EECS 151/251A Homework 8

Due Monday, April 17, 2023

Problem 1: Memory Composition

Neatly draw a block diagram for a 2048×64 single-port RAM using 1024×32 single-port RAMs. You are also allowed to use logic gates and multiplexers. The single-port RAMs have 4 input signals (`addr`, `din`, `ren`, `wen`) and one output signal (`dout`).

Problem 2: Stack

Draw a block diagram for a hardware *stack*, a.k.a. LIFO (Last In First Out) buffer. It has a similar interface as a FIFO with 3 input signals (`ren`, `wen`, `din`) and 3 output signals (`empty`, `full`, `dout`). Assume either `ren` or `wen` can be 1 at a time. The bit-width of data is 32 and we are using a 10×32 single-port synch-read-write RAM as storage (stack is full when it has 10 data). You may also use logic gates, multiplexers, arithmetic blocks, and flip-flops. Remember to annotate multi-bit wires with their bit-widths.

Problem 3: Direct-Mapped Write-Through Cache

Draw a block diagram for a direct-mapped write-through cache using a single-port async-read sync-write RAM for tag and data as well as the valid bits. Cache has 16 lines and the number of data word per line is 1. Both memory address and data words are 32 bit-wide. You may use logic gates, multiplexers, arithmetic blocks, and flip-flops. The interface between memory and cache and between CPU and cache is shown below.

Between memory and cache:

- Read
  - Cache sets `mem_ren` to 1 and `mem_addr` to the source address.
  - Memory reads those signals at the next positive clock edge and starts reading.
  - After a few cycles, memory sets `mem_valid` to 1 and `mem_dout` to the read data.
  - Memory maintains the values of those signals until the next positive clock edge.
- Write
  - Cache sets `mem_wen` to 1, `mem_addr` to the destination address, and `mem_din` to the data to write.
  - Memory reads those signals at the next positive clock edge and starts writing. Cache does not need to wait for the write to finish.

Between CPU and cache:

- Read
  - CPU sets `ren` to 1 and `addr` to the source address.
  - Cache reads those signals immediately, and sets `valid` to 1 and `dout` to the read data in case of cache hit.
  - Otherwise, cache sends a read request to memory, while CPU stalls its datapath.
  - When memory returns the data (`mem_valid` is 1), cache sets `valid` to 1 and `dout` to the read data. CPU resumes opeartions.
  - Cache stores the tag and data at the next positive clock edge.
- Write
  - CPU sets `wen` to 1, `addr` to the destination address, and `din` to the data to write.
  - Cache stores the tag and data. It also sends a write request to memory.

Remember to set the valid bit to 1 when data is stored.

Problem 4: Loop Unrolling

Draw a block diagram for (direct hardware implementation of) $y_i = a * y_{i-1} + x_i$ with loop unrolling of interval 3, where a is constant. $\{x_i\}$ is the input sequence and $\{y_i\}$ is the output sequence. The unrolled circuit takes $\{x_j, x_{j+1}, x_{j+2}\}$ as input and generates $\{y_j, y_{j+1}, y_{j+2}\}$ as output where j is a multiple of 3. Minimize the logic (arithmetic operation) depth.

Problem 5: Pipelining

Write a block diagram for $y_i = y_{i-2} + y_{i-4} + x_i$ using only 2 adders and 4 registers. You must pipeline the connection between the adders i.e. you cannot connect the output of one adder directly to the input of the other. *Hint: when x_i is input, y_{i-1} is output.*