Abstract

The purpose of this project was to simulate a simple cache controller and attain practical experience in the design and implementation of custom logic controllers and interfacing to SRAM memory units and other logic devices. The functionality and interactions between the cache controller, SRAM and SDRAM controller was designed and then studied. Another goal of the project was to learn VHDL-coding technique within the Xilinx ISE CAD environment of the system and a hardware evaluation platform based on the Xilinx Spartan-3E FPGA. The cache controller, SRAM memory and SDRAM controller were all implemented using VHDL while the CPU generation VHDL file was provided.

Introduction

The CPU block was pre-built and provided a series of hard-coded outputs. These outputs included, Address[16](made up of Tag[8], Index[3] and Offset[5]), Data[8], and Write/Read[1]. These outputs are summarized below in Table 1.

Table 1: CPU Output Values in Hexadecimal

Address				
Tag	**Index**	**Offset**	**Data**	**Write/Read**
11	0	00	AA	Write
11	0	02	BB	Write
11	0	00	xx	Read
11	0	02	xx	Read
33	2	06	xx	Read
44	2	04	xx	Read
55	0	04	CC	Write
66	0	06	xx	Read

The purpose of these outputs are to simulate Write and Read commands being sent to the cache controller. The CPU block also output a CS[1] bit which was to indicate to the cache controller that it has completed loading its new output values (from table 1) and they are ready to be processed.

A Bram block was used to act as the SRAM Component.

An SDRAM block was created which simulated the Synchronous Dynamic RAM by storing bits in an array of 6 Blocks of 32 Addresses of 8 bits. This array was initialized with values 0x14, 0x8F, 0x67 sequentially throughout.

The cache controller was to be designed as a Finite State Machine which would be able to receive the signal from the CPU, then determine if the requested memory address's block is already in SRAM (hit) or if it is not and needs to be retrieved from SDRAM. If the block to be replaced in SRAM is 'dirty', that is to say it has had*.*. a write operation performed on it, it must first be written back to SDRAM. This cache controller was then evaluated on its performance timings for each type of memory access.

System Specifications

CPU block (Functional Specifications):

1. Simulates Write and Read commands being sent to the cache controller
2. CPU block also output a CS[1] bit which was to indicate to the cache controller that it has completed loading its new output values (from table 1) and they are ready to be processed.
3. Once CS is de-asserted, the CPU block must be ready to repeat the process.

CPU block (Technical Specifications):

1. The CPU is synchronized to the rest of the system via a clock frequency
2. Periodically issues read or write transaction requests
3. When the CPU issues a transaction, it first sets the appropriate address on the address bus and sets the read/write indicator to the correct value
   a) Read : the WR/RD signal is low (0)
   b) Write: the WR/RD signal is high (1), appropriate data is set on the DOUT
4. When all transaction signals are stable, the strobe CS is asserted, and stays asserted for 4 clock cycles

Cache Controller block (Functional Specifications):

1. Receive Write and Read requests from the CPU
2. Determine whether the requested data is currently in SRAM (hit)
3. If necessary, move data from SDRAM to SRAM and vice-versa to maintain data integrity
4. Manage and control block Tag[8], Valid[1] and Dirty[1] bits corresponding to SRAM data
5. Perform read and write operations to SRAM based on CPU requests.

Cache Controller block (Technical Specifications):

1. When CS is asserted by the CPU the Address, Data and WR/RD signals are received by the cache controller from the CPU based on Table 1.
2. The given Address Block's tag and index values are looked for in SRAM.
3. If the block does not exist in SRAM, the dirty bit of the block currently at that index position in SRAM is checked.
4. If the dirty bit is 1, that block is written back to SDRAM; over a period of 64 clock cycles by toggling memstrb every clock cycle and pushing SRAM's Dout to SDRAM's Din. During this process SDRAM's WR/RD (write/read) signal is asserted and an offset value is incremented every 2 clock cycles such that 32 incrementations occur, and 32 sequential address values are passed to both the SDRAM and SRAM.
5. If the dirty bit is 0, or if step 4 has been performed the block is written to SRAM by toggling memstrb every clock cycle for 64 clock cycles and pushing SDRAM's Dout to SRAM's Din. During this process SRAM's Wea (write enable) signal is asserted and an offset value is incremented every 2 clock cycles such that 32 incrementations occur, and 32 sequential address values are passed to both the SDRAM and SRAM.
6. In the case of a write operation, the Wea (write enable) signal of the SRAM is asserted, and Dout, index and offset signals from the CPUare passed to the SRAM Din and Address respectively. The corresponding tag value is updated to match the first 8 bits of the CPU address and the corresponding Valid and Dirty bits are set to '1'
7. In the case of a read operation, the Wea (write enable) signal of the SRAM is set to 0, and Dout, index and offset signals from the CPU are passed to the SRAM Din and Address respectively. The corresponding tag value is updated to match the first 8 bits of the CPU address and the corresponding Valid bit is set to '1'
8. The ready signal is then asserted and sent to the CPU
9. The Cache Controller then remains in a ready state waiting for the CS strobe to be re-asserted by the CPU

Cache SRAM block (Functional Specifications):

1. Acts as local memory for the cache controller

Cache SRAM block (Technical Specifications):

1. All operations issued to the cache controller are synchronized on the rising edge of the clock
2. Write Operation:
   a) Correct address is set on the Address bus by the Cache Controller
   b) WEN signal is asserted (1)
   c) On subsequent rising edge, data is written
3. Read Operation:
   a) Correct address is set on the Address bus by the Cache Controller
   b) addressed data will propagate to the output DOUT after the next rising edge of the clock

SDRAM Controller block (Functional Specifications):

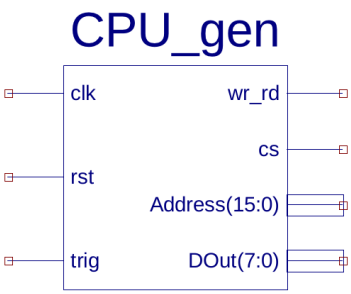
1. Responds to block read and write requests from the Cache Controller
2. Data can be written from the cache to the SDRAM through the Din bus when the WR/RD signal is asserted
3. Data can be read from the SDRAM to the cache through the Dout bus when the WR/RD signal is de-asserted

SDRAM Controller block (Technical Specifications):

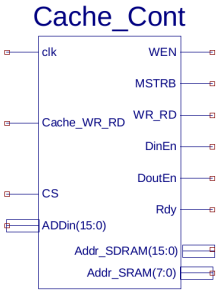
1. Synchronized to Cache and the rest of the system via a clock frequency
2. Write Operation:
   a) Correct base address is set on the ADD port by the Cache Controller
   b) WR/RD signal is asserted (1)
   c) once all other signals are stable (one clock cycle later), the strobe MEMSTRB is asserted for one clock cycle
   d) Process repeated 32 times for 1 full block
3. Read Operation:
   a) Correct base address is set on the ADD port by the Cache Controller
   b) WR/RD signal is de-asserted (0)
   c) once all other signals are stable (one clock cycle later), the strobe MEMSTRB is asserted for one clock cycle
   d) Process repeated 32 times for 1 full block

Device Descriptions

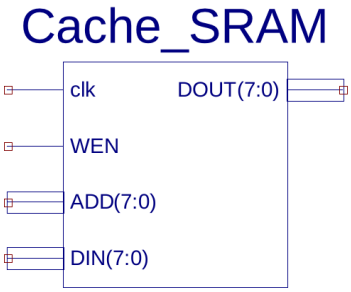
Symbols



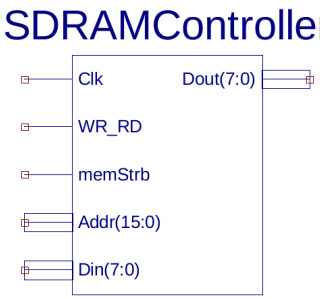
(a) CPU Generator Symbol



(b) Cache Controller Symbol



(c) Cache Static RAM Symbol



(d) Synchronous Dynamic RAM Symbol

Figure 1: Symbol Diagrams for Cache Controller Project

Block Diagrams

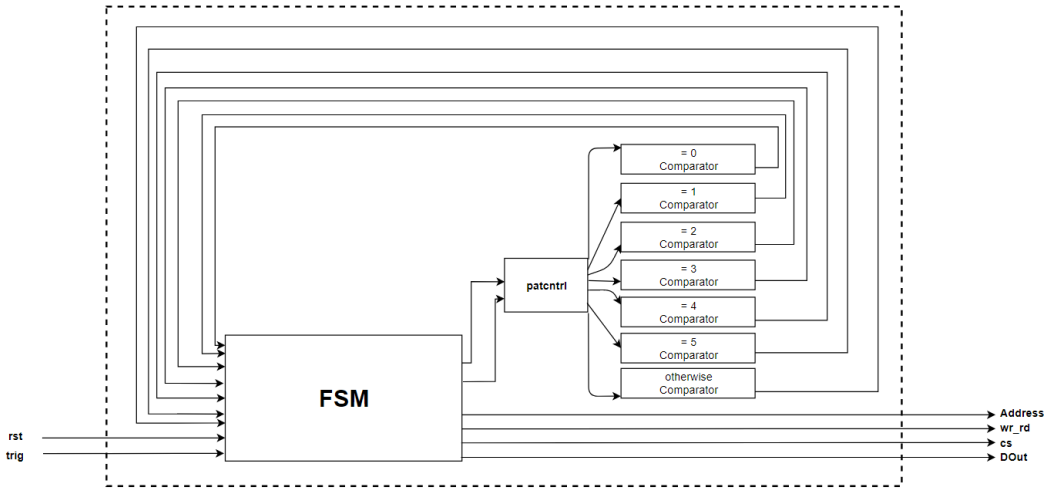


Figure 2: CPU Block Diagram

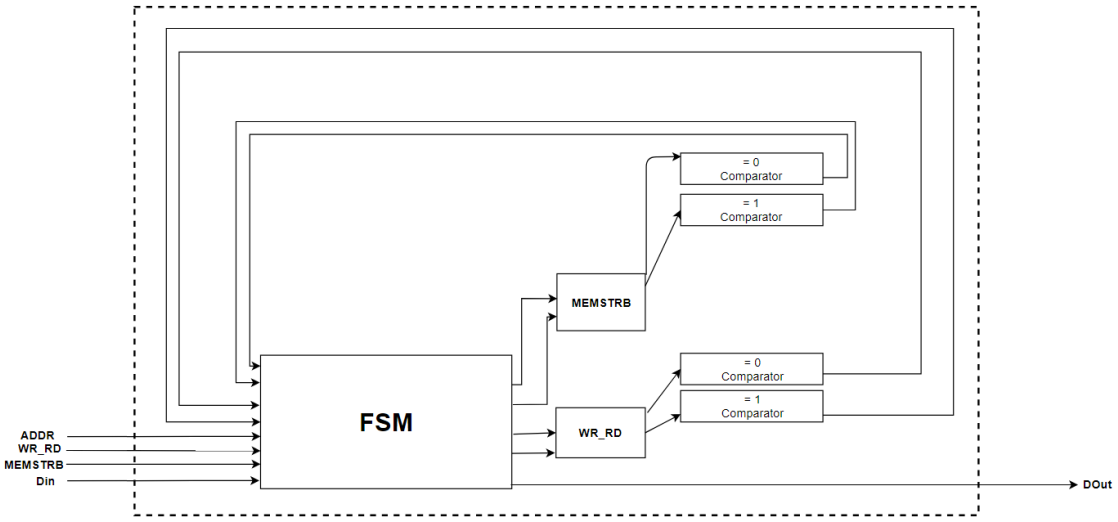


Figure 3: SDRAM Block Diagram

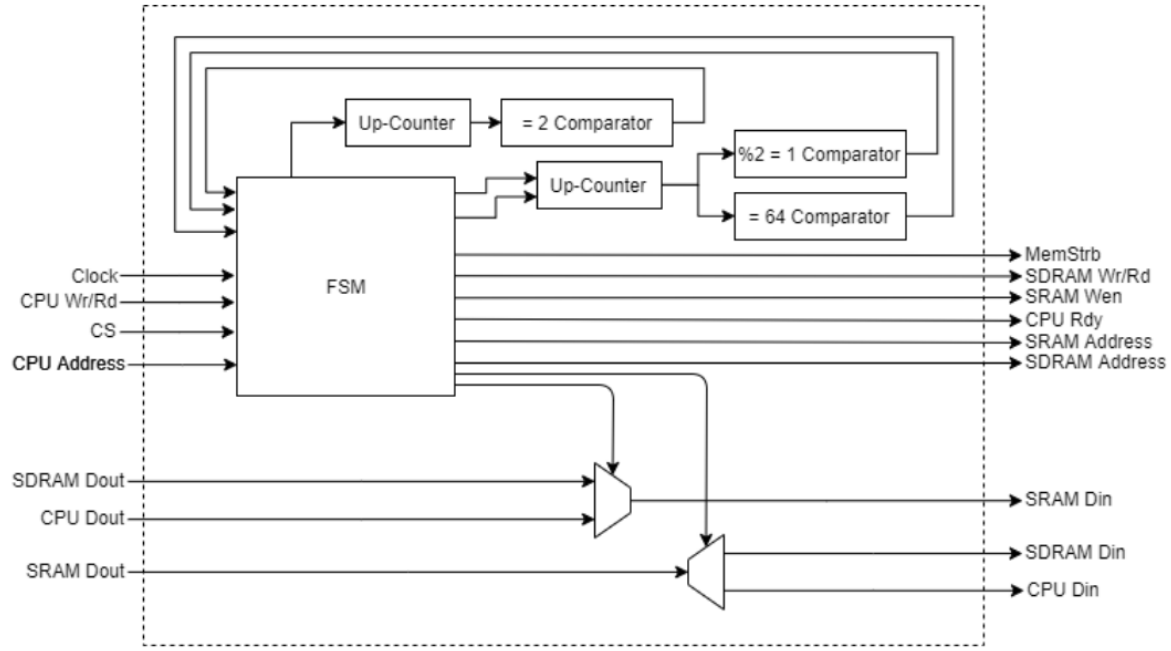


Figure 4: Cache Controller Block Diagram

State Diagrams

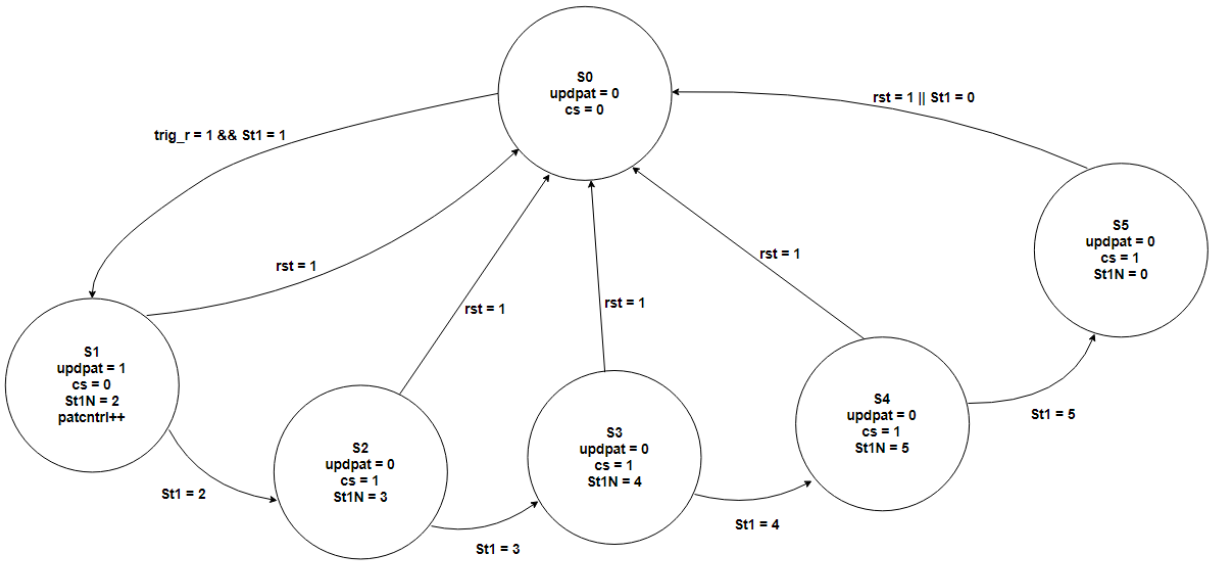


Figure 5: CPU State Diagram

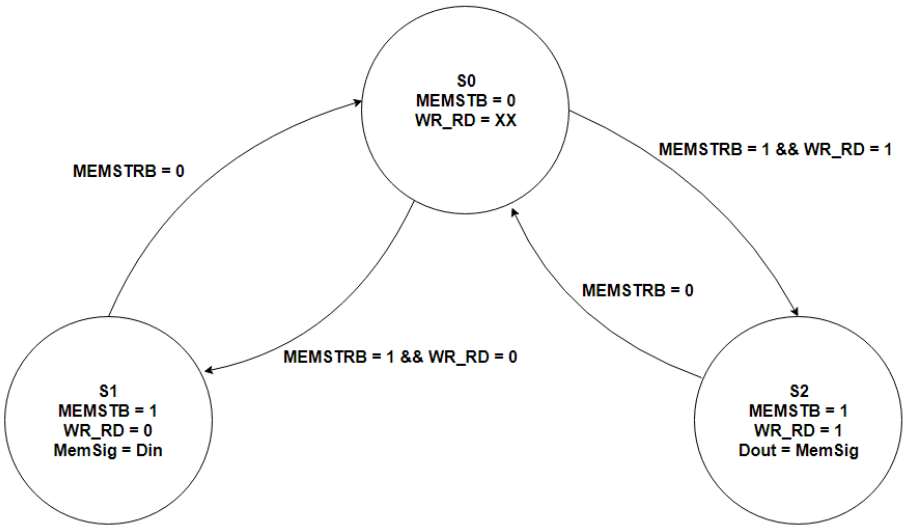


Figure 6: SDRAM State Diagram

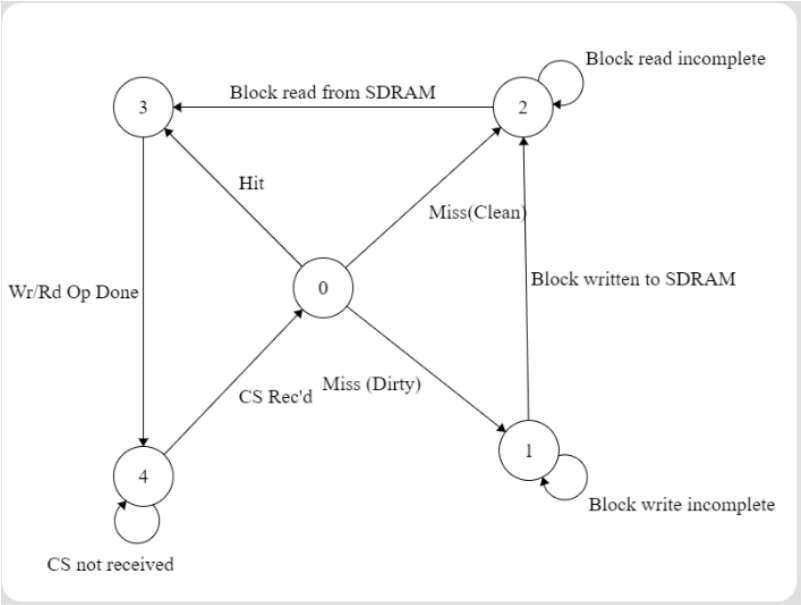


Figure 7: Complete Cache Controller State Diagram

Table 2: Complete Cache State Operation Table

State	**Operation**
0	Evaluation of CPU ADDR
1	Write block to SDRAM
2	Read block from SDRAM
3	Write/Read To/From SRAM From/To CPU
4	Ready and waiting

Table 3: Complete Cache Behavior Table

Case	**Behavior**	**State Progression**
1	Write a word to cache [hit]	0 - 3 - 4 - 0
2	Read a word from cache [hit]	0 - 3 - 4 - 0
3	Read/Write from/to cache [miss] and dirty bit = 0	0 - 2 - 3 - 4 - 0
4	Read/Write from/to cache [miss] and dirty bit = 1	0 - 1 - 2 - 3 - 4 - 0

Process Diagram

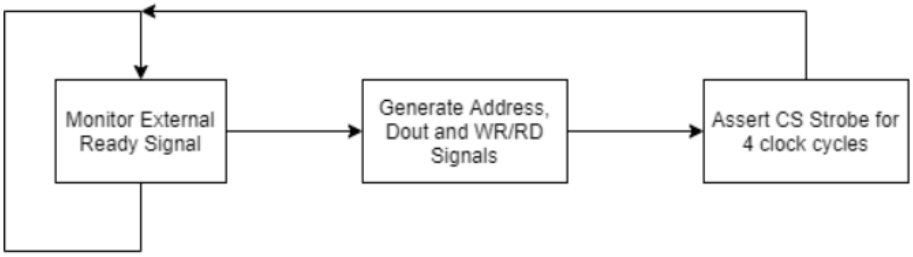


Figure 8: CPU Process Diagram

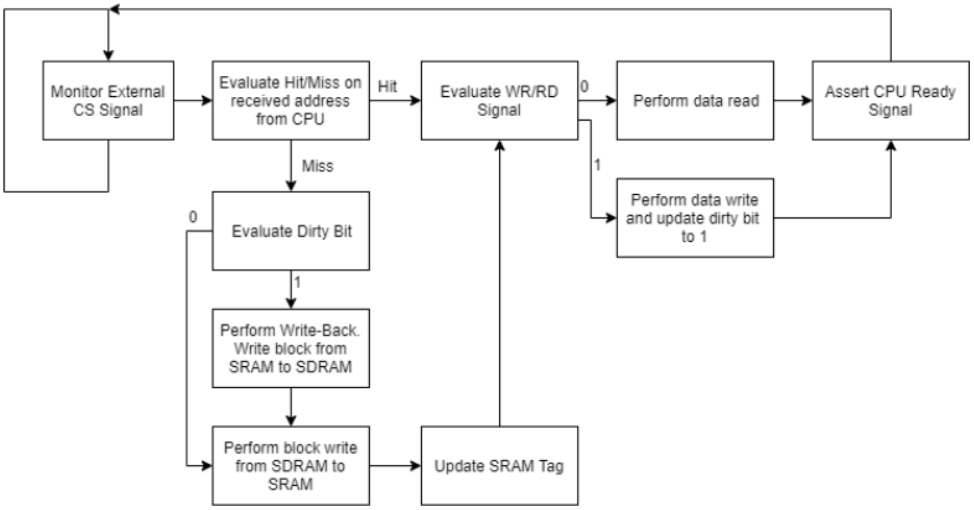


Figure 9: Cache Controller Process Diagram

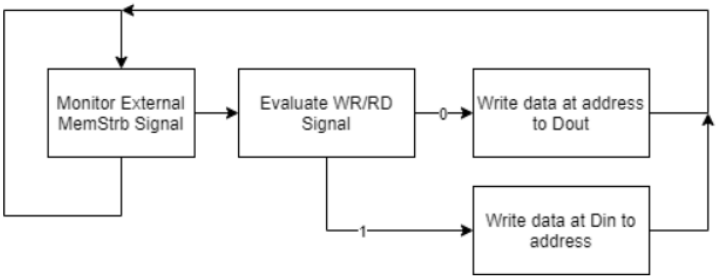


Figure 10: SDRAM Process Diagram

Results

Timing Diagrams

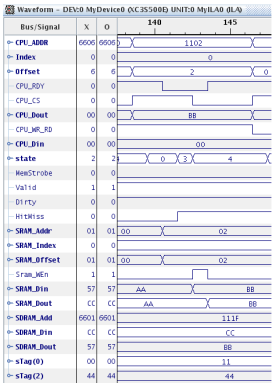


Figure 11: Write a word to cache [hit]

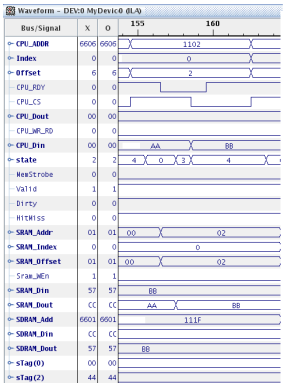


Figure 12: Read a word from cache [hit]

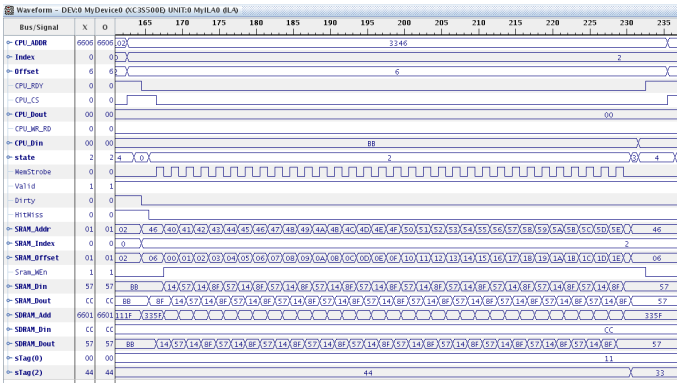


Figure 13: Read/Write from/to cache [miss] and dirty bit = 0

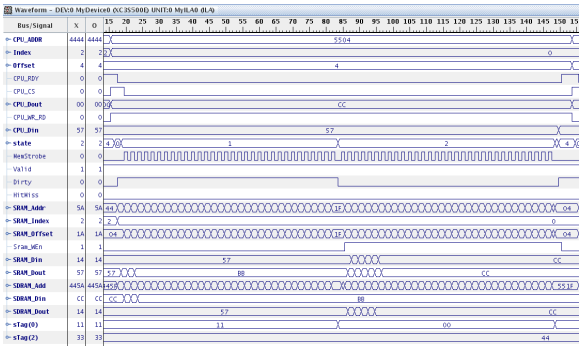


Figure 14: Read/Write from/to cache [miss] and dirty bit = 1 (See Appendix for Detailed)

Cache Parameters

N	Cache performance parameter	Time (ns)	Clock Cycles
1	Hit/Miss Determination Time	2	2
2	Data Access Time	1	1
3	Block Replacement Time	130	130
4	Hit Time (Case 1 and Case 2)	7	7
5	Miss Penalty for Case 3 (when D-bit = 0)	65	65
6	Miss Penalty for Case 4 (when D-bit = 1)	130	130

Table 4: Cache Performance Table

Brief Explanation

Conclusion

In this lab, the objective benefits of creating and utilizing a simple cache controller was accomplished. As shown in table 4, when hit occurs in the cache controller the access time for that particular data is far smaller when compared to the access time of SDRAM. In most cache controllers, the hit rate is upwards of 95%, therefore the overall access time of a digital system which utilizes a cache memory will be much less than those that do not.

The secondary goals of learning and expanding the knowledge of VHDL within the Xilinx ISE CAD environment and gaining experience in the design and implementation of custom logic controllers was also achieved.

References

1. D. A. Patterson, J. L. Hennessy, and P. Alexander, Computer organization and design: the hardware/ software interface. Amsterdam: Morgan Kaufmann, 2015.

2.[Online]. Available: https://www.ee.ryerson.ca/ lkirisch/ele758/labs/Cache Project[12-09-10].pdf. [Accessed: 03-Nov-2019].

Appendix

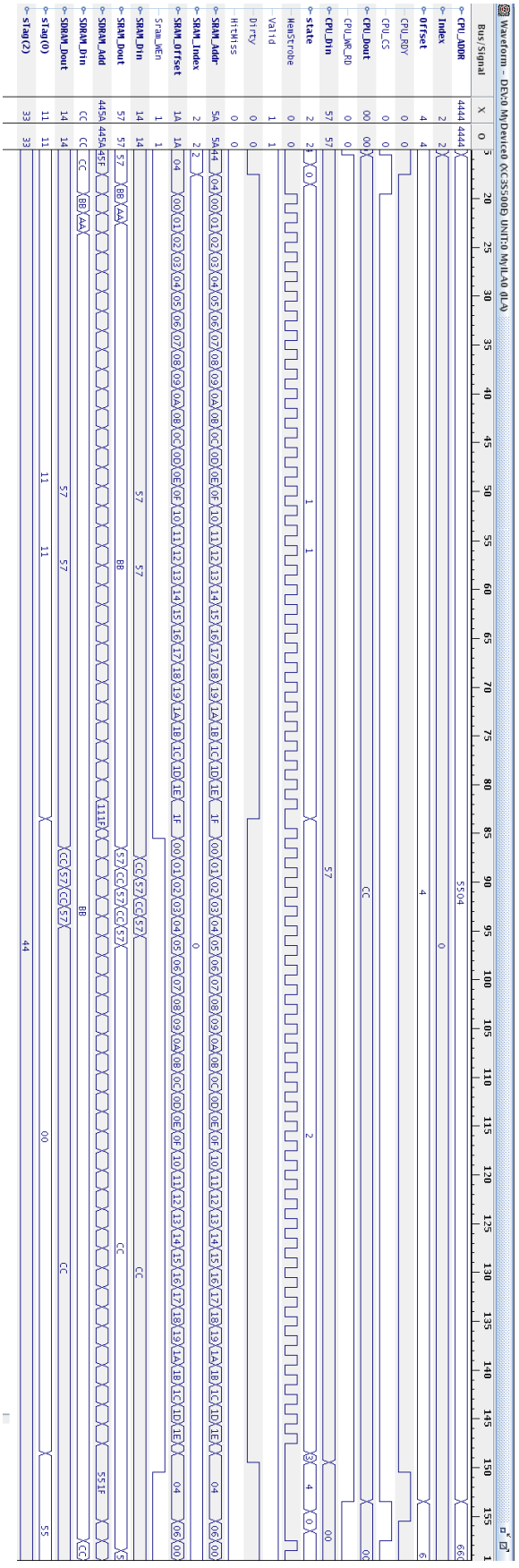


Figure 15: Read/Write from/to cache [miss] and dirty bit = 1