output diagram of controller is shown in Figure 2. The output signals from the cache controller are given as input to the cache memory and main memory. All the pins in the pin diagram are described in Table 1. Direction and description of the pins are also given.

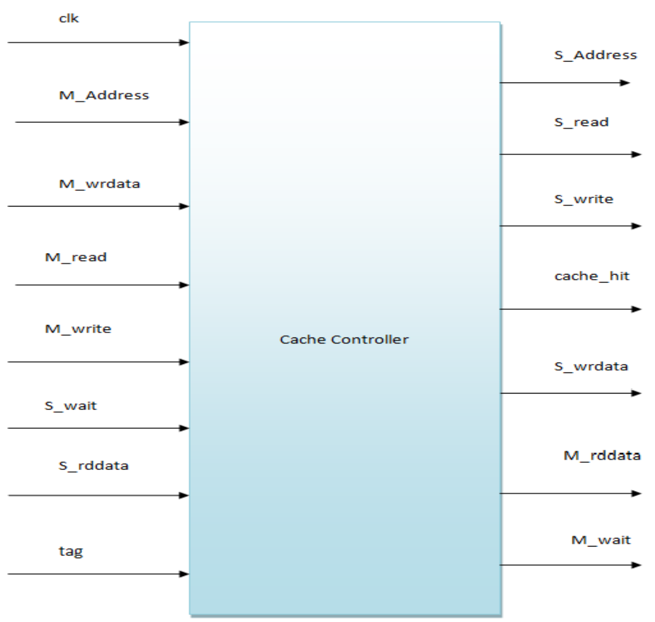

Figure 2: Input output diagram of cache controller

PIN	DIRECTION	DESCRIPTION
clk	Input	Input clock to the controller
M_Address	Input	Address issued by the processor
M_wrdata	Input	Data issued by the processor during write operation
M_write	Input	Input to the controller to perform write operation
M_read	Input	Input to the controller to perform read operation
S_rddata	Input	Data that is read from the slave
S_wait	Input	Signal to the controller indicating that the slave is busy
tag	Input	Tag address given to the cache controller
S_wrdata	Output	Data that is written to the slave
M_wait	Output	Signal indicating that the master is busy
cache_hit	Output	Signal indicating whether the access is a hit or miss
S_read	Output	Signal to the slave indicating read operation
S_write	Output	Signal to the slave indicating write operation
M_rddata	Output	Data that is given to the processor after read operation
S_Address	Output	Address issued to the slave

Table 1: Pin description of cache controller

2.4 Architecture Detailed

Controller has a host interface on one side and cache memory and main memory on the other side. The controller architecture consists of Control logic, built in replacement block, CAM and an encoder. Detailed explanations of each of the blocks are given below.

1. Control logic:

Control logic generates the control signals needed to the memory, registers, CAM and replacement block. The control signals generated by the control unit are S_write and S_read and M_wait. S_write is the control signal given to the slave unit indicating slave write and S_read is the signal indicating Slave read. M_wait indicates that the master unit is busy.

2. Replacement Block:

Replacement policy used in the controller is counter based replacement policy. The counter is implemented using the built in lpm_counter megafunction. The counter based replacement policy provides a simple and effective way to select data to be evicted from the cache. The counter always points at the next value, to be evicted in the cache. The counter increments with each new value cached. The values that enter the cache least recently are evicted first. This policy does not take into account how many times a piece of data was accessed. However it requires simpler circuitry, which reduces silicon penalty.

3. Encoder:

The encoder is the largest block of logic and is within the system's critical path. The size of the encoder is determined by the depth of the cache, Increases in cache depth result in a longer critical path and smaller fmax. So an efficient implementation of encoder is required to overcome this reduction in fmax.

4. CAM:

A cache must search through a number of tags so as to find a match this process of searching through the tags can be done by using a Content Addressable Memory (CAM). A CAM is the inverse of RAM in the sense, while RAM is given an address as input and outputs the data stored at that address, a CAM receives data and returns the address where the data is stored, or indicates that the data is not currently in the CAM. This makes CAMs ideal for searching through tags and detecting cache hits. The pattern given to the CAM is tag and the address returned by the CAM is the position of data in the Data RAM.

5. Registers:

Registers are implemented using the built in flip flops of the memory blocks. The altsyncram megafunction enables users to specify whether these registers should be used or not. If the registers are not needed they can be bypassed. This optimization is to provide a latency of two or three cycles.

3 PRINCIPLE OF OPERATION

The different operations of the memory are Read operation and Write operation. The different stages of the cache controller for read and write operation can be explained by a Finite State Machine (FSM). FSM for read operation and write operation are shown in Figure 3 and Figure 4 respectively.

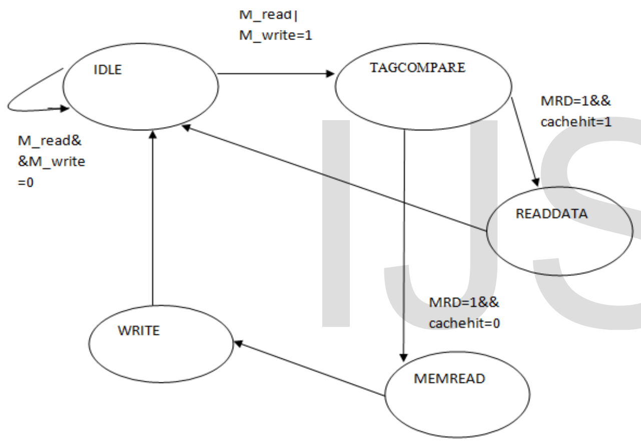

Figure 3: FSM for read operation

When both M_read and M_write inputs are active low, controller remains in the IDLE state, i.e., there is no memory access, underway. When M_read or M_write is high, controller interprets it as a valid request. Then the control moves to the TAGCOMPARE state to see if the tags match. If the tags match i.e., it is a cache hit and M_read is high, control moves to the READDATA state. In the READDATA state data is read from the cache memory. After completion of cache memory read control moves to the IDLE state.

If the tags don't match and M_read is high, the control moves to the MEMREAD state to access main memory. During this state data is read from the main memory. After completion of main memory read, control moves to the WRITE state. In this state data is written to the cache memory. Data read from the main memory is provided to the processor and control moves to the IDLE state.

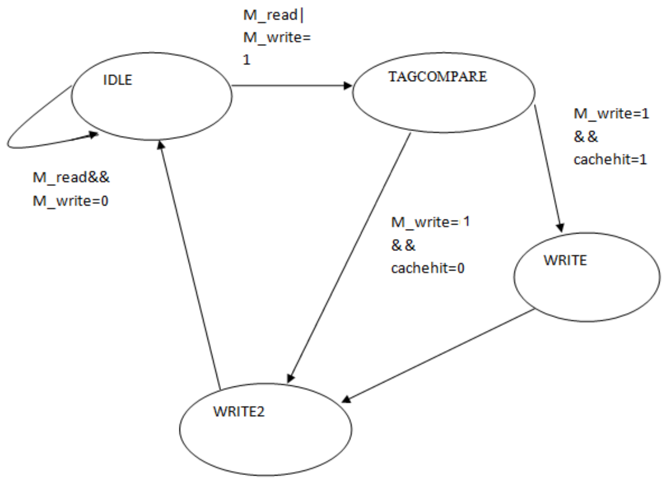

Figure 4: FSM for write operation

When both M_read and M_write inputs are active low, controller remains in the IDLE state, i.e., there is no memory access, underway. When M_read or M_write is high, controller interprets it as a valid request. Then the control moves to the TAGCOMPARE state to see if the tags match. If the tags match i.e., it is a cache hit and M_write is high, control moves to the WRITE state. In the WRITE state data is written to the cache memory. After completion of cache memory write control moves to the WRITE2 state, i.e., data is written to the main memory as well. After completion of main memory write control moves to the IDLE state.

If the tags don't match and M_write is high, the control moves to the WRITE2 state to initiate write-through to main memory. During this state data is read from the main memory.

4 SIMULATION RESULTS

Simulation of cache controller IP core was done using ModelSim SE®. Simulation results for read operation are shown in Figure 5 and 6.

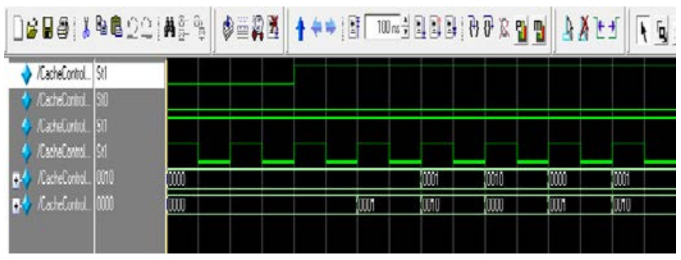

Figure 5: Read operation (Hit)

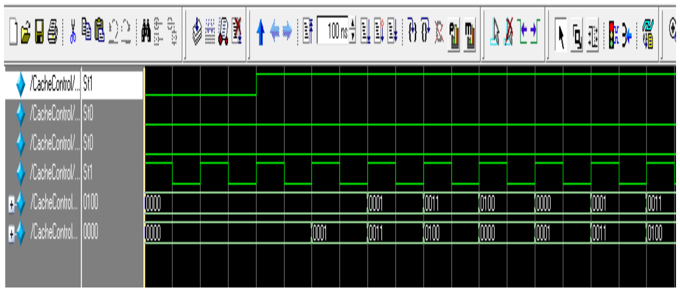

Figure 6: Read operation (Miss)

Simulation results for write operation are shown in Figure 7 and 8.

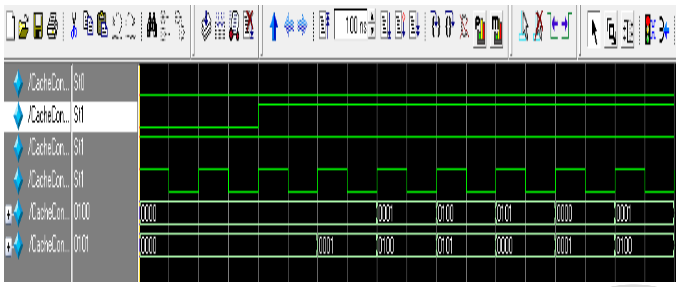

Figure 7: Write operation (Hit)

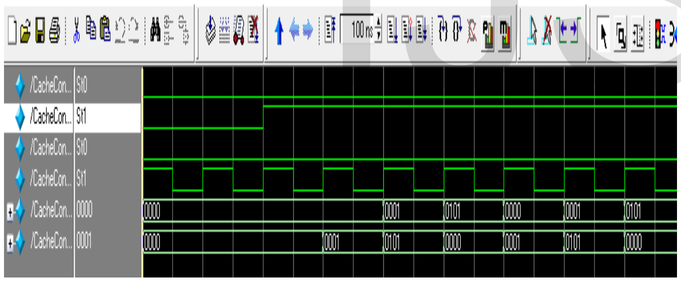

Figure 8: Write operation (Miss)

5 SYNTHESIS RESULTS

Design was synthesized by using Altera Quartus® II Design suite.

5.1 Resource Utilization Report:

Resource utilization report is given in Table 2.

FAMILY	Cyclone II		
DEVICE	EP2C20F484C7		
Resource	**Used**	**Available**	**Utilization**
Total logic elements	1868	18752	10%
Total combination functions	766	18752	4%
Dedicated logic registers	1591	18752	8%
Total registers	1591	18752	8%
Total memory bits	14966	239616	6%
Embedded multiplier 9-bit elements	0	52	0%
Total PLLs	0	4	0%

Table 2: Resource utilization report

5.2 Power Analysis Report:

Quartus II PowerPlay Power Analyzer® was used to perform power analysis.

Total Thermal Power Dissipation	79.19 mW
Core Dynamic Thermal Power Dissipation	0.00 mW
Core Static Thermal Power Dissipation	47.37 mW
I/O Thermal Power Dissipation	31.81 mW

Block type	Total thermal power	Block thermal dynamic power	Block thermal static power
JTAG	0.00 mW	0.00 mW	-
I/O	12.87 mW	0.00 mW	12.87 mW
M4K	0.00 mW	0.00 mW	-
Combinational cell	0.00 mW	0.00 mW	-
Register cell	0.00 mW	0.00 mW	-
Clock control block	0.00 mW	0.00 mW	-

Table 3: Power analysis report

6 HARDWARE TEST RESULTS

The SignalTap II ® Logic Analyzer was used for on chip debugging of the IP core. Signaltap II ® Logic Analyzer captures and displays real time signal behavior.

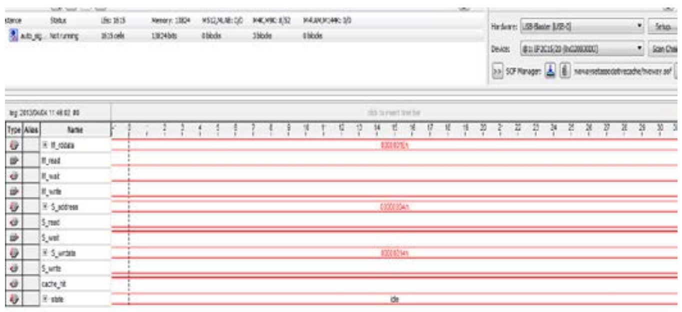

Figure 9: Idle condition

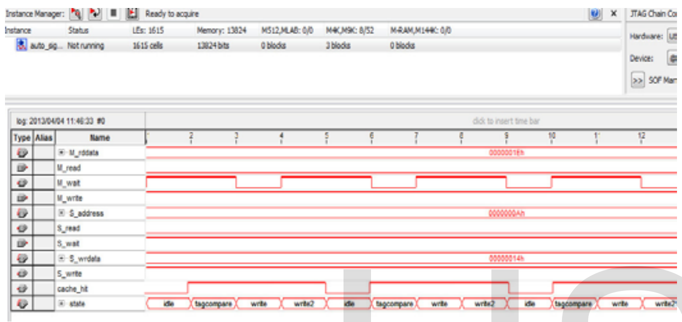

Figure 10: Write operation (Hit)

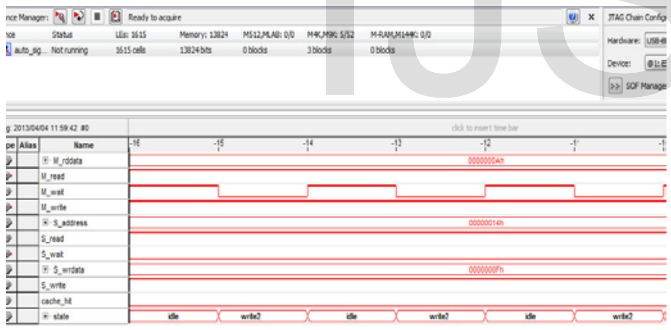

Figure 11: Write operation (Miss)

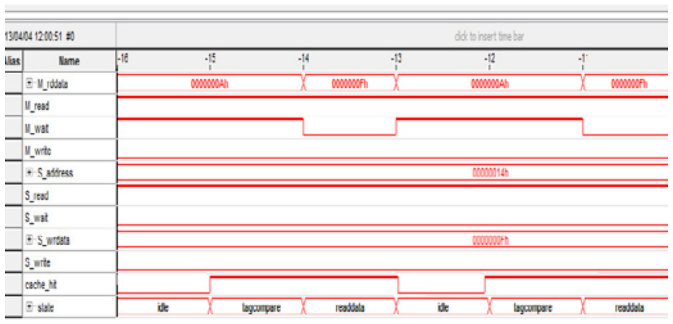

Figure 12: Read operation (Hit)

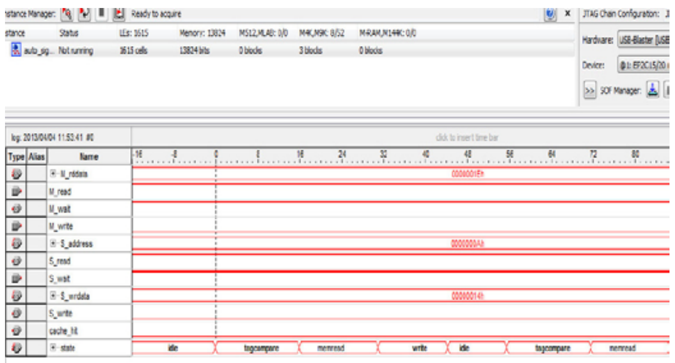

Figure 13: Read operation (Miss)

7 CONCLUSION

The architecture of a custom controller with counter based replacement policy for a fully set associative cache memory was designed, prototyped and characterized for resource utilization and power consumption. The controller serves as a reliable interface to the cache memory as well as to the main memory. The proposed design was tested by implementing the design on Altera® FPGA development board having EP2C20F484C7 FPGA belonging to Cyclone™ family.

REFERENCES

[1] Altera Corporation. "Altera Stratix FPGA Family Data Sheet".December2002,http://www.altera.com/literature/ds/ds_stx.pdf.

[2] John L. Hennessy, Davis A. Patterson. Computer architecture: a quantitative approach (2nd edition), Morgan Kaufmann, January 1996.

[3] Vincent P. Heuring and Harry F. Jordan, Computer Systems Design and Architecture, second edition, Prentice Hall, Upper saddle river, New Jersey, 2004.

[4] A. J. Smith, "Cache memories," Computing Surveys, vol. 14, no. 3, pp. 473-530, 1982.

[5] A. J. Smith, "Cache memory design: An evolving art,"IEEE Spectrum,vol. 24, no. 12, pp. 40-44, Dec. 1987.

[6] J. P. Hayes, Computer Architecture and Organization, 3rd ed., McGraw-Hill Book Company, 1998

[7] J. E. Smith and J. R. Goodman,_Instruction cache replacement polices and organizations,_IEEE Transactions on Computers, Vol. C-34, No. 3, 1985, pp. 234-241.

[8] L. Colagiovanni and A. Shaout, " Cache memory replacement policy for a uniprocessor system, " IEE Electronic Letters, 1990, Vol. 26, No. 8, pp. 509-510.

[9] Vipin S. Bhure , Praveen R. Chakole, "Design of Cache Controller for Multi-core Processor System" international Journal of Electronics and Computer Science Engineering, ISSN: 2277-1956.

[10] Deepa C, Nandakumar R, "Design and characterization of two way set associative cache controller IP core" International Journal of Electronics and Communication Technology, Vol.4 ,Issue2, April-June 2013, pp. 357-359.