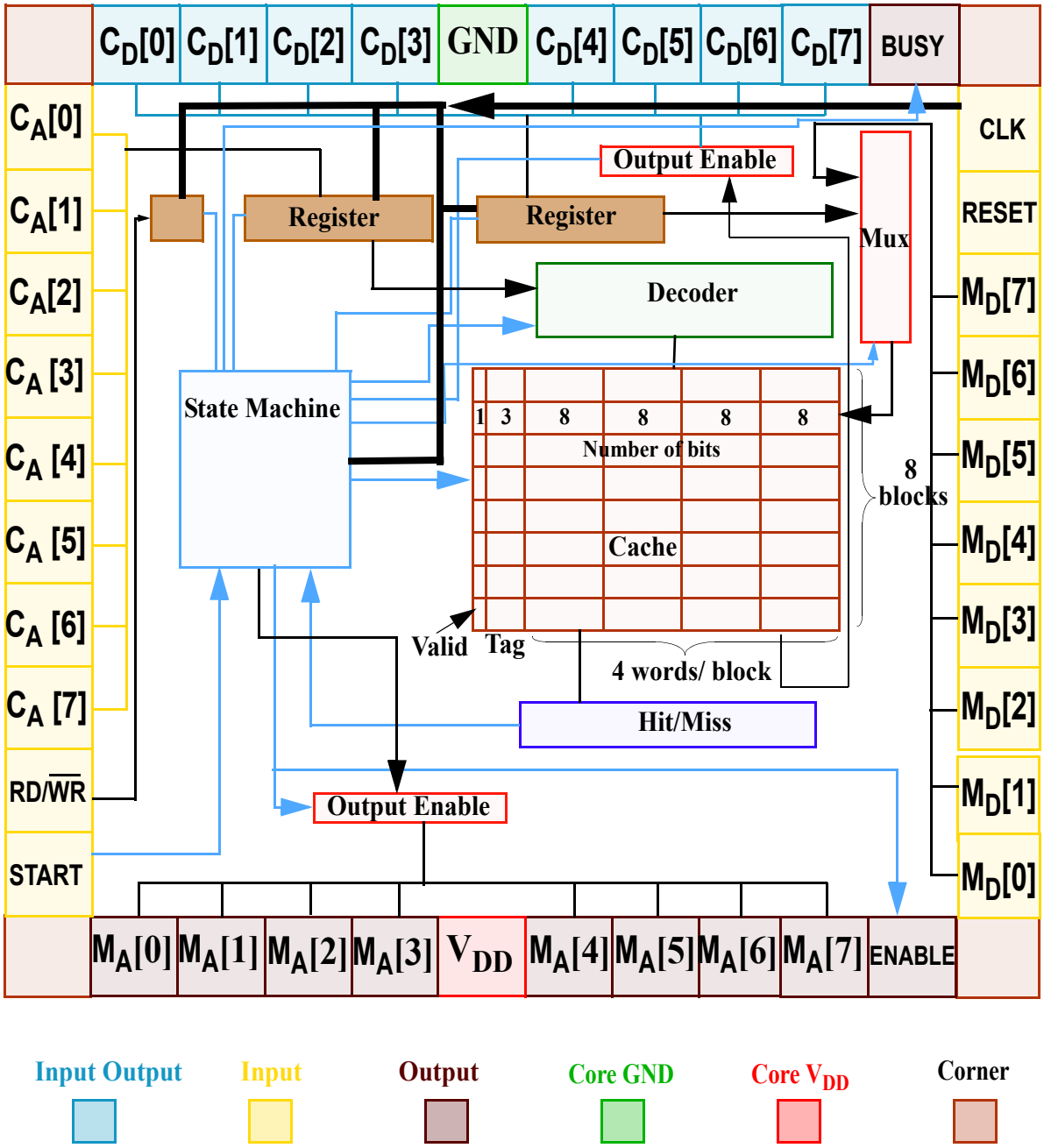
CMPE 413 Project Specification

Project Specification: Cache Design

(Please refer to the webpage for any changes to this specification over the next couple of weeks).

Assigned:Wed, Mar 30th
Due: Last day of class.
Code Submission: In three weeks, details will be provided on the webpage and discussed in class.

Description:



CMPE 413 Project Specification

Details

Your assignment is to design, implement and simulate in VHDL, and do the layout for a cache block described below. The details of a cache and it's use will be discussed in the class.

- Direct Mapped Cache (see extensions for extra credit)
- Byte addressable (8-bit data per location)
- 4 byte blocks, requires 2 address bits to decode
- 8 blocks, requires 3 address bits to decode
- 8 bit address, 3 bit tag
- Write through with no write allocate
- Read hit: 2 clocks
- Write hit/miss: 3 clocks
- Read miss: 19 clocks

Chip interface

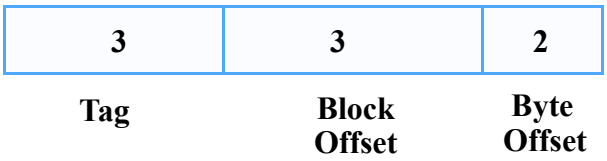
A cache usually needs two address and data interfaces, one to the upper level cache/processor and another to lower level cache/memory. In your design, you have both these interfaces, 8-bit address and data connections to both the CPU and the memory.

CPU Address (C_A):8-bit address input from the CPU for both read and write requests. Address will be provided by the CPU on the rising edge of clock along with the start signal
CPU Data (C_D)*:* 8-bit input/output data bus. Used as input for write requests and output for read requests. Data will be provided on the rising edge of clock for write requests along with the start signal
Start*:* Handshaking signal indicating the start of a read/write request from the CPU, goes high on a positive edge. (Internally your chip works on the negative edge. The address, start, read_write control and data, will be setup half a clock cycle before you start operating on the negative edge)
Busy: Handshaking signal indicating to the CPU that your chip is processing the previous request.
Read_Write (RD/WR): If signal is high CPU is requesting a read operation, if low a write operation
Reset: Master reset to the chip. A high on reset should invalidate all the entries in the cache and reset your state machine to it's reset state. Reset any other registers as required.
Clock (clk): Clock input to the chip. The CPU inputs/outputs and the memory inputs/outputs will be synchronized with this clock signal also.
Memory Address (M_A): Address output to the memory in case of read miss. The last two bits of the address should always be 00 i.e. you provide the address for the last word in the block. The memory controller will automatically increment the address four times and provide you data for the whole block (4 bytes).
Memory Data (M_D): Data input from the memory. It takes the memory 8 clock cycles after getting the enable signal to provide the first byte of data. It will sequentially provide four bytes of data required for the whole block. The first data byte will become valid on the 8th negative edge after asserting enable and will stay stable for 2 clock cycles. The next byte will be provided on the

10th negative edge after asserting enable and will stay stable for 2 clock cycles. The last two data bytes will have similar timing and will be provided on the 12th and 14th negative edges.
Enable: The enable signal is an output requesting the memory to perform a read operation starting at the address provided on M_A. The memory returns four bytes as explained above starting at the 8th clock cycle after asserting this signal.

Operations

The cache has 8 blocks, with four bytes per block. Therefore, you need 3 bits to select one of the 8 blocks and 2 bits to select the correct byte from the block. The remaining 3 bits in the address are used as a tag.



The start signal from the CPU marks the beginning of a operation. The CPU provides the address for read operations and the address as well as data for write operations. All these signals are provided on the rising edge of clock. Your chip internally works at the falling edge of clock so the signals are stable before you need to latch them thus avoiding setup time violations. The signals will be removed by the CPU on the next rising edge of clock. The busy signal triggers the removal of these signals. Busy is an output that should go high on the negative edge after receiving the start signal from the CPU.

Read Hit

For a read operation, the CPU turns start high, provides the address and turns read_write signal high on the positive edge of clock. On the negative edge you need to latch all these required signals and turn on the busy signal. The inputs will be removed by the CPU on the positive edge once it receives a busy signal. You determine whether the data being referenced is in the cache, by comparing the tag bits of the address with the tag bits stored in the block given by the block offset in the address. Also you need to check that the block is valid. Simultaneously you need to read the correct byte from the cache. If the tag and valid checking operation signals a hit, output enable should go high on the next negative edge, the data should be latched in the output register so that it stays stable for one whole clock cycle and the busy signal should be turned off. The CPU will read the data off the data bus on the positive edge. The next operation could be requested on any subsequent clock cycle.

Write Hit

CPU signals follow the same timing as the read, but read_write is set to low and the data to be written is provided. The required inputs should be latched on the negative edge, busy should be turned on and tag/valid compare should be performed. If the result is a hit, on the next negative edge the data should be written to the correct byte in the selected block, busy should still stay high. On the second negative edge after receiving the write request busy should go low signalling the end of the write operation. The CPU will provide a new input on any subsequent clock cycle.

Write Miss

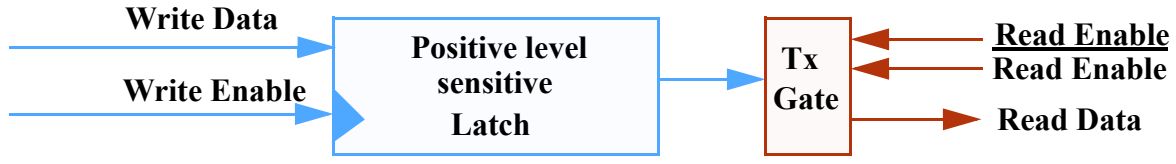
As we are using write through with no write allocate, you don't have to get the block from memory on a write miss. The timing and operations are similar to the write hit case. Only difference is that as you don't have a hit, no write operation should be performed on the cache. The busy should go low after two clocks from receiving the write request as in the hit case and the CPU can provide a new request after busy is low.

Read Miss

The signals are provided by the CPU as described in the read hit operation. On the second negative edge of the clock, you get a miss in this case i.e. the block is not present in the cache. On this edge you keep busy high, turn on the enable signal to the memory requesting a block read and provide the byte address for the last byte in the block that you need. The memory latches the address once it sees enable is high on a positive edge. Enable should be turned low after one clock cycle (third negative edge since receiving the request). The memory needs 8 clock cycles to access the data. After 8 clock cycles, the memory provides the data for the last byte on the negative edge. The data will stay stable for 2 clock cycles. You should setup the correct address for write and update the cache with the new data byte. During the first write operation you should also update the tag and turn on the valid bit. ***Write timing is very critical, if there are any glitches in the address lines or the data lines while the write signal to the cache is enabled, you could write the wrong data or overwrite another location in the cache. Your write signal should be turned on exactly for one clock cycle from the positive edge after the data is valid to the positive edge before the new data byte is provided.*** The address should be stable for 2 clock cycles, similar to the data. The address should be stable for half a clock cycle before the write is turned on and for half a clock cycle after the write is turned off. Once the bytes are written, the correct byte requested by the CPU should be read and latched on the output register, busy should be turned off and the output enabled on a negative edge. Output enable should go low after one clock cycle as in the case of read hit. CPU will read the data on the positive edge after output enable goes high and will provide a new request on any subsequent clock cycle.

Cache Cell

Usually SRAM and CAM (content access memory) cells are used for caches. However, extensive simulations need to be performed to evaluate the effect of loads on these designs. To avoid such situations use the following cell for your cache, which provides, simultaneous read and write capability at the expense of extra area.



Chip Waveforms

The chip timing waveforms explaining the above operations are provided as a separate document.

CMPE 413 Project Specification

Extra Credit (25%)

Make a two-way set associative cache i.e. instead of a 32 byte cache you have a 64 byte cache implementing 2 sets. For block replacement implement a least recently used (LRU) scheme.

Report Requirements and Deadlines

Detailed report requirements will be provided for both the submissions. The first submission (VHDL code, simulation results and imported schematics) will be due in three weeks and the full project submission (corrections required from the first submission, the chip layout, LVS and simulation results) will be due on the last day of classes. During the last Friday session before the final submission we will meet each team to determine progress.

Late penalty will be applied for submissions that don't make the above deadlines. There is an extra credit component to this project worth 25% (that translates to more than half a letter grade in the overall class grade!!!). If you have intentions of doing the extra credit think about how you will integrate it into you design before starting the initial design process. The extra credit part is designed to be relatively easy to integrate into your code and layout once the primary project is done.