Multi Core Architecture

Lab Assignment 2
Valid-Invalid Protocol

Group members,
Shriraam Mohan <shriraam.mohan@mailbox.tu-berlin.de>
Tamilselvan Shanmugam <tamilselvan@mailbox.tu-berlin.de>

Due date: 10 Nov 2014, Monday

Cache design:

Cache size: 32 x 1024 B
Block size: 32 B
Number of ways: 8

Number of sets = (cache size) / (Block size * No.of ways) = 128

Calculations from Address:

Parameter	Variable	Bits	Calculation	Comments
Given address	address	m31-m0		
Set Number	set_no	m11-m5	(address & 4064) >> 5	7 bits after the block offset
Tag Number	tag_no	m31-m12	(address & 4294963200U)>>12	20 bits after block offset and set number
Word	word_in_line	m4-m2	(address &28) >> 2	Most significant three bits in the block offset

Address

m31... m12	m11... m5	m4... m2	m1 m0
Tag	Set Number	Word	

pseudo LRU algorithm:

Present state							Replace	Next State						
b6	b5	b4	b3	b2	b1	b0		b6	b5	b4	b3	b2	b1	b0
x	x	x	0	x	0	0	Line_0	-	-	-	1	-	1	1
x	x	x	1	x	0	0	Line_1	-	-	-	0	-	1	1
x	x	0	x	x	1	0	Line_2	-	-	1	-	-	0	1
x	x	1	x	x	1	0	Line_3	-	-	0	-	-	0	1
x	0	x	x	0	x	1	Line_4	-	1	-	-	1	-	0
x	1	x	x	0	x	1	Line_5	-	0	-	-	1	-	0
0	x	x	x	1	x	1	Line_6	1	-	-	-	0	-	0
1	x	x	x	1	x	1	Line_7	0	-	-	-	0	-	0

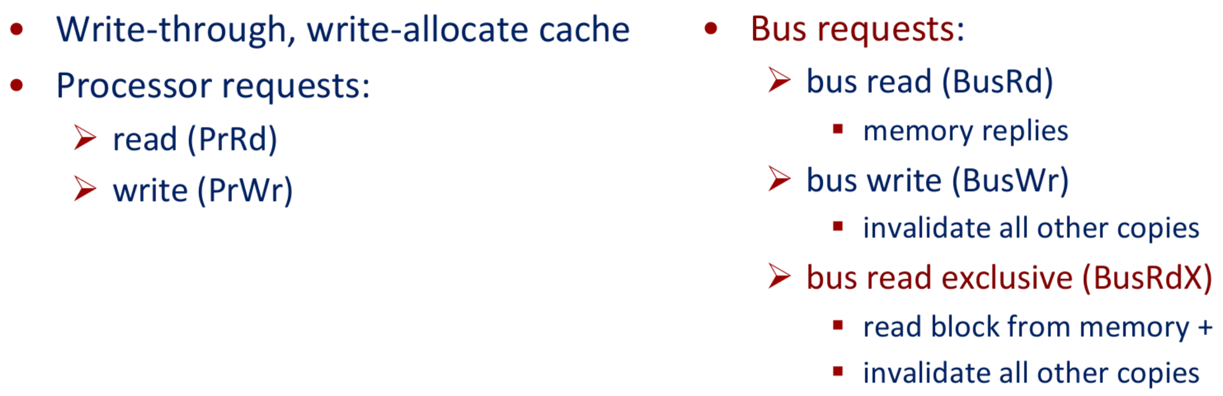
Read operation:

Cache controller determines the line number of the given address. Then checks for the tag bit and subsequently the status bit. Read hit occurs when the tag number is matching and the data is valid. Otherwise its reported as read miss. The data has to be brought from memory.

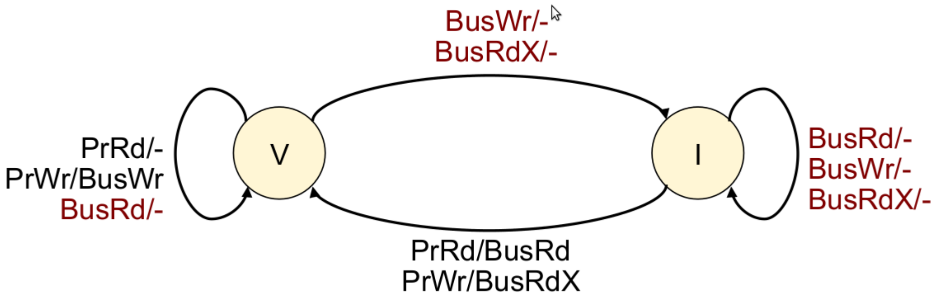
Write operation:

Its almost the same way as reading from cache. Whenever the tag is not matching, it is ou should also investigate what happens to cache hit and miss rates when snooping is deactivated. reported as write miss. Otherwise write hit counter increases its value by 1.

Valid-Invalid Protocol:



- Write-through, write-allocate cache
- Processor requests:
  - read (PrRd)
  - write (PrWr)
- Bus requests:
  - bus read (BusRd)
    - memory replies
  - bus write (BusWr)
    - invalidate all other copies
  - bus read exclusive (BusRdX)
    - read block from memory +
    - invalidate all other copies



Results:

Dbg single processor

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	40	19	21	60	26	34	45.000000

Main memory access rates :

Bus had 21 reads and 26 writes and 34 readX.
A total of 81 accesses.

Average time for bus acquisition :

There were 0 waits for the bus.
Average waiting time per access: 0.000000 cycles.

There were 0 waits to maintain data consistency

0 addresses found while snooping bus requests

Total execution time is 11781 ns, Avg per-mem-access time is 145.444444 ns

Dbg two processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	30	0	30	25	1	24	1.818182
1	32	0	32	35	1	34	1.492537

Main memory access rates

Bus had 62 reads and 2 writes and 58 readX.
A total of 122 accesses.

Average time for bus acquisition

There were 0 waits for the bus.
Average waiting time per access: 0.000000 cycles.

There were 0 waits to maintain data consistency

0 addresses found while snooping bus requests

Total execution time is 10334 ns, Avg per-mem-access time is 84.704918 ns

Dbg four processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	8	0	8	8	1	7	6.250000
1	27	0	27	32	0	32	0.000000
2	43	1	42	38	2	36	3.703704
3	45	0	45	42	0	42	0.000000

Main memory access rates

Bus had 122 reads and 3 writes and 117 readX.
A total of 242 accesses.

Average time for bus acquisition

There were 4 waits for the bus.
Average waiting time per access: 0.016529 cycles.

There were 0 waits to maintain data consistency

2 addresses found while snooping bus requests

Total execution time is 13178 ns, Avg per-mem-access time is 54.454545 ns

Dbg eight processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	6	0	6	4	0	4	0.000000
1	34	0	34	22	0	22	0.000000
2	35	0	35	43	0	43	0.000000
3	39	2	37	46	2	44	4.705882
4	36	0	36	55	0	55	0.000000
5	52	0	52	47	0	47	0.000000
6	48	3	45	51	2	49	5.050505
7	42	1	41	55	5	50	6.185567

Main memory access rates

Bus had 286 reads and 9 writes and 314 readX.
A total of 609 accesses.

Average time for bus acquisition

There were 52 waits for the bus.
Average waiting time per access: 0.085386 cycles.

There were 0 waits to maintain data consistency

0 addresses found while snooping bus requests

Total execution time is 14914 ns, Avg per-mem-access time is 24.489327 ns

FFT_16 one processor

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	**86298**	**7652**	**78646**	**43195**	**10882**	**32313**	**14.312743**

Main memory access rates

Bus had 78646 reads and 10882 writes and 32313 readX.
A total of 121841 accesses.

Average time for bus acquisition

There were 0 waits for the bus.
Average waiting time per access: 0.000000 cycles.

There were 0 waits to maintain data consistency

0 addresses found while snooping bus requests

Total execution time is 15796227 ns, Avg per-mem-access time is 129.646236 ns

FFT_16 two processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	**29313**	**3403**	**25910**	**15417**	**2013**	**13404**	**12.108205**
1	**29262**	**3151**	**26111**	**14801**	**1898**	**12903**	**11.458593**

Main memory access rates

Bus had 52021 reads and 3911 writes and 26307 readX.
A total of 82239 accesses.

Average time for bus acquisition

There were 233 waits for the bus.
Average waiting time per access: 0.002833 cycles.

There were 0 waits to maintain data consistency

0 addresses found while snooping bus requests

Total execution time is 5635864 ns, Avg per-mem-access time is 68.530308 ns

FFT_16 four processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	11274	1742	9532	6553	940	5613	15.044595
1	9417	1457	7960	6083	762	5321	14.316129
2	9834	1531	8303	5523	608	4915	13.928502
3	9881	1498	8383	5972	711	5261	13.934271

Main memory access rates
Bus had 34178 reads and 3021 writes and 21110 readX.
A total of 58309 accesses.

Average time for bus acquisition
There were 541 waits for the bus.
Average waiting time per access: 0.009278 cycles.

There were 0 waits to maintain data consistency

18 addresses found while snooping bus requests

Total execution time is 2242131 ns, Avg per-mem-access time is 38.452572 ns

FFT_16 eight processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	4956	1115	3841	3154	733	2421	22.786683
1	3983	819	3164	2890	545	2345	19.845773
2	3997	881	3116	2703	498	2205	20.582090
3	4043	862	3181	2695	488	2207	20.035619
4	4055	854	3201	2662	477	2185	19.815394
5	4055	855	3200	2733	509	2224	20.094284
6	4074	830	3244	2710	487	2223	19.413325
7	4124	837	3287	2705	465	2240	19.065749

Main memory access rates
Bus had 26234 reads and 4202 writes and 18050 readX.
A total of 48486 accesses.

Average time for bus acquisition
There were 1102 waits for the bus.
Average waiting time per access: 0.022728 cycles.

There were 0 waits to maintain data consistency
35 addresses found while snooping bus requests
Total execution time is 975866 ns, Avg per-mem-access time is 20.126758 ns

Rnd one processor

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	33031	18659	14372	32505	18306	14199	56.404114

Main memory access rates

Bus had 14372 reads and 18306 writes and 14199 readX.
A total of 46877 accesses.

Average time for bus acquisition

There were 0 waits for the bus.
Average waiting time per access: 0.000000 cycles.

There were 0 waits to maintain data consistency

0 addresses found while snooping bus requests

Total execution time is 6285549 ns, Avg per-mem-access time is 134.085991 ns

Rnd two processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	16496	385	16111	16402	393	16009	2.364885
1	24556	865	23691	24804	877	23927	3.529173

Main memory access rates

Bus had 39802 reads and 1270 writes and 39936 readX.
A total of 81008 accesses.

Average time for bus acquisition

There were 178 waits for the bus.
Average waiting time per access: 0.002197 cycles.

There were 0 waits to maintain data consistency

0 addresses found while snooping bus requests

Total execution time is 7405769 ns, Avg per-mem-access time is 91.420218 ns

Rnd four processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	8039	85	7954	8251	89	8162	1.068140
1	20450	0	20450	20221	0	20221	0.000000
2	26553	982	25571	26498	1056	25442	3.841586
3	29600	0	29600	29695	3	29692	0.005059

Main memory access rates

Bus had 83575 reads and 1148 writes and 83517 readX.
A total of 168240 accesses.

Average time for bus acquisition

There were 963 waits for the bus.
Average waiting time per access: 0.005724 cycles.

There were 0 waits to maintain data consistency

10 addresses found while snooping bus requests

Total execution time is 9083356 ns, Avg per-mem-access time is 53.990466 ns

Rnd eight processors

CPU	Reads	RHit	RMiss	Writes	WHit	WMiss	Hitrate
0	4113	22	4091	4030	35	3995	0.699988
1	18318	0	18318	18440	0	18440	0.000000
2	25834	21	25813	25470	31	25439	0.101357
3	29549	1275	28274	28801	1180	27621	4.207369
4	31366	2	31364	30548	1	30547	0.004845
5	32015	12	32003	31782	26	31756	0.059564
6	32327	1626	30701	32285	1653	30632	5.074909
7	32536	1687	30849	32584	1771	30813	5.310197

Main memory access rates

Bus had 201413 reads and 4697 writes and 199243 readX.
A total of 405353 accesses.

Average time for bus acquisition

There were 5118 waits for the bus.
Average waiting time per access: 0.012626 cycles.

There were 0 waits to maintain data consistency

6 addresses found while snooping bus requests

Total execution time is 9748117 ns, Avg per-mem-access time is 24.048464 ns

Effect of snooping deactivation:

As snooping only invalidates the data if its present in its own cache, deactivating will lead to increase the hit rate. Since the data is not invalidated, its no longer consistent with the memory. Processor keeps on consuming the old invalid data without knowing its no longer valid.

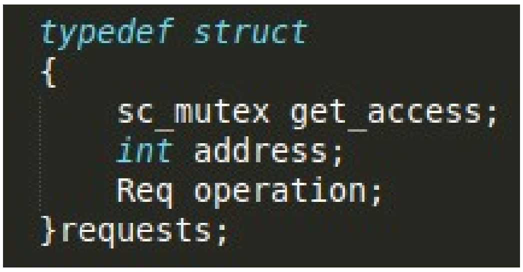
Maintaining data consistency

The following corner cases should be handled to maintain data consistency across processors.

1. READ followed by READX or WRITE

Assume P1 is issues READ request to a block of data and soon after P2 is requesting READX access to the same block of data. After 100 cycles of memory delay, P1 gets data from memory. But after 200 cycles since P2 has issued READX request, it will update the same block. Now P1 will have the old data and P2 have updated one.

One of the solutions to avoid this conflict is waiting for P1 to complete its READ and then serving P2's READX request. We have implemented sc_mutex to wait for another operation to complete. Bus keeps track of issued requests per processor granularity.



```
typedef struct
{
    sc_mutex get_access;
    int address;
    Req operation;
}requests;
```

When a READ request is issued, it gets sc_mutex get_access,registers type of request and addresses. The sc_mutex is released after performing the memory access. If another processor requests READX request in this time period, it waits until first READ operation is completed. Advantages of this method is Bus will not be hogged to maintain data consistency.

This case is extended to READ followed by WRITE, READX followed by READX and READX followed by READ cases. Also we have reported number of such waits to maintain data consistency.

Sample output

```
@209 ns: P3: Write to 67092788
        @209 ns: C3 Write miss
        @209 ns: locked readx mutex 3 addr 67092788
        @209 ns: C3 locked bus with READX req
        @209 ns: Cache 3 snoops own readx req
        @210 ns: C3 released bus from READX req
```