Final Project

ECE472
Overall Structure
Cache Structure
Cache Control State Machine

- **STATE**: IDLE
  - Dstrobe && DRW
  - Match && Valid

- **STATE**: READ
  - Dstrobe && DRW
  - Match && Valid

- **STATE**: READMISS
  - !Match || !Valid
  - nreadmiss++

- **STATE**: READMEM
  - WCSLoadValue = READ_WAITCYCLES + 1
  - IWSCSig/

- **STATE**: READDATA
  - WSCSig/

- **STATE**: IDLE
  - Dstrobe && IDRW
  - Match && Valid

- **STATE**: WRITE
  - Dstrobe && IDRW
  - !Match || !Valid

- **STATE**: WRITEHIT
  - WCSLoadVal = WRITE_WAITCYCLES + 2
  - nwritehits++

- **STATE**: WRITEMEM
  - nwritehit++

- **STATE**: WRITEMISS
  - IWSCSig/

- **STATE**: WRITEDATA
  - WCSSig/
Cache Controller -- States

- **IDLE**: no memory access underway
- **READ**: Read access initiated by driver; Cache is checked during this state. If hit, access is satisfied from cache during this cycle and control returns to IDLE state at next transition. If miss, transition to READMISS state to initiate main memory access.
- **READMISS**: Initiate memory access following a read miss. Wait state counter is loaded to time the wait for completion of the main memory access. Transition to READMEM State.
- **READMEM**: Main memory read in progress. Remain in this state until wait state counter expires then transition to READDATA state. (Main memory read requires READ_WAITCYCLES cycles to complete)
- **READDATA**: Data available from main memory read. Write this data into the cache line and use it to satisfy the original processor (driver) read request
• **WRITE**: Write access initiated by Driver. If cache is hit, transition to WRITEHIT state. If miss, transition to WRITEMISS state.

• **WRITEHIT**: Cache has been hit on a write operation. Complete write to cache and initiate write-through to main memory. Load wait state counter to time main memory access waiting period. Transition to WRITEMEM state.

• **WRITEMISS**: Cache has been missed on a write operation. Write to cache (cache load) and initiate write-through to main memory. Load wait state timer to time main memory waiting period.

• **WRITEMEM**: Main memory write in progress. Wait for expiration of wait state counter, then transition to WRITEDATA state.

• **WRITEDATA**: Last Cycle of Main memory write. Assert Ready signal to Driver to indicate completion of write.
Cache Controller Signals

- **IDLE**: none
- **READ**: DReadyEnable, DDataOE, Hit (if read hit)
- **READMISS**: Miss, WSCLoad, MStrobe, MRW, DDataOE
- **READMEM**: MRW, DDataOE
- **READDATA**: Ready, Write, MRW, CacheDataSelect, DDataSelect, DDataOE
- **WRITE**: DReadyEnable
- **WRITEHIT**: Hit, WSCLoad, Write, MStrobe, CacheDataSelect, DDataSelect, MDataOE
- **WRITEMISS**: Miss, WSCLoad, Write, MStrobe, MDataOE
- **WRITEMEM**: MDataOE
- **WRITEDATA**: Ready, CacheDataSelect, DDataSelect, MDataOE

Note: Signals Hit and Miss are not shown on the diagrams or used in the implementation of the direct mapped cache. You may use these signals if you find them helpful.

Note: Signals shown in blue appear to be “don’t cares”—i.e., their assertion during the indicated cycle has no effect.
Increasing Cache Size

• The header file Cache.h contains parameters that specify the cache size.

• For a direct mapped cache, be aware of how the size of the index and tag should change given a change in the cache size.
Increasing the Cache Line Size

• The cache line size will be increased from 1 word to 2.
• The size of the memory bus will stay the same, which means you need to do 2 memory reads in the event of a read miss.
• Different for write miss – only 1 mem read.
Additional Info

• Do not need to touch the driver module or the main memory module for this.

• You will need to modify the cache controller.
  – Make sure you understand the FSM that it uses before making any changes.
Modified Cache Control State Machine

Note: This is for conceptual purposes only. Additional states may be needed to differentiate paths.
Extra Credit

• Convert direct mapped cache into 2-way set associative.