NC State University — ECE 406 — Spring 2008
ECE Department — Design of Complex Digital Systems — W. Rhett Davis

Project #2: Cache Design for the LC3 Microcontroller

1. Introduction

The intent of this lab is to improve the LC3 microcontroller you have designed in Project #1. Specifically, you need to create a unified cache. The detailed specification of the cache is described below.

2. Learning Objectives

- Complete a cache design that includes a single level-1 cache for both data and instructions.
- Implement complex message communication mechanisms *using Responsive Handshaking.*

3. Project Report

You are expected to turn in a report after the end of this project. Follow the project report format given on the *Laboratories* page on the course web-site. Be sure to include all items listed in that report format for full credit.

4. Wolfware Submission

You also need to submit your Verilog code electronically through Wolfware as **proj2.v**. This file should contain five modules called *CacheController*, *ProcInterface*, *MemInterface*, *ValidArray,* and *CacheData*, as described below. It may use the *'include* directive to include other files, if you wish, but they must also be submitted with Wolfware. Your code must successfully execute with the test-bench provided on the course web-site (called **proj2test.v**) and give the correct expected output as listed in the test-bench. The memory file (**proj2.dat**) is also given, along with the expected output of the instruction set simulator (**proj2.out**). This program is taken from the example given in class. In addition, a second program will be used to test your code that will not be provided.

Note also that the LC-3 CPU has been provided for you, in the file **lc3.v**. In addition, two of the UnifiedCache blocks are provided in the file **proj2test.v**, including the *CacheRAM* and the top-level *UnifiedCache* module (described below).

Finally, please also ensure that the instance name of your *CacheRAM* (inside the *CacheData* module) is "cram". This will aid us in grading the project.

5. Lab Design: Implementation of a unified cache

In the actual designs, cache sizes are often quite small so that their access times are reduced. In this part of the lab, you will implement a realistic cache. The cache specifications are described as follows:

1) The cache hierarchy is 2-level. There are only a level-1 cache and the off-chip memory.
2) The cache block size is 4 words, so the block offset is 2-bits wide
3) The cache is a direct-mapped cache
4) The size of the cache is 64 (2^6) words. Therefore, there are 16 valid data blocks in the cache at most.
5) The write policies that you need to implement are write-through and write-allocate. As a result, whenever a write-hit happens, the word is written to both the memory and cache. Whenever a write-miss happens, the corresponding block is fetched onto the chip.

5.1 Data transfer specifications

The LC-3 CPU can communicate **only** with the cache. The cache will be in charge of both delivering data from and to the off-chip memory as well as the LC-3 CPU. As a result, whenever the data/instruction required is not in the cache, the cache controller needs to load the data from the memory. The LC-3 CPU is not even aware of the detailed structure of the cache. Figure 1 illustrates the interface relationship between the LC-3 CPU, cache, and the off-chip memory. Note that the LC-3 CPU is unchanged from Project #1, except for the addition of a signal called **macc** (Memory Access). The **macc** signal is "1" whenever the LC-3 CPU is in the *Fetch*, *Read Memory*, *Write Memory*, or *Read Indirect Address* States.

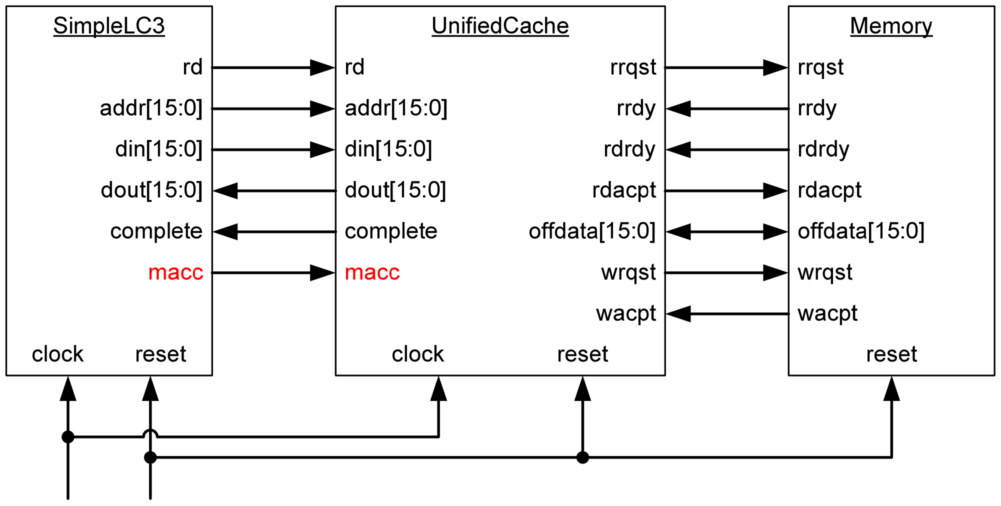

Figure 1. Data transfer interface

The off-chip memory interface signals are the following:

- Read Request (rrqst)
- Read Ready (rrdy)
- Read Data Ready (rdrdy)
- Read Data Accept (rdacpt)
- Write Request (wrqst)
- Write Accept (wacpt)

The protocols under read-hit, read-miss, write-hit, and write-miss are described below.

1) Read-hit: the target data should be fetched in one clock cycle
2) Read-miss:

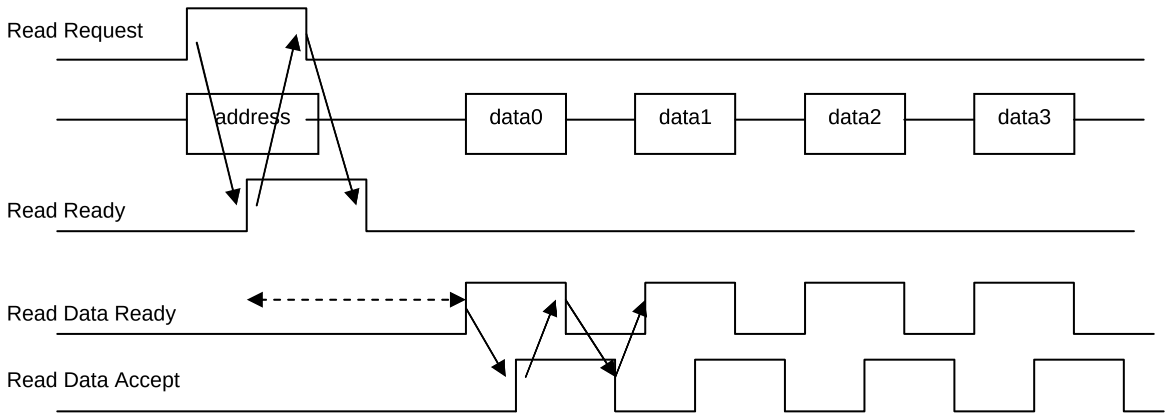

Figure 2. Signal waveforms for a Read-Miss

3) Write-hit:

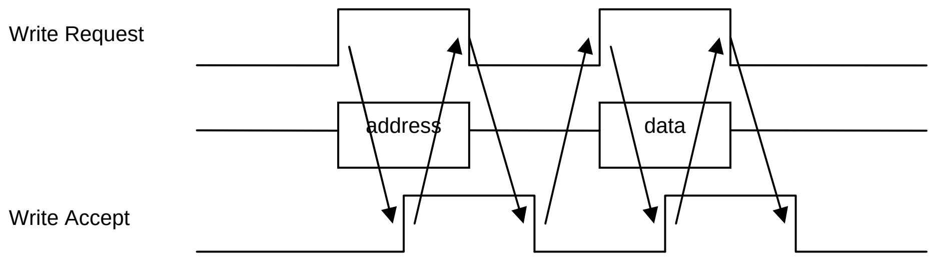

Figure 3. Signal waveforms for a Write-Hit

4) Write-miss:

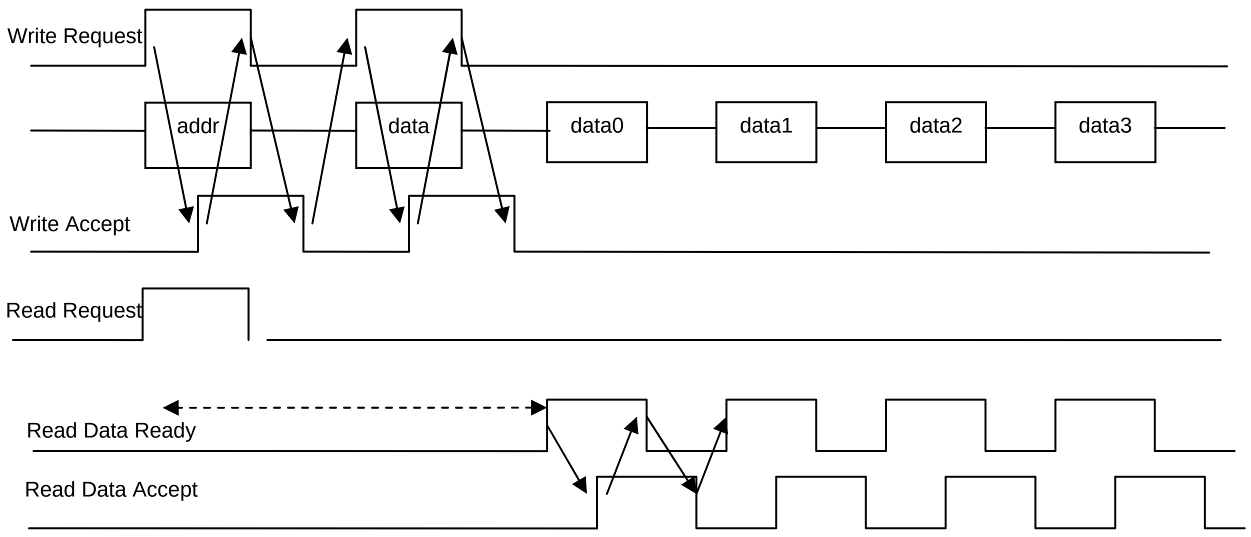

Figure 4. Signal waveforms for a Write-Miss

The solid arrows represent the handshaking and the dotted arrows represent the memory latency. Both the handshaking time and memory latency can change. However, you can assume that the handshaking time is longer than 1 clock cycle and the memory latency is longer than 6 handshaking times. You might notice that during the write-miss protocol, one of the four words loaded from the memory is the same with the one written into the off-chip memory and therefore is redundant. You may try to get rid of it to increase the performance. Such an optimization is not required, however.

5.2 Cache Design Block Diagram

A schematic of the complete cache is shown in Figure 5. The key components are described briefly below.

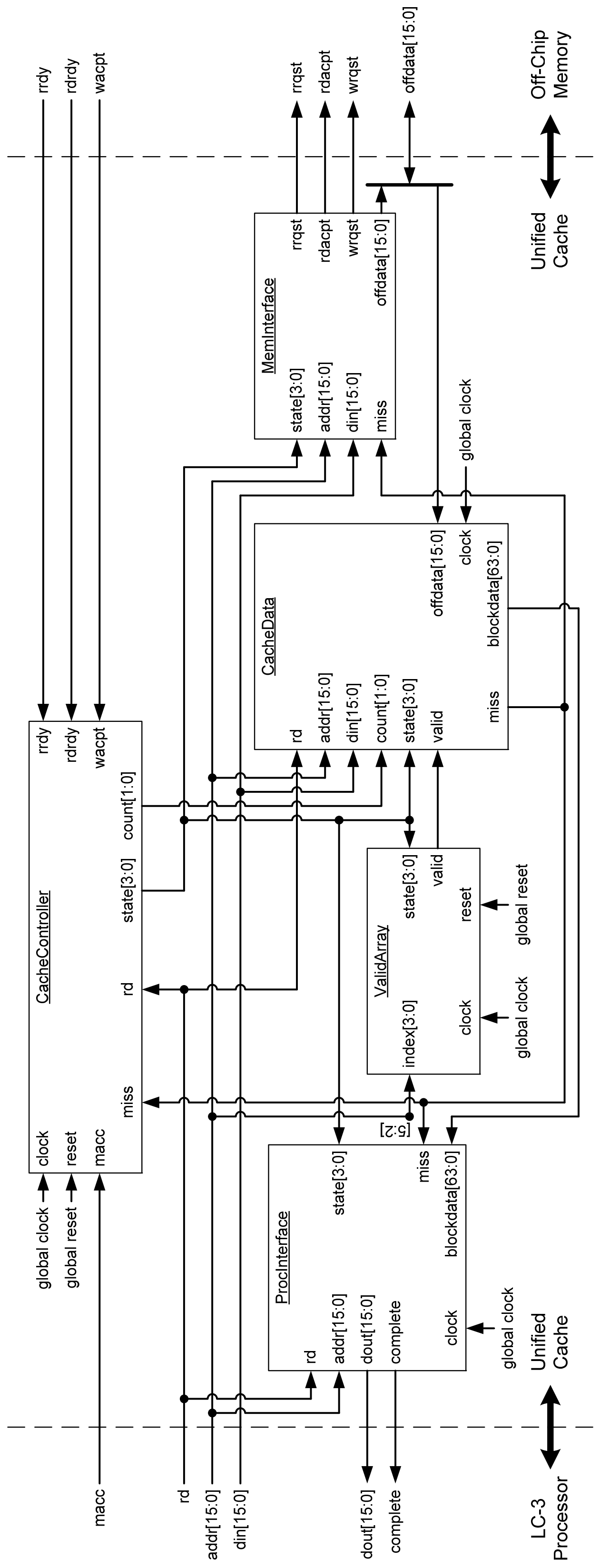

Figure 5. UnifiedCache Schematic

5.2.1 CacheController

This module is a finite state machine that controls the state of the cache. The state transition diagram is given in Figure 6. The state sequences for the 4 possible accesses are:

a) read-hit: 0
b) read-miss: 0 1 2 3 2 3 2 3 2 3 8 0
c) write-hit: 0 4 5 6 7 8 0
d) write-miss: 0 4 5 6 7 2 3 2 3 2 3 2 3 8 0

It takes the following control signals as inputs:

- **clock –** the system clock shared with the CPU datapath
- **reset –** the system reset shared by the entire design including CPU datapath and off-chip memory
- **macc –** memory access enable signal from the CPU datapath, with 1 indicating a memory access is required (from the LC-3 CPU)
- **rd –** 1 indicates a read access and 0 indicates a write access (from the LC-3 CPU)
- **miss –** whether the current memory access is a miss (1) or hit (0) (determined by the CacheData block)
- **rrdy –** read ready (from the off-chip memory)
- **rdrdy –** read data ready (from the off-chip memory)
- **wacpt –** write accept (from the off-chip memory)

The outputs of this module include the variable state and a counter that keeps track of the number of words that have been loaded. The state transition diagram is given in Figure 6.

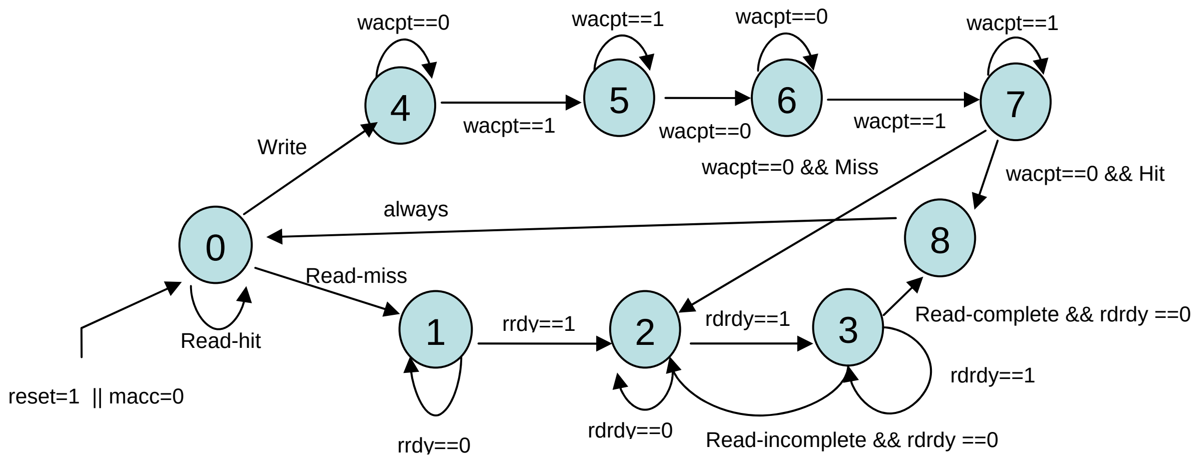

Figure 6: FSM transition graph of the cache controller

5.2.2 ProcInterface

This block communicates with the LC-3 CPU, setting dout[15:0] to correct 16-bit word, based on the the two least-significant address bits and the 64-bit bits from the cache blockdata. A register stores dout[15:0], and this value should be updated when there is a read-hit or when a read-miss operation completes. Ths block should also generate the complete signal when any cache operation completes. A suggested schematic for the ProcInterface block is given in Figure 7.

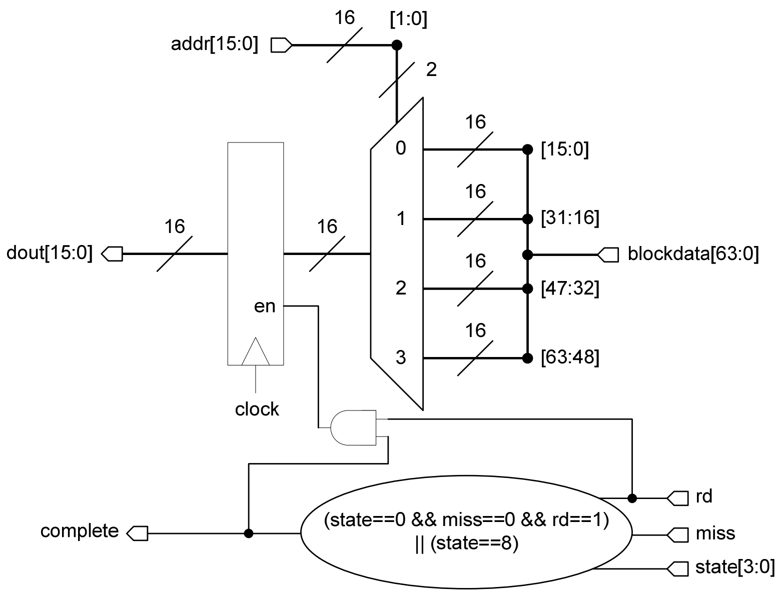

Figure 7: ProcInterface Schematic

5.2.3 MemInterface

This block communicates with the off-chip memory. It contains only combinational logic (no flip-flops). It generates the rrqst, rdacpt, and wrqst signals based on the current state, according to the waveforms in Figures 2-4 and the state transition diagram in Figure 6. It also drives the offdata[15:0] bus with the addr[15:0] or din[15:0] signals from the LC-3 CPU, or is high-impedance when the signal should not be driven.

5.2.4 ValidArray

This module keeps the valid bits of all 16 cache blocks. It contains three inputs:

- **clock –** The global system clock
- **reset –** The global system reset. When reset is 1, the entire valid array should be set to 0.
- **index[3:0] –** The block index that is being accessed. In Figure 5, note that this value is connected to bits [5:2] of the 16-bit address from the LC-3 CPU.
- **state[3:0] –** The current state from the CacheController block. When in state 8, the Valid Array should be updated by setting the valid bit for the current index to 1.

The ValidArray block has only one output, called valid, which should be the valid-bit for the current index. A suggested schematic for the ValidArray block is given in Figure 8.

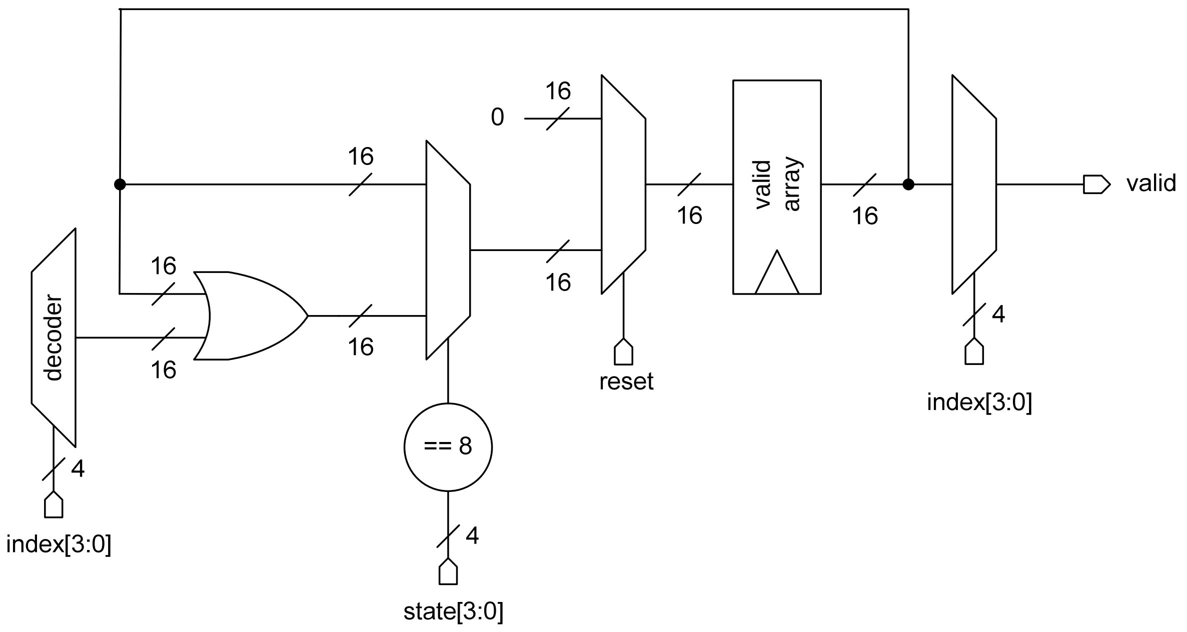

Figure 8: ValidArray Schematic

5.2.5 CacheData

The CacheData block has an instance of the CacheRAM and takes care of reading and writing to the CacheRAM. The CacheRAM is a 16x74 asynchronous memory that has been provided for you in the file **proj2test.v**. The address input of the CacheRAM is connected to the current cache index (bits [5:2] of the address from the LC-3 CPU).

A suggested schematic for the CacheData block is given in Figure 9. It contains the CacheRAM and four 16-bit block-registers that are used to hold the data to be written into the CacheRAM. The block has the following inputs:

- **clock** – The global system clock.
- **state[3:0]** – The current CacheController state.
- **count[1:0]** – The offset count from the CacheController.
- **rd** – The read signal from the LC-3 CPU.
- **valid** – The current valid-bit from the ValidArray.
- **addr[15:0]** – The current address from the LC-3 CPU.
- **din[15:0]** – Data-In from the LC-3 CPU. This value should written into one of the block-registers at the end of state 4, depending on the value of the offset (bits [1:0] of the address from the LC-3 CPU).
- **offdata[15**:0] – Data from the Off-chip Memory. This value should be written into one of the block-registers at the end of state 2, depending on the count from the CacheController.

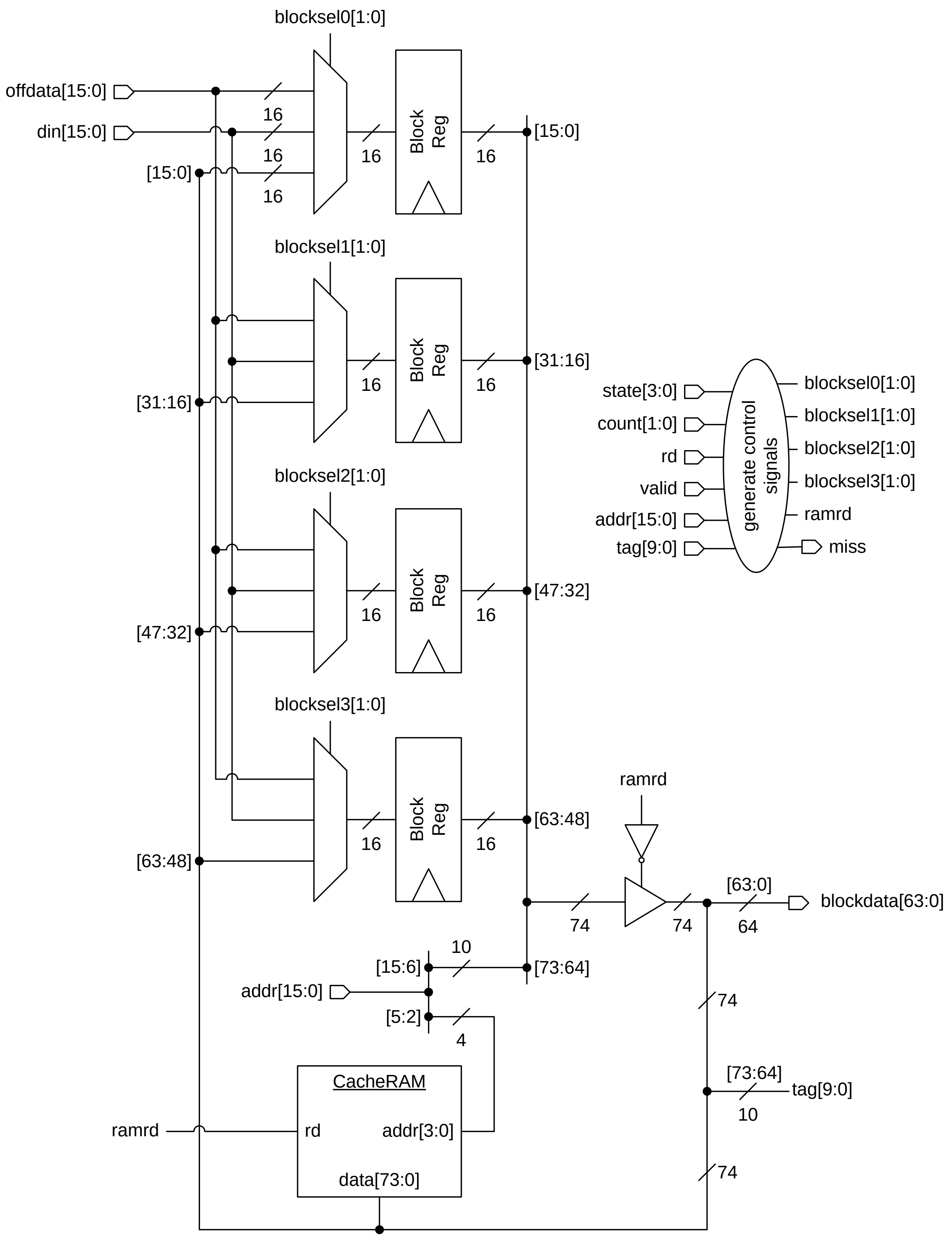

Figure 9: CacheData Schematic

Figure 9 shows five internal control signals that can be generated with from the control signals above:

- **ramrd** – 1 when the CacheRAM is being read, 0 when it is being written. This signal should be 1 at all times **except** the following two cases:
    - The current state is 3 (This will allow writing the values on a cache miss)
    - The current state is 5 and there is a cache hit. (This will allow writing the values on a write hit)
- **blocksel0-3** – These four signals control the values written to the block-registers on each cycle. The current output from the CacheRAM should always be written to the block-registers, **except** for the following two cases:
    - In state 4, the block-register indicated by the offset bits (bits [1:0] of the address from the LC-3 CPU) should get din[15:0] from the LC-3 CPU.
    - In state 2, the block-register indicated by the count from the CacheController should get offdata[15:0] from the off-chip memory.

Lastly, there are two outputs:

- miss – Indicates if there has been a cache-miss, determined by the valid input and a check to see if the tag bits in the CacheRAM match bits [15:6] of the address from the LC-3 CPU.
- blockdata[63:0] – The data in the currently selected block in the cache (sent to the ProcInterface block)