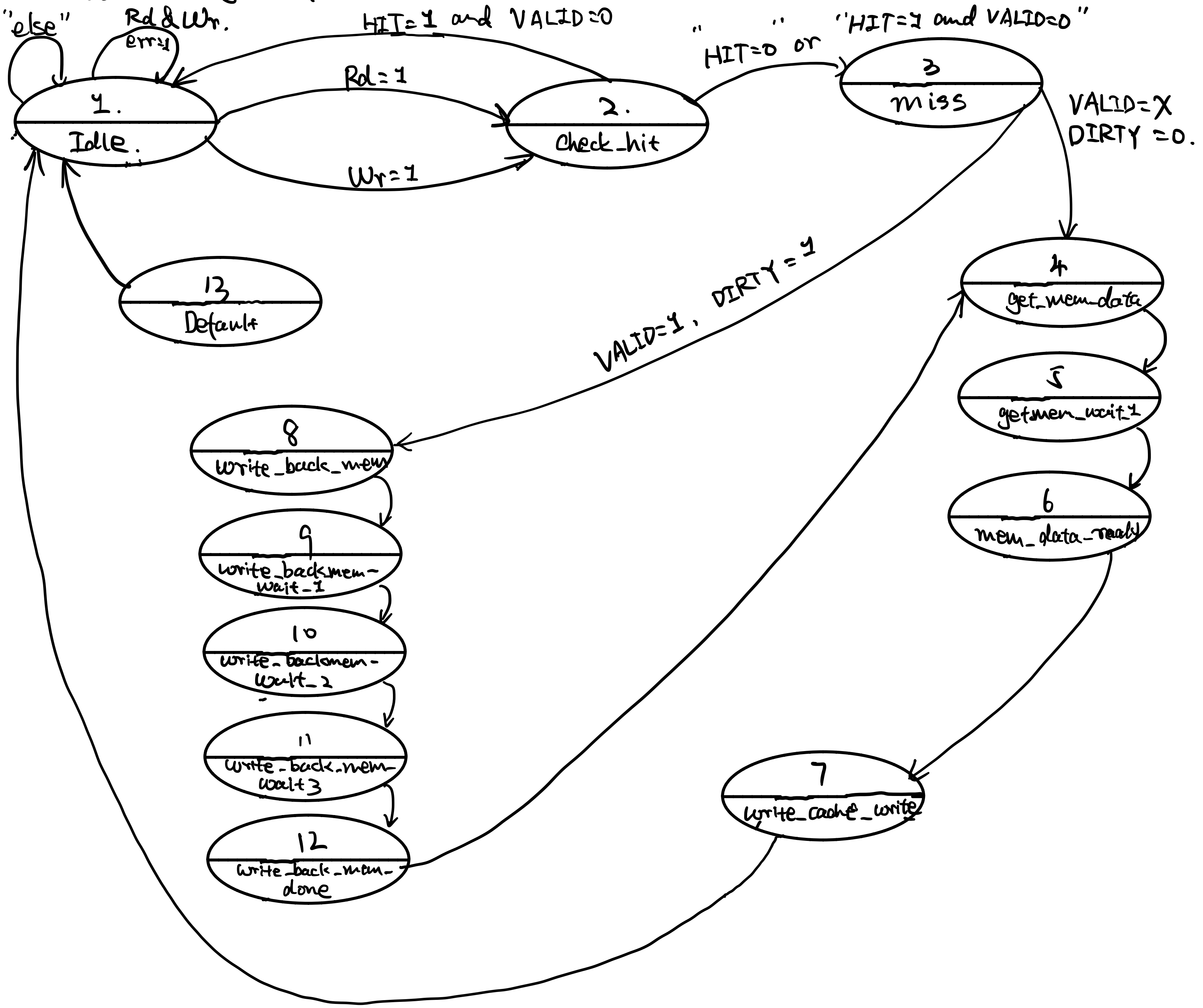
State machine diagram for the 2-way associative



mem_system.v Interface
Provided Output:
DataOut, Done, Stall, CacheHit, err,
Provided Input:
Addr, DataIn, Rd, Wr, createdump, clk, rst

// Cache/Mem module signals are CAPITALIZED
STATES

We chose to use the same FSM for both the direct mapped and 2-way set associative caches
Added assignment/logic on the schematic of the set associative cache to make FSM compatible

----

Set-ups regardless of the state:

- assign cache_TAG_IN = Addr_reg[15:11]
- assign cache_INDEX = Addr_reg[10:3]
- assign cache_OFFSET = Addr_reg[2:0]   // if Addr_reg[0] = 1, set err = 1
- dff Addr_reg = Addr
- assign cache_RD = Rd_reg
- dff Rd_reg = Rd
- dff Wr_reg = Wr
- assign cache_DATA_IN = DataIn_reg
- dff DataIn_reg = DataIn

0. Default Settings:

- Next state: idle
- cache_ENABLE = 0
- cache_COMP = 0
- cache_WRITE = 0
- cache_VALID_IN = X
- mem_RD = 0
- mem_WR = 0
- fsm_err = 0

1. Idle

- If RD & WR = 1
  - Set fsm_err = 1
  - Next state: **idle**

- If RD = 1
  - Set inputs to the cache module
    - cache_ENABLE = 1
    - cache_COMP = 1
  - Next state: **check_hit**

- If WR = 1
    - Set inputs to the cache module
        - cache_ENABLE = 1
        - cache_COMP = 1
        - cache_WRITE = 1
    - Next state: **check_hit**
- Else
    - Next state: **idle**

2. Check_hit
    - if HIT = 1 AND VALID = 0
        - Next state: miss
    - if HIT = 1 AND VALID = 1
        - set DONE = 1
        - if RD_reg = 1: DataOut = cache_data_out
        - Next state: idle
    - if HIT = 0
        - Next state: miss

3. miss
    - if VALID = (0 or 1) and DIRTY = 0
        - Don't need to write back to memory
        - Get data from memory, write to cache, and set DataOut if RD_reg is 1
        - Next state: get_mem_data
    - if VALID = 1 and DIRTY = 1
        - Need to write back to memory
        - After write back, get data from memory, write to cache, set DataOut if RD_reg is 1
        - Next state: write_back_mem

4. Get_mem_data
    - Set inputs to mem module
        - mem_RD = 1
        - ADDR is direct wire, no need to set here
        - Next state: get_mem_wait_1

5. Get_mem_wait_1
    - Next state: mem_data_ready

6. mem_data_ready (Write the value to the cache)
    - Set inputs to the cache module
        - cache_ENABLE = 1
        - cache_COMP = 0

    - cache_WRITE = 1
    - cache_DATA_IN = mem_DataOut
    - cache_VALID_IN = 1
- Next state: wait_cache_write

7. Write_cache_write
   - if RD_reg : DataOut = mem_DataOut
   - DONE = 1
   - Next state: idle

8. Write_back_mem
   - mem_Data_in = cache_Data_Out
   - mem_WR = 1
   - mem_ADDR = {tag_out, index, offset}
   - Next state: get_mem_wait_1

9.Write_back_mem_wait_1
   - Next state: get_mem_wait_2

10.Write_back_mem_wait_2
   - Next state: get_mem_wait_3

11.Write_back_mem_wait_3
   - Next state: get_mem_done

12.Write_back_mem_done
   - Next state: get_mem_data

13. Default
   - Next state: idle