Akshay Sawant
ECE – 6100
GTID: 903067934

Project 3: Cache Coherence

In this project, we have created a simulator that maintains coherent caches for a 4, 8 and 16 core CMP (Chip Multiprocessor). We have implemented MSI, MESI, MOSI, MOESI, and MOESIF protocols for a bus-based broadcast system.

The various states that have been used in cache coherence are as follows:

M (Modified) state: This state indicates that the block has been modified in the cache. The data in cache is inconsistent with the main memory. A cache with block 'M' has the responsibility to write the block back to the main memory when it is evicted.

S (Shared) state: This state indicates that the block is unmodified and it exists as a read-only state in at least one cache. The cache can evict this block without modifying the main memory.

I (Invalid) state: This state indicates that the block is not present in the current cache. If the block has to be stored in this cache, it has to be fetched from memory or another cache.

O (Owned) state: This state indicates that the current cache is one of the several caches with valid copy of the cache line, but it has the exclusive right to make changes to it. It must broadcast these changes to all the caches sharing the line.

E (Exclusive) state: This state indicates that the current cache has the only copy of the block, but the block is unmodified. It may be changed to shared or modified state depending on a read or a write request.

F (Forwarder) state: This state is a specialized form of the S state, and it indicates that the current cache should act as a designated responder for any request for the given line.

Part 1] Validation Results

The results are validated using the verification script that is provided.

```
make[1]: Leaving directory `/home/akshay/Codes/ACA3_Verify/project3/sim'
g++ -o sim_trace  -Llib/ -lsim -lprotocols
----------------------------
Verifying protocol: MSI
traces/4proc_validation: PASS
traces/8proc_validation: PASS
traces/16proc_validation: PASS
----------------------------
Verifying protocol: MESI
traces/4proc_validation: PASS
traces/8proc_validation: PASS
traces/16proc_validation: PASS
----------------------------
Verifying protocol: MOSI
traces/4proc_validation: PASS
traces/8proc_validation: PASS
traces/16proc_validation: PASS
----------------------------
Verifying protocol: MOESI
traces/4proc_validation: PASS
traces/8proc_validation: PASS
traces/16proc_validation: PASS
----------------------------
Verifying protocol: MOESIF
traces/4proc_validation: PASS
traces/8proc_validation: PASS
traces/16proc_validation: PASS
----------------------------
akshay@Lenovo:~/Codes/ACA3_Verify$
```

The experiments are performed on the next page.

Part 2] Experiments

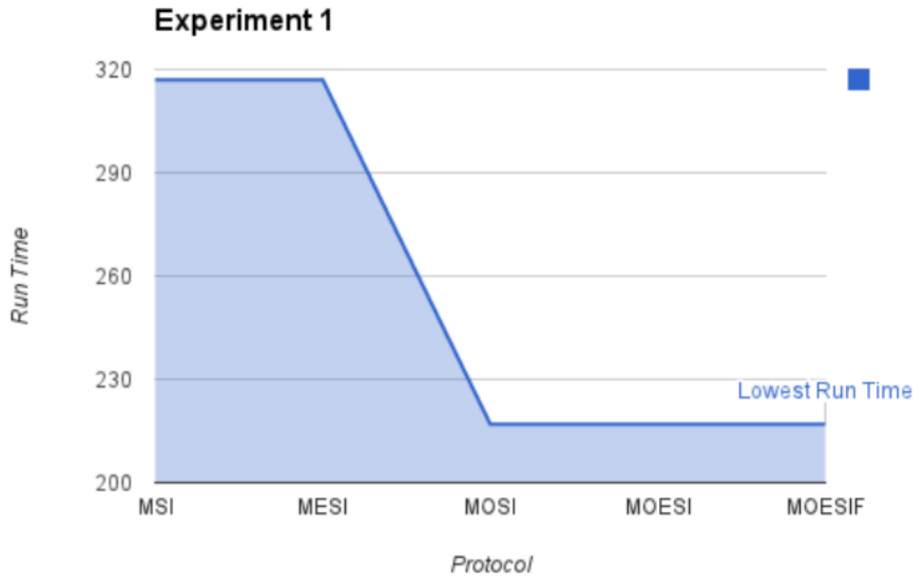
The experiment results are stored in the folder 'Experiment Outputs'.

Experiment 1:

The final statistics of Experiment 1 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	317	7	12	0	4	
MESI	317	7	12	0	4	
MOSI	217	7	12	0	5	Lowest Run Time
MOESI	217	7	12	0	5	Lowest Run Time
MOESIF	217	7	12	0	5	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



In this experiment we observe that MOSI, MOESI and MOESIF protocols have the same execution time for the given traces. From the results, we observe that there are no silent upgrades at all. So Exclusive state becomes redundant over here and we can remove it. Also, we observe that cache to cache transfers increase once we add the Owned state, but not when we add the Forwarder state. So we can remove the Forwarder state as well. Because of the cache to cache transfers, we don't have to go to the main memory for the data. So the execution time decreases.

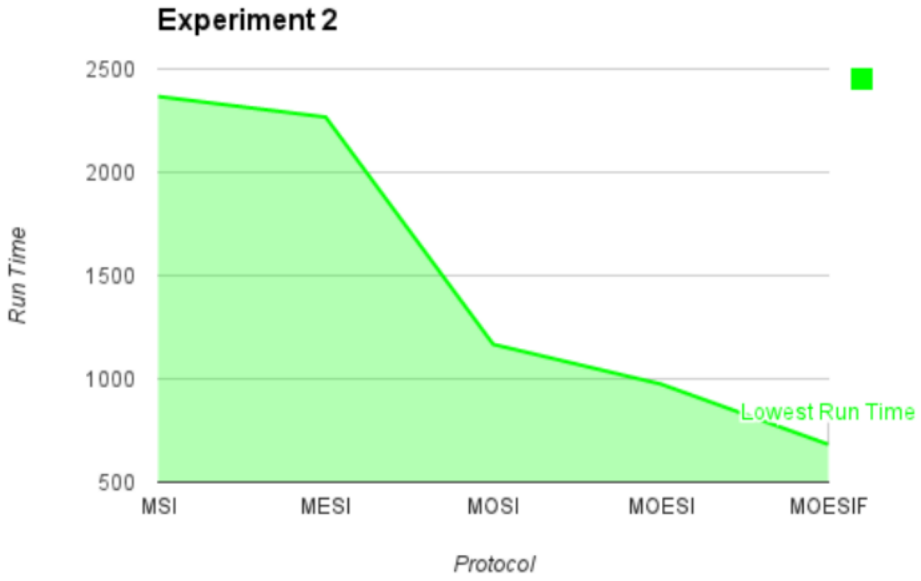
Hence, the best protocol for this experiment is **MOSI.**

Experiment 2:

The final statistics of Experiment 2 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	2367	30	104	0	7	
MESI	2267	30	104	1	8	
MOSI	1167	30	104	0	19	
MOESI	975	31	104	1	22	
MOESIF	683	34	104	1	28	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



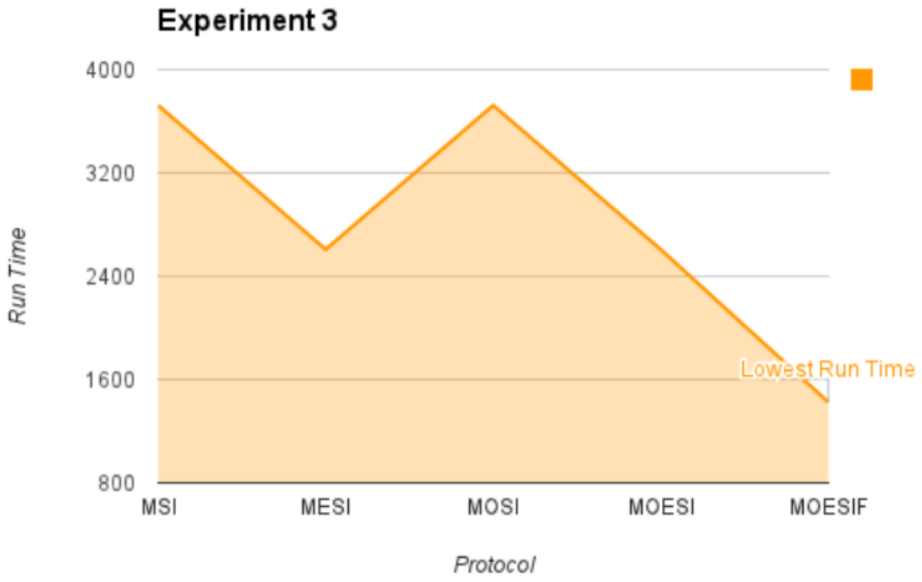
In this experiment, we observe that MOESIF protocol offers the lowest run time. From the results, we observe that there is one silent upgrade. Also, the run time does not decrease considerably when we compare MSI and MESI protocol. So, there aren't many new accesses. We also observe that run time decreases significantly when we compare MSI and MOSI protocol. So we can conclude that modified data is being accessed more, which results in more cache to cache transfers (7 in MSI to 19 in MOSI). Also, the run time decreases significantly when we compare MOESI and MOESIF protocols. So we can conclude that shared data is accessed again and again, resulting in more cache to cache transfers (22 in MOESI to 28 in MOESIF). Thus, we use Exclusive, Owned and Forwarder state in addition to MSI. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.

So, we conclude that **MOESIF** protocol is the best protocol for this trace.

Experiment 3:
The final statistics of Experiment 3 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	3723	56	200	0	20	
MESI	2607	48	200	8	23	
MOSI	3723	56	200	0	20	
MOESI	2607	48	200	8	23	
MOESIF	1425	48	200	8	35	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



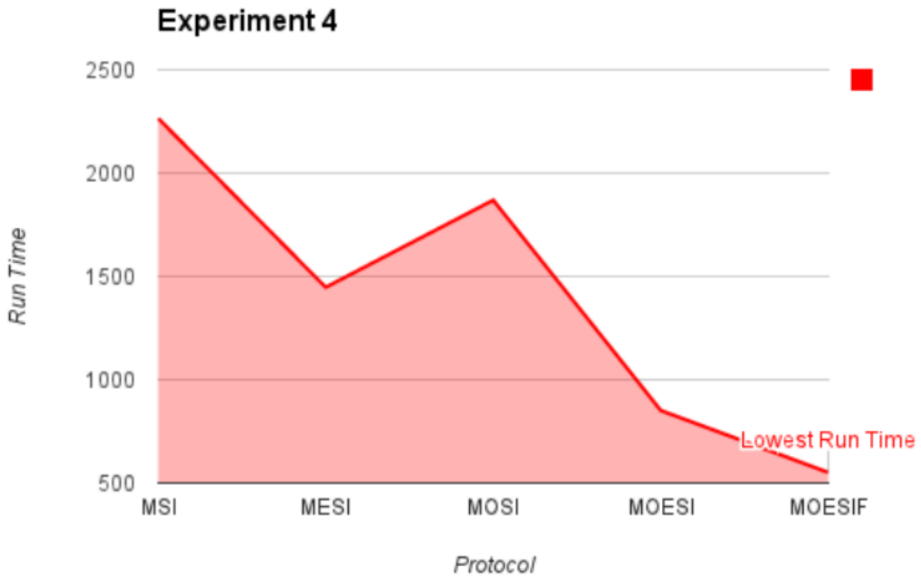
In this experiment, we observe that MOESIF protocol offers the lowest run time. From the results, we observe that there are 8 silent upgrades. So the run time decreases considerably when we compare MSI and MESI protocol. So, there are many new accesses. We also observe that run time does not decrease at all when we compare MSI and MOSI protocol. So we can conclude that modified data is not being accessed at all, which results in no change in the number of cache to cache transfers. Also, the run time decreases significantly when we compare MOESI and MOESIF protocols. So we can conclude that shared data is accessed again and again, resulting in more cache to cache transfers (23 in MOESI to 35 in MOESIF). Thus, we use Exclusive and Forwarder state in addition to MSI. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.

So, we conclude that **MOESIF** protocol is the best protocol for this trace. (Although MESIF protocol could have been used instead of MOESIF. Both would have given the same run time.)

Experiment 4:
The final statistics of Experiment 4 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	2265	27	60	0	5	
MESI	1447	19	60	3	5	
MOSI	1869	29	60	0	11	
MOESI	851	19	60	3	11	
MOESIF	551	19	60	3	14	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



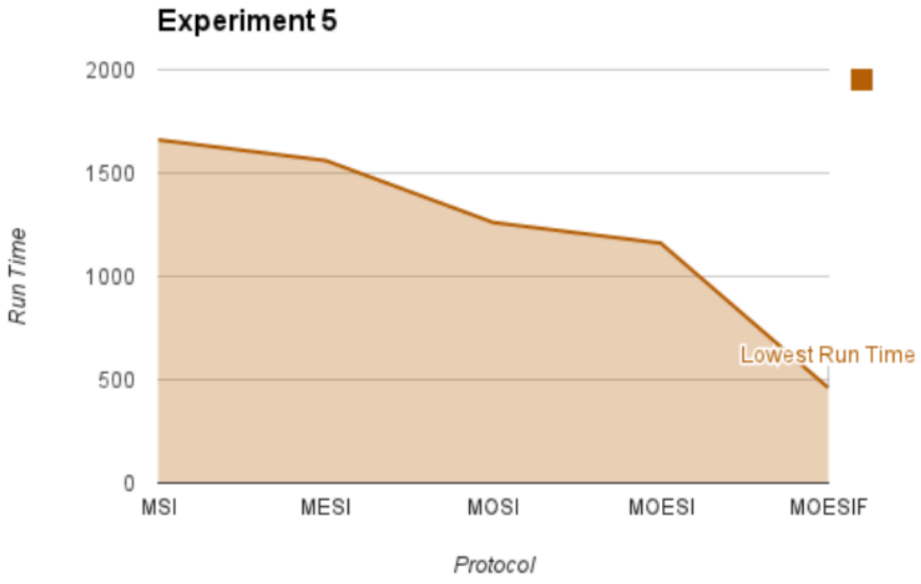
In this experiment, we observe that MOESIF protocol offers the lowest run time. From the results, we observe that there are 3 silent upgrades. So the run time decreases considerably when we compare MSI and MESI protocol. So, there are many new accesses. We also observe that run time decreases when we compare MSI and MOSI protocol. So we can conclude that modified data is being accessed a lot, which results in more cache to cache transfers (5 in MSI to 11 in MOSI). Also, the run time decreases even more when we compare MOESI and MOESIF protocols. So we can conclude that shared data is accessed again and again, resulting in more cache to cache transfers (11 in MOESI to 14 in MOESIF). Thus, we use Exclusive, Owned and Forwarder state in addition to MSI. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.

So, we conclude that **MOESIF** protocol is the best protocol for this trace.

Experiment 5:
The final statistics of Experiment 5 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	1661	21	37	0	5	
MESI	1561	21	37	0	6	
MOSI	1261	21	37	0	9	
MOESI	1161	21	37	0	10	
MOESIF	461	21	37	0	17	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



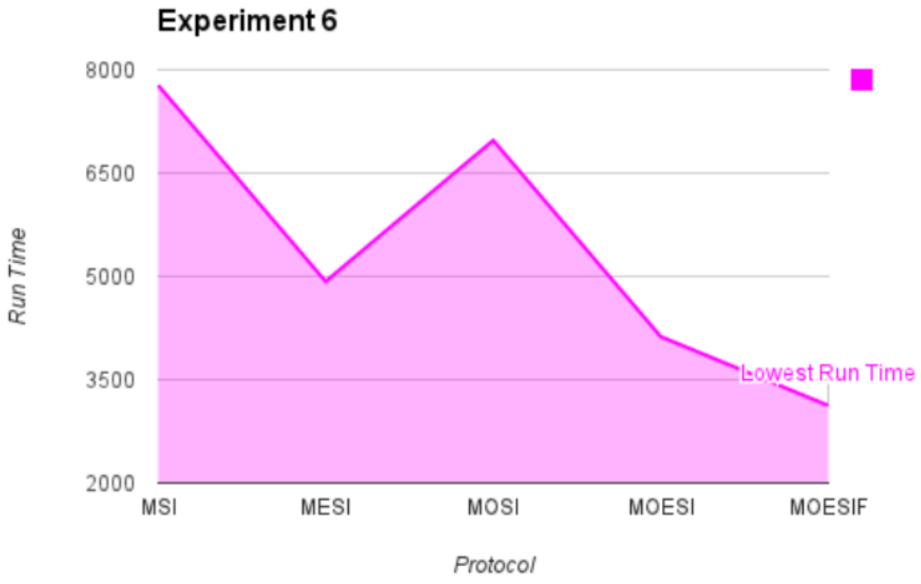
In this experiment, we observe that MOESIF protocol offers the lowest run time. From the results, we observe that there are no silent upgrades. Also, the run time decreases slightly when we compare MSI and MESI protocol. So, there aren't many accesses from Exclusive state. We also observe that run time decreases significantly when we compare MSI and MOSI protocol. So we can conclude that modified data is being accessed more, which results in more cache to cache transfers (5 in MSI to 9 in MOSI). Also, the run time decreases significantly when we compare MOESI and MOESIF protocols. So we can conclude that shared data is accessed again and again, resulting in more cache to cache transfers (10 in MOESI to 17 in MOESIF). Thus, we use Exclusive, Owned and Forwarder state in addition to MSI. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.

So, we conclude that **MOESIF** protocol is the best protocol for this trace.

Experiment 6:
The final statistics of Experiment 6 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	7775	87	747	0	12	
MESI	4925	62	747	25	15	
MOSI	6975	87	747	0	20	
MOESI	4125	62	747	25	23	
MOESIF	3125	62	747	25	33	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



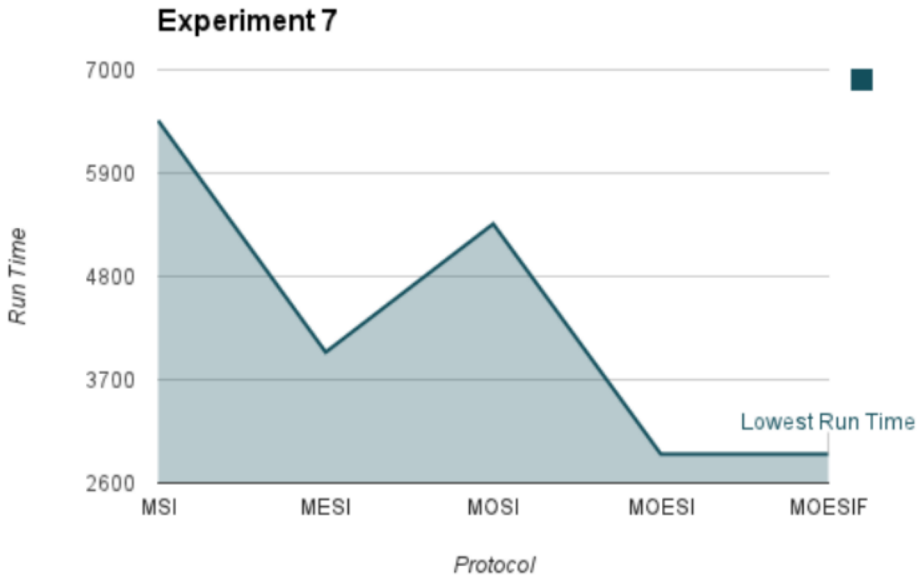
In this experiment, we observe that MOESIF protocol offers the lowest run time. From the results, we observe that there are 25 silent upgrades. So the run time decreases considerably when we compare MSI and MESI protocol. So, there are many new accesses, thus resulting in much more Exclusive states. We also observe that run time decreases when we compare MSI and MOSI protocol. So we can conclude that modified data is being accessed, which results in more cache to cache transfers (12 in MSI to 20 in MOSI). Also, the run time decreases significantly when we compare MOESI and MOESIF protocols. So we can conclude that shared data is accessed again and again, resulting in more cache to cache transfers (23 in MOESI to 33 in MOESIF). Thus, we use Exclusive, Owned and Forwarder state in addition to MSI. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.

So, we conclude that **MOESIF** protocol is the best protocol for this trace.

Experiment 7:
The final statistics of Experiment 7 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	6459	79	952	0	17	
MESI	3993	55	952	24	17	
MOSI	5359	79	952	0	28	
MOESI	2909	55	952	24	28	Lowest Run Time
MOESIF	2909	55	952	24	28	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



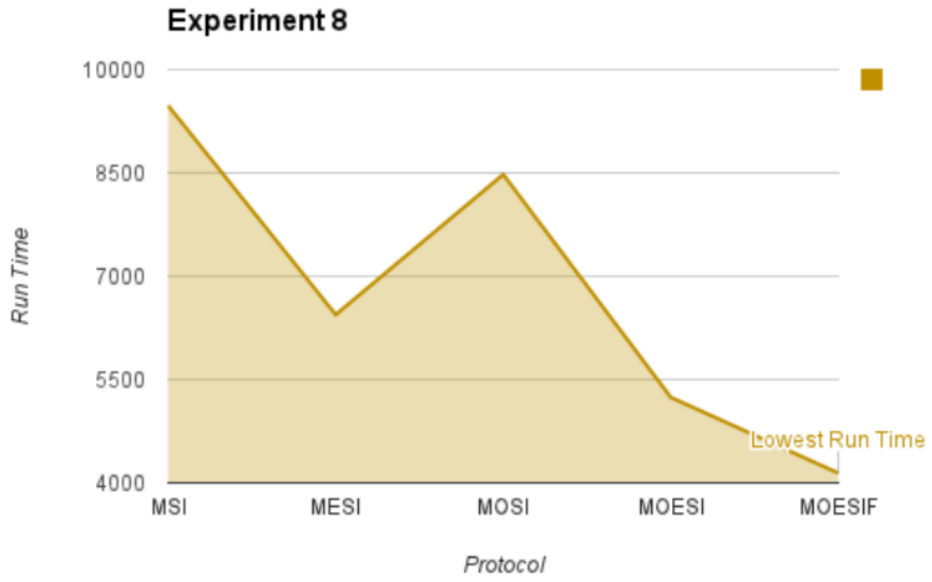
In this experiment, we observe that MOESI and MOESIF protocols offer the lowest run time. From the results, we observe that there are 24 silent upgrades. So the run time decrease considerably when we compare MSI and MESI protocol. So, there are many new accesses, resulting in many Exclusive states. We also observe that run time decreases significantly when we compare MSI and MOSI protocol. So we can conclude that modified data is being accessed more, which results in more cache to cache transfers (17 in MSI to 28 in MOSI). Also, the run time does not decrease at all when we compare MOESI and MOESIF protocols. So we can conclude that Forwarder state is redundant. Thus there is no change in the number of cache to cache transfers. Thus, we use Exclusive and Owned state in addition to MSI. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.

So, we conclude that **MOESI** protocol is the best protocol for this trace.

Experiment 8:
The final statistics of Experiment 8 are as follows:

Protocol	Run Time	Cache Misses	Cache Accesses	Silent Upgrades	$-to-$ Transfers	Comments
MSI	9477	110	800	0	18	
MESI	6441	92	800	19	30	
MOSI	8477	110	800	0	28	
MOESI	5241	92	800	19	42	
MOESIF	4141	92	800	19	53	Lowest Run Time

The graph of Protocol vs Run Time is plotted:



In this experiment, we observe that MOESIF protocol offers the lowest run time. From the results, we observe that there are 19 silent upgrades. So the run time decreases considerably when we compare MSI and MESI protocol. So, there are many new accesses. We also observe that run time decreases when we compare MSI and MOSI protocol. So we can conclude that modified data is being accessed a lot, which results in more cache to cache transfers (18 in MSI to 28 in MOSI). Also, the run time decreases significantly when we compare MOESI and MOESIF protocols. So we can conclude that shared data is accessed again and again, resulting in more cache to cache transfers (42 in MOESI to 53 in MOESIF). Thus, we use Exclusive, Owned and Forwarder state in addition to MSI. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.

So, we conclude that **MOESIF** protocol is the best protocol for this trace.

Part 3] Conclusion:

Cache coherence is nothing but the consistency of shared resource data that ends up stored in multiple local cache. In this project, we have used snooping coherency mechanism. Snooping is a process where the individual caches monitor address lines for accesses to memory locations that they have cached. From the outputs, we observe the following fundamental properties of cache coherence:

i. A processor can read what it wrote, unless there are no intervening writes. This condition is related with the program order preservation. Thus, each processor still works as a uniprocessor.
ii. All writes are globally seen eventually.
iii. All writes are ordered globally in the system.

From the overall results, we observe that **MOESIF** protocol is the best protocol.

MOESIF protocol uses Exclusive, Owned and Forwarder state in addition to the traditional MSI states. Exclusive state allows silent upgrade to M. MOESIF protocol allows dirty sharing of data and reduces traffic due to writebacks. In all the experiments performed above, we observe that MOESIF offers the lowest run time. The run time decreases because of silent upgrades and cache to cache transfers, as we don't have to go back to the main memory to get data.