1. Read request from L1 data.
   1. Hit
      1. Modified
         1. MESI - Stays M
         2. Pass data to L1 Data
      2. Exclusive
         1. MESI - Stays E
            1. Pass data to L1 Data
      3. Shared
         1. MESI – Stays S
            1. Pass data to L1 Data
      4. Invalid
         1. HIT
            1. MESI – S

Get data from shared bus

* + - 1. MISS (HIT')
         1. MESI – E

Get data from shared bus as RFO

* 1. Miss
     1. HIT/HITM from other processor
        1. MESI – S
        2. Read data from shared bus
        3. Place in L2
        4. Pass to L1 Data.
     2. MISS from other processor
        1. MESI - E
        2. Read data from shared bus as RFO
        3. Place in L2
        4. Pass to L1 Data

1. Write request from L1 data
   1. Hit
      1. Modified
         1. MESI – Stays M
         2. Write to L2 cache
      2. Exclusive
         1. MESI – M
         2. Write to L2 cache
         3. Send invalidate on shared bus
      3. Shared
         1. MESI – M
         2. Read from shared bus with RFO or Invalidate
         3. Write to L2 cache
      4. Invalid
         1. MESI – M
         2. Read from shared bus with RFO
         3. Write to L2 cache
   2. Miss
      1. Read from shared bus with RFO
      2. Write to L2
      3. MESI: M
      4. Pass to L1
2. Read request from L1 instruction
   1. Hit
      1. Modified
         1. Shouldn’t happen because it is an instruction
      2. Exclusive
         1. MESI - Stays E
            1. Pass data to L1 Instruction
      3. Shared
         1. MESI – Stays S
            1. Pass data to L1 Instruction
      4. Invalid
         1. read from shared bus
            1. HIT

MESI – S

Get data from shared bus

Write to L2

Pass to L1I

* + - * 1. MISS (HIT')

MESI – E

Get data from shared bus as RFO

Write to L2

Pass to L1I

* 1. Miss
     1. HIT from other processor
        1. MESI – E
        2. Read data from shared bus
        3. Place in L2
        4. Pass to L1 Data.
     2. MISS from other processor
        1. MESI - E
        2. Read data from shared bus as RFO
        3. Place in L2

1. Snooped invalidate command (Another processor is modifying the data)
   1. Modified
      1. HITM on snoopbus
      2. Write back to shared bus
      3. MESI: I
   2. Exclusive
      1. MESI: I
   3. Shared
      1. MESI - I
   4. Invalid
      1. MESI – Stays Invalid
2. Snooped read request (another processor is trying to read)
   1. HIT/HITM
      1. Modified
         1. HITM on snoopbus
         2. MESI – S
         3. Write back to shared bus
      2. Exclusive
         1. HIT on snoopbus
         2. MESI – S
      3. Shared
         1. HIT on snoopbus
         2. MESI – Stay S
      4. Invalid
         1. Do nothing
   2. MISS
      1. Put miss
         1. Do nothing
3. Snooped write request (another processor is trying to write)
   1. HIT/HITM
      1. Modified
         1. Put HITM on snoopbus
         2. Write to sharedbus
         3. MESI: I
      2. Exclusive
         1. Put HIT on snoopbus
         2. MESI: I
      3. Shared
         1. Put HIT on snoopbus
         2. MESI: I
      4. Invalid
         1. Do nothing
   2. MISS
      1. Do nothing
4. Snooped read with intent to modify(another processor has RFO and we are snooping)
   1. HIT/HITM
      1. Modified
         1. MESI – I
         2. HITM on snoopbus
         3. Write back to shared bus
      2. Exclusive
         1. HIT on snoopbus
         2. MESI – I
      3. Share
         1. HIT on snoopbus
         2. MESI – I
      4. Invalid
         1. Do nothing
   2. MISS
      1. Do nothing
5. Clear and reset all
   1. Invalidate all memory locations. It is not necessary to clear the memory locations.
6. Print contents and state of each valid line
   1. Print contents and states of all(only) valid lines

Notes:

1. Read For Ownership (RFO) – Do a memory read but indicate to snooping processors on FSB our intent to write that memory location (slide 62)
2. Invalidate – “Invalidate your copy and I have the only valid one now.” Address only transaction (i.e.: Puts address on the FSB but doesn’t request data from memory)
3. IMPORTANT: Slide 68
   1. Constraints are met
   2. If miss in both L1 and L2, line is placed in both caches
      1. If a dirty line is evicted from L1 it will be held in L2 (IP).
      2. If a dirty line is evicted from L2 it must also be evicted from L1 (IP)
   3. Miss to L1 and hit to L2
      1. Due to a line being evicted from L1 where it is written back to L2 (IP)
   4. On a snoop hit to L2, L2 instructs L1 to invalidate the line in its cache
      1. Minimize invalidates by keeping single bit/line in L2 to indicate in L1
   5. Relaxed constraints
      1. L2 line size = C \* L1 line size (where C >= 1)
      2. Need additional inclusion bits if C > 1
      3. Associativity L2 / L2 line size >= Associativity L1 / L1 line size
4. IMPORTANT: Slide 65
   1. Has a MESI FSM that was used by PowerPC that incorporates the MESI protocol and the required functions for all cases into the one FSM. FSM and notes found below.

