

CSSE3001/7000 2006 Tutorial 6 - Multiprocessor Systems

Discussion questions.

1. To make shared memory viable, your best-case scenario is if you don't share memory. Is this a fair statement?
2. If a multilevel cache doesn't have inclusion, what difference will this make to coherence protocols?
3. What is the role of the snooping hardware? What does it need to keep track of?

Problem-Solving

1. (9.5) [5] <§9.3> Count the number of transactions on the bus for the following sequence of activities involving shared data. Assume that both processors use write-back caches, write-update cache coherency, and a block size of **four** words. Assume that all the blocks in both caches are initially clean. *Bonus question: is there any false sharing? Why (or why not)?*

Step	Processor	Memory Activity	Memory address (word)
1	processor 1	write	100
2	processor 2	write	104
3	processor 1	read	100
4	processor 2	read	104
5	processor 1	write	104
6	processor 2	read	100

2. Consider the initial adds in the parallel add example:

```
sum[Pn] = 0; // Pn is processor no., 0 to 99
for (i = 1000*Pn; i < 1000*(Pn+1); i = i + 1)
    sum[Pn] = sum[Pn] + A[i]; // sum assigned areas
```

a. Is there any genuine or false sharing here?

b. Assuming

- each processor has 64KB of direct-mapped data cache with 32B blocks
- the data caches are initially empty
- no instruction cache misses
- only array references require D-cache accesses

calculate the total misses and invalidations for all 100 processors.

3. Now consider the accumulating adds in the parallel add example:

```
half = 100; // no. of processors
do // serious multiprocessor part starts here
    synch(); // wait for partial sum completion
    if (half%2 != 0 && Pn == 0)
        // if half odd, P0 gets missing element
        sum[0] = sum[0] + sum[half-1];
    half = half/2; // dividing line on who sums
    if (Pn < half)
        sum[Pn] = sum[Pn] + sum[Pn+half];
while (half != 1); // exit with final sum in sum[0]
```

a. Is there any genuine or false sharing here?

- b. Under the same assumptions as in **2**, calculate total misses and invalidations for all 100 processors.
- 4.** Calculate the total time spent on bus transactions in **2** and **3**:
- a. With the given write-invalidate protocol (Fig 9.3.4) and 100 cycles for DRAM read, 20 cycles for all other bus transactions.
 - b. With the write-invalidate protocol modified to:
 - allow a modified block to write back (20 cycles) without being invalidated
 - allow a miss to be satisfied from another cache (20 cycles: no cost to source)