ECE 6100 PR5 REPORT:

Shubhojit Chattopadhyay
GTID : 902694799
ssc3@gmail.com

MOESI Invalidation Based Coherence Protocol

The aim of this project was to implement a MOESI Invalidation based Coherence Protocol for a 4-way shared memory multiprocessor. The implementation is done using basic C++.

Description of Code Flow:

For DM Cache:
A structure (struct cacheline) is defined where the object “line” is a 2D representation of each cacheline. The structure has two members, namely “state” and “tag”.

The code is broken heavily into functions where different paradigms of the state machine has been implemented. Initially all the lines have “I” state and tag were “0xFFFFFFFF”.

A counter is run, which is incremented each cycle. Each trace file is accessed sequentially in a loop. If there is is a match between the counter and the timestamp field of the trace files, the that line is read and the values are stored in variables “timestamp”, “rw” and “address” respectively and the file pointer goes to the next line. If there is no match in the timestamp, the file pointer is moved back to the start of the line of the trace file.

Processor Read/Write logic:
Now, once a line of a processor is getting accessed, the tags of all the others processors (with the same line) are checked for same data using '*initialstatechecker ( )*” function. Now, the tag of the current line and the input address tags are compared. If there is no tag match, the state of that line is made 'I' and respective writeback counter is incremented.

'*newstate*' function comes next which calculates the new state based on current state of the line in the same processor and state of the same indexes in other processors. Based on this, '*setsnoop*' function sets the snoop signal (R = Bus Read, X = BusRdX, U = BusUpgrade, O = Nothing)

To change the state of own processor line, an '*ownstatechanger*' function is used. The current line is then assigned the newstate and the new tag is copied in it regardless of read or write.

Snoop Logic:

All the other processors are checked for their ownstate and based on the snoop signal generated in Processor Read/Write Logic, an action (2 = Flush*, 1 = Flush, 0 = Do nothing) is generated using '*snoopaction*' function.

If there is a tag match in those lines and action = Flush, then writeback counter is incremented again.
If there is a tag match in those lines and action = Flush*, then invalidation counter is incremented.
The state of that line is changed is then changed using *changesnoop ( )* function.

If there is a state change to 'I' due to snoop signal, the invalidation counter is invoked, counting the invalidations.

For Set-Associative Cache:
Everything is the same except the calculations for the tag/index and the fact that my 2D object of cache structure is made 3D, the additional dimension indicating the way. Also, an lru member is added to the structure to indicate lru status. LRU = 0 implies it is least recently used. LRU = 1 implies it is Most Recently used.

Results:

For DM cache:

Cache to cache transfers:
p0p1 p0p2 p0p3 = 31 8 4
p1p0 p1p2 p1p3 = 8 12 4
p2p0 p2p1 p2p3 = 8 55 21
p3p0 p3p1 p3p2 = 16 38 18

Invalidations:
P0: m0 o0 e0 s0 i0 = 11 7 30 10 0
P1: m1 o1 e1 s1 i1 = 15 4 18 12 0
P2: m2 o2 e2 s2 i2 = 6 9 15 3 0
P3: m3 o3 e3 s3 i3 = 9 6 11 4 0

Writebacks:
P0 WB = 55
P1:WB = 85
P2:WB = 102
P3:WB = 97

Number of lines in each state in each line:
P0 Counts: m o e s i =0 29 0 19 208
P1 Counts: m o e s i =44 11 14 75 112
P2 Counts: m o e s i =15 65 8 30 138
P3 Counts: m o e s i =9 55 0 25 167

For Set Associative cache:

Cache to cache transfers:
p0p1 p0p2 p0p3 = 12 9 4
p1p0 p1p2 p1p3 = 10 6 3
p2p0 p2p1 p2p3 = 9 43 11
p3p0 p3p1 p3p2 = 12 23 14

Invalidations:
P0: m0 o0 e0 s0 i0 = 13 6 16 15 0
P1: m1 o1 e1 s1 i1 = 15 5 9 13 0
P2: m2 o2 e2 s2 i2 = 6 7 8 7 0
P3: m3 o3 e3 s3 i3 = 11 5 7 3 0

Writebacks:
P0 WB = 30
P1:WB = 44
P2:WB = 66
P3:WB = 44

Number of lines in each state in each line:
P0 Counts: m o e s i =0 14 0 10 232
P1 Counts: m o e s i =25 8 0 60 163
P2 Counts: m o e s i =8 43 0 19 186
P3 Counts: m o e s i =4 31 0 15 206