**CS4223 Mini-Project**

**Cache Coherence Simulator for Multicore Architectures**

Abdul Rahman ( ) and Rahumathmarini (A0138471A)

|  |
| --- |
|  |

Table of Contents

[1. Introduction 2](#_Toc530514617)

[1.1 Implemented Cache Coherence Protocols 2](#_Toc530514618)

[1.1.1 MESI Protocol 2](#_Toc530514619)

[1.1.2 Dragon Protocol 3](#_Toc530514620)

[1.2 Inputs & Outputs 4](#_Toc530514621)

[2. Implementation 5](#_Toc530514622)

[2.1 Overview of Components 5](#_Toc530514623)

[2.2 Components Description 6](#_Toc530514624)

[3. Quantitative Analysis 6](#_Toc530514625)

[3.1 Analysis for BlackScholes with Associativity 1, 2 and 4 7](#_Toc530514626)

[3.2 Analysis for BlackScholes with Block Size = 8, 16 and 32) 9](#_Toc530514627)

[3.3 Analysis for BodyTrack with Associativity 1, 2 and 4) 12](#_Toc530514628)

[3.4 Analysis for BodyTrack with Block Size 8, 16 and 32) 14](#_Toc530514629)

[3.5 Analysis for FluidAnimate with Associativity 1, 2 and 4) 17](#_Toc530514630)

[3.6 Analysis for FluidAnimate with Block Size 8, 16 and 32) 19](#_Toc530514631)

[3.6 Conclusion 22](#_Toc530514632)

[APPENDIX I – Pre-Analysis Data 22](#_Toc530514633)

[APPENDIX II – Sample Raw Data 28](#_Toc530514634)

# 1. Introduction

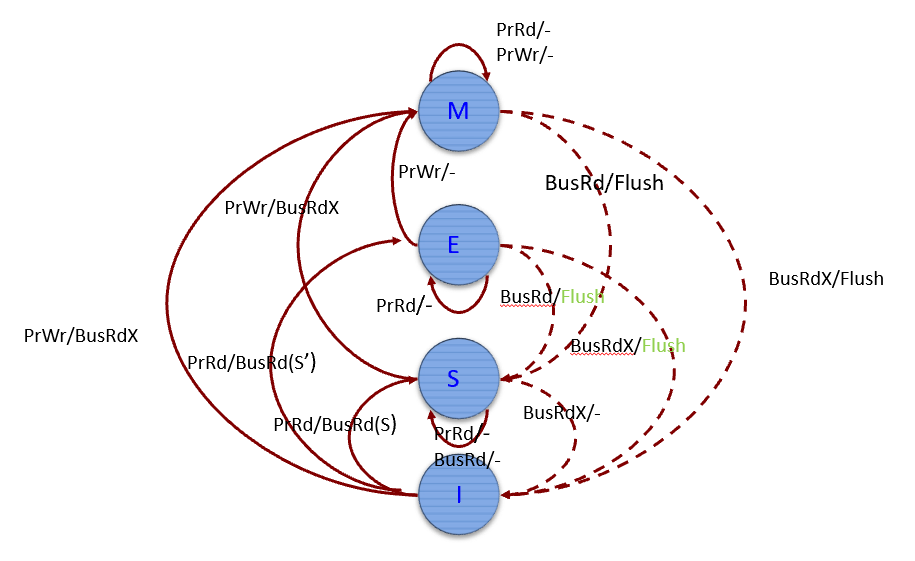
The objective of this project is to implement a trace-driven cache coherence simulator for multi-core architectures. This report will describe the implementation of the python-based simulator and provides quantitative analysis for two different protocols. Optimal design point is discovered by simulating with varying parameters such as block size and cache associativity across 3 benchmarks.

## 1.1 Implemented Cache Coherence Protocols

The two protocols implement is MESI and Dragon. In this section we will summarise the core concepts of each of the two protocols.

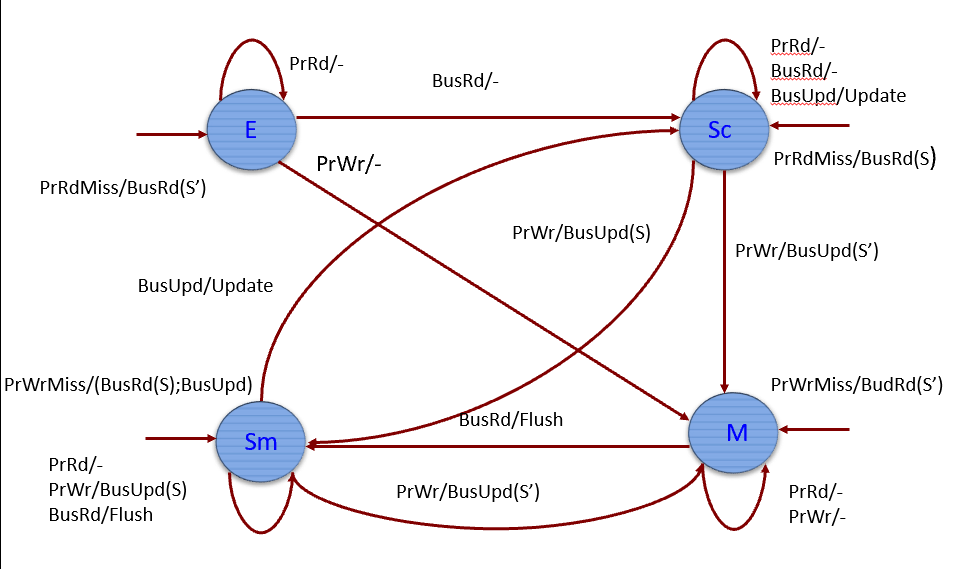
### 1.1.1 MESI Protocol

The MESI protocol is an Invalidation-based cache coherence protocol which supports write-back caches. It has 4 states namely, Invalid, Exclusive. Shared and Modified. The state transition diagram below explains the behaviour of this protocol.



### 1.1.2 Dragon Protocol

The dragon protocol is an update-based cache coherence protocol which supports write through. It has 4 states namely, Exclusive, Shared Clean, Shared Modified, Dirty. The state transition diagram below explains the behaviour of this protocol.



## 1.2 Inputs & Outputs

|  |  |
| --- | --- |
| **Inputs** | |
| Protocol | MESI or Dragon |
| Benchmark Files | blackscholes, bodytrack or fluidanimate |
| Cache Size | 4096 Bytes |
| Associativity | 1, 2 or 4 |
| Block Size | 8, 16, 32 Bytes |

The simulator should be run on command prompt / terminal using the above-mentioned input arguments. The format for the command is as follows:

Python coherence.py [Protocol] [Benchmark] [Cache size] [Associativity] [Block size]

E.g. Python coherence.py mesi blackscholes 4096 2 32

|  |  |
| --- | --- |
| **Outputs** | |
| Overall Simulation Result: | |
| Time Taken | Time taken to execute the entire trace to complete in seconds |
| Overall Execution Cycle | Total number of cycles taken for entire trace to complete |
| Bus-Specific Result: | |
| Data traffic | Total amount of data sent across the bus in bytes |
| Invalidations | Total BusRdx requests |
| Bus updates | Total Bus Updates |
| Core-Specific Results: | |
| Total Cycles | Total number of cycles taken per core |
| Compute Cycles | Total number of compute cycles per core |
| Load Cycles | Total number of load cycles per core |
| Store Cycles | Total number of store cycles per core |
| Idle Cycles | Total number of idle cycles per core |
| Data miss rate | Overall percentage of miss rate |
| Private data accesses | Access count to data internal cache blocks |
| Public data access | Access count to modified data in other caches blocks |

# 2. Implementation

In this section we will describe the overview and implementation of the components in the simulator.

## 2.1 Overview of Components

Programming Language: Python 3

Data Structure: Python Collection

## 2.2 Components Description

**Coherence.py** – Top level file that receives input form command prompt/ terminal and passes to simulation.py

**Simulation.py** – Runs simulation and prints out the result of simulation.

**Cache.py** – Performs cache operation such as eviction and allocations. Cache.py calls lru.py to perform LRU operations.

**Mesi.py** – Performs MESI specific operations

**Dragon.py** – Performs Dragon specific operations

**Core.py** – Reads instructions and execute/ stall instructions for specific cores

**Snooping.py** – Check if a cache is exclusive or share to perform necessary operation

# 3. Quantitative Analysis

## 3.1 Analysis for BlackScholes with Associativity 1, 2 and 4

The total cycle for both MESI and Dragon protocols has decreases when associativity is increased which could be because there is less conflict misses as there is more buffer to store data in a single block ID. The total cycle for both MESI and Dragon is similar for each level of associativity.

MESI protocol has generally lesser Data than Dragon protocol. This is because, only flushes are taken into consideration for MESI protocol. The data traffic decreases as associativity increases which could be because of the decreasing data misses with increased associativity. This means more data are being found in local caches while processors that has shared data are not modifying the data.

The total number of invalidations is more or less the same for all the 3 level of associativity for MESI protocol. As expected, the dragon protocol does not have invalidation as it is a update-based protocol.

The number bus update is for associativity of 1 is much higher than associativity of 2 and 4 for both protocols. Although the overall trend observed to be same for both protocols, the MESI protocol has lesser number of bus update as compared to the Dragon Protocol.

The average data miss rate for both protocols seem to be high for associativity of 1 and it significantly drops for associativity of 2. There is a drop in the average cache miss form associativity of 4 but it’s not significant. There is no clear difference between the two protocols here.

## 3.2 Analysis for BlackScholes with Block Size = 8, 16 and 32)

The total cycles for both protocols have dropped when block size is increased. This could be due to a higher amount spatial locality for this benchmark. MESI protocol has taken slightly more cycles than Dragon protocol, this is because MESI do more memory reads.

The is not much difference in data traffic between the different block size for both protocols. However, the Dragon protocol has a higher amount of data traffic than MESI protocol and this is due to the fact data traffic only affects MESI protocol whenever an M block gets invalidated.

The total number of invalidations decreases as the block size increases in MESI protocol this could be because of the main factor of the decreased cycle for higher block size and naturally less validation will occur. For Dragon Protocol, there is no invalidation as it is an update-based protocol, as mentioned earlier.

The number of bus update is lesser in MESI protocol as compared to Dragon protocol.

The data miss at its highest for associativity of 2 for both protocols. There is no difference between both protocols here.

## 3.3 Analysis for BodyTrack with Associativity 1, 2 and 4)

As associativity increases, the total cycle decreases for both MESI and Dragon protocol. Although both graph ahs similar trends with increased associativity, the MESI protocol has slightly more total cycles than the dragon protocol for each level of associativity.

The data traffic decreases with increased associativity for both protocols. As expected the MESI protocol has lesser Data Traffic as compared to Dragon protocol as MESI protocol by nature does not have a lot of bus traffic, and the only bus traffic created by MESI protocol is due to write-back of M blocks on invalidation.

The total invalidation increases with increase associativity. For earlier mentioned reasons, invalidations does not occur in Dragon protocol.

The number of bus updates are higher for MESI as compared to the Dragon protocol.

The average data miss rate decreases with increased associativity. This indicates that the benchmark has tendency to cause conflict miss when associativity is low. There is no clear difference between both the protocols.

## 3.4 Analysis for BodyTrack with Block Size 8, 16 and 32)

The total cycle taken for both the protocols are the same. In both cases the total cycle taken is decreases with increased block size.

The data traffic follows a similar trend as before, with higher block size there is an decreased amount of data traffic. Again, MESI has lesser data traffic due to the fact the only affecting factor for MESI is when an write-back occur during the invalidation of M state.

As observed in BlackScholes, the total number of invalidations for MESI protocol decreases as the block size increases.

The bus update is lower for MESI protocol as compared to the Dragon protocol.

The average data miss rate has similar trend with total cycles as both the total cycles and average data miss rate decreases with increased block size. Both the protocols are identical in trend and amount.

## 3.5 Analysis for FluidAnimate with Associativity 1, 2 and 4)

FluidAnimate’s total cycles decreases with increasing associativity at about the same rate in both protocols.

The data traffic decreases with increasing associativity for both protocols and the MESI protocol has lesser data traffic than dragon protocol. This is similar to earlier observation using benchmark BlackScholes and BodyTrack.

The number of total invalidations for MESI protocol has increased with increasing associativity. As expected there is not invalidation for Dragon protocol.

The number of bus updates is lesser for MESI protocol as compared to the Dragon protocol. For both protocols the number of bus updates decreases as associativity increases. Moreover, the decrease from one associativity to another is steeper in Dragon protocol compared to MESI protocol.

The average data miss rate drops with increased associativity. The decrease between associativity of 1 and 2 is more than the decrease between the associativity of 2 and 4.

## 3.6 Analysis for FluidAnimate with Block Size 8, 16 and 32)

The total cycles taken to run FluidAnimate decreases as the block size increases. Once again, this indicates that the spatial locality is taken advantage with increased block size. There is no difference between the 2 protocols.

The data traffic decreases again with increasing block size and MESI has lesser data traffic compared to Dragon as observed in other 2 benchmarks.

The total number of invalidations for MESI protocol decreases with increasing block size as observed in other benchmarks. Invalidation does not occur for dragon protocol.

The bus updates, as observed earlier has increased with increasing data size and the Dragon protocol has higher bus updates than MESI because MESI only updates when M state data becomes invalidated.

Again, the average data miss rate decreases with increasing block size due to the reason that there is a decreasing total with increasing block size.

## 3.6 Conclusion

For all the 3 benchmarks, MESI protocol is leading in terms of total cycles taken by a margin of 20,000 cycles. MESI protocol is also more desirable in terms of data traffic as it has a lot of lesser data traffic compared to Dragon protocol. As there is less data traffic for MESI protocol, it would require less power to operate. MESI is notably the better candidate as it has a better speed (lesser total cycle) and efficient power consumption which are the 2 important matrics for supercomputers today.

# APPENDIX I – Pre-Analysis Data



Figure 1: BlackScholes, MESI



Figure 2: BlackScholes, Dragon



Figure 3: BodyTrack, MESI



Figure 4: BodyTrack, Dragon



Figure 5: FluidAnimate, MESI



Figure 6: FluidAnimate, Dragon

# APPENDIX II – Sample Raw Data

Only raw data with associativity of 1 and block size 16 are shown, all the raw data is available on request.

BLACKSCHOLES, MESI

**('mesi', 'blackscholes', 4096, 1, 16)**

SIMULATION REPORT

------------------------------

Time taken: 226.88953375816345

Overall Execution Cycle: 22291364

Bus snooping results

Data traffic: 403065 | Invalidations: 432 | Bus updates: 56829

------------------------------

Core 0 | Total cycles: 20783974 | Compute cycles: 10430314 | Load cycles: 1489888 | Store cycles: 1007461 | Idle cycles: 7820673 | Data miss rate: 3.9634828772430284 | Private data accesses: 8949 | Public data accesses: 41859

------------------------------

Core 1 | Total cycles: 20570386 | Compute cycles: 10383276 | Load cycles: 1485857 | Store cycles: 1004611 | Idle cycles: 7661963 | Data miss rate: 3.929984243925238 | Private data accesses: 9030 | Public data accesses: 41147

------------------------------

Core 2 | Total cycles: 22291364 | Compute cycles: 10430338 | Load cycles: 1492629 | Store cycles: 1016428 | Idle cycles: 9325869 | Data miss rate: 4.2708475734110465 | Private data accesses: 8880 | Public data accesses: 42657

------------------------------

Core 3 | Total cycles: 20699932 | Compute cycles: 10394904 | Load cycles: 1493736 | Store cycles: 1009391 | Idle cycles: 7769225 | Data miss rate: 3.9526560178528696 | Private data accesses: 9822 | Public data accesses: 40980

------------------------------

BODYTRACK, MESI

**('mesi', 'bodytrack', 4096, 1, 16)**

SIMULATION REPORT

------------------------------

Time taken: 357.5287194252014

Overall Execution Cycle: 46433275

Bus snooping results

Data traffic: 753573 | Invalidations: 1943 | Bus updates: 99356

------------------------------

Core 0 | Total cycles: 46433275 | Compute cycles: 17729254 | Load cycles: 2380720 | Store cycles: 889412 | Idle cycles: 25438374 | Data miss rate: 7.372638168734473 | Private data accesses: 21142 | Public data accesses: 20845

------------------------------

Core 1 | Total cycles: 45860702 | Compute cycles: 17120545 | Load cycles: 2388005 | Store cycles: 899247 | Idle cycles: 25452660 | Data miss rate: 7.5591710036224775 | Private data accesses: 27367 | Public data accesses: 19256

------------------------------

Core 2 | Total cycles: 19235544 | Compute cycles: 17556877 | Load cycles: 74523 | Store cycles: 43175 | Idle cycles: 1563271 | Data miss rate: 10.468317218644327 | Private data accesses: 695 | Public data accesses: 1469

------------------------------

Core 3 | Total cycles: 46414841 | Compute cycles: 17140113 | Load cycles: 2416052 | Store cycles: 908867 | Idle cycles: 25960648 | Data miss rate: 7.524453979179643 | Private data accesses: 27290 | Public data accesses: 19920

------------------------------

FLUIDANIMATE, MESI

**('mesi', 'fluidanimate', 4096, 1, 16)**

SIMULATION REPORT

------------------------------

Time taken: 408.3678376674652

Overall Execution Cycle: 62141844

Bus snooping results

Data traffic: 1133623 | Invalidations: 2311 | Bus updates: 25583

------------------------------

Core 0 | Total cycles: 58975965 | Compute cycles: 11337782 | Load cycles: 1832392 | Store cycles: 744111 | Idle cycles: 45217921 | Data miss rate: 11.157836804381754 | Private data accesses: 2387 | Public data accesses: 10470

------------------------------

Core 1 | Total cycles: 55732818 | Compute cycles: 11290799 | Load cycles: 1821846 | Store cycles: 585998 | Idle cycles: 42178850 | Data miss rate: 11.20325901511892 | Private data accesses: 2823 | Public data accesses: 9212

------------------------------

Core 2 | Total cycles: 62141844 | Compute cycles: 11337671 | Load cycles: 1838008 | Store cycles: 766181 | Idle cycles: 48369770 | Data miss rate: 11.831360934248627 | Private data accesses: 2109 | Public data accesses: 13922

------------------------------

Core 3 | Total cycles: 55142598 | Compute cycles: 11301515 | Load cycles: 1832174 | Store cycles: 579291 | Idle cycles: 41571095 | Data miss rate: 11.07600566460637 | Private data accesses: 3516 | Public data accesses: 8569

------------------------------

------------------------------

BLACKSCHOLES, Dragon

**dragon blackscholes 4096 1 16**

SIMULATION REPORT

------------------------------

Time taken: 237.16698956489563

Overall Execution Cycle: 21656461

Bus snooping results

Data traffic: 689745 | Invalidations: 0 | Bus updates: 219384

------------------------------

Core 0 | Total cycles: 20398428 | Compute cycles: 10430314 | Load cycles: 1489888 | Store cycles: 1007461 | Idle cycles: 7677492 | Data miss rate: 3.964043471697388 | Private data accesses: 14475 | Public data accesses: 39106

------------------------------

Core 1 | Total cycles: 20184390 | Compute cycles: 10383276 | Load cycles: 1485857 | Store cycles: 1004611 | Idle cycles: 7524236 | Data miss rate: 3.93155021465845 | Private data accesses: 12510 | Public data accesses: 37413

------------------------------

Core 2 | Total cycles: 21656461 | Compute cycles: 10430338 | Load cycles: 1492629 | Store cycles: 1016428 | Idle cycles: 8924519 | Data miss rate: 4.274952701353536 | Private data accesses: 9826 | Public data accesses: 38209

------------------------------

Core 3 | Total cycles: 20305951 | Compute cycles: 10394904 | Load cycles: 1493736 | Store cycles: 1009391 | Idle cycles: 7618096 | Data miss rate: 3.95677087099456 | Private data accesses: 14772 | Public data accesses: 38767

------------------------------

BODYTRACK, Dragon

**('dragon', 'bodytrack', 4096, 1, 16)**

SIMULATION REPORT

------------------------------

Time taken: 278.12783646583557

Overall Execution Cycle: 45929537

Bus snooping results

Data traffic: 1339688 | Invalidations: 0 | Bus updates: 192486

------------------------------

Core 0 | Total cycles: 44946746 | Compute cycles: 17729254 | Load cycles: 2380720 | Store cycles: 889412 | Idle cycles: 24174285 | Data miss rate: 7.354443184556464 | Private data accesses: 36469 | Public data accesses: 27079

------------------------------

Core 1 | Total cycles: 45929537 | Compute cycles: 17120545 | Load cycles: 2388005 | Store cycles: 899247 | Idle cycles: 25742998 | Data miss rate: 7.557437032512262 | Private data accesses: 28903 | Public data accesses: 28619

------------------------------

Core 2 | Total cycles: 19193069 | Compute cycles: 17556877 | Load cycles: 74523 | Store cycles: 43175 | Idle cycles: 1536619 | Data miss rate: 10.443677887474724 | Private data accesses: 682 | Public data accesses: 1520

------------------------------

Core 3 | Total cycles: 44963644 | Compute cycles: 17140113 | Load cycles: 2416052 | Store cycles: 908867 | Idle cycles: 24712097 | Data miss rate: 7.50881450044347 | Private data accesses: 36686 | Public data accesses: 32131

------------------------------

FLUIDANIMATE, Dragon

**dragon fluidanimate 4096 1 16**

SIMULATION REPORT

------------------------------

Time taken: 388.4426610469818

Overall Execution Cycle: 61814841

Bus snooping results

Data traffic: 2217664 | Invalidations: 0 | Bus updates: 57509

------------------------------

Core 0 | Total cycles: 58987707 | Compute cycles: 11337782 | Load cycles: 1832392 | Store cycles: 744111 | Idle cycles: 45301128 | Data miss rate: 11.154227260748387 | Private data accesses: 2078 | Public data accesses: 11039

------------------------------

Core 1 | Total cycles: 55695748 | Compute cycles: 11290799 | Load cycles: 1821846 | Store cycles: 585998 | Idle cycles: 42209694 | Data miss rate: 11.196489473570546 | Private data accesses: 3211 | Public data accesses: 9616

------------------------------

Core 2 | Total cycles: 61814841 | Compute cycles: 11337671 | Load cycles: 1838008 | Store cycles: 766181 | Idle cycles: 48100131 | Data miss rate: 11.828327360264558 | Private data accesses: 2306 | Public data accesses: 14305

------------------------------

Core 3 | Total cycles: 55089084 | Compute cycles: 11301515 | Load cycles: 1832174 | Store cycles: 579291 | Idle cycles: 41585924 | Data miss rate: 11.070987967895036 | Private data accesses: 3939 | Public data accesses: 10738

------------------------------