INF2270, Mini-CPU write to memory instruction

P. Häfliger

March 9, 2011

Abstract

In this exercise we will have a closer look at a part of the mandatory exercise

The Task

The Problem

One of the instructions of the Mini-CPU of the mandatory exercise 1 that poses a bit of a challenge and that might cause problems is the one that writes the content of the register into the data memory. Major cause of the problem is that the $\overline{\text{WE}}$ signal (labelled 'R' on the Diglog cell 'SRAM8K') will write to the memory as long as it is low. In other words, it is a level triggered writing operation (i.e. writing proceeds *while* $\overline{\text{WE}}$ is low) and not an edge triggered writing operation (i.e. writing would occure only at the very instance when there is a signal *transition* of $\overline{\text{WE}}$ from high to low).

Thus, one needs to be careful that all other inputs to the RAM are for the total duration *while* $\overline{\text{WE}}$ is low: the obvious way of implementing the write operation where the $\overline{\text{WE}}$ signal is simply the decoded 'write' opcode (figure 1 on the left) fails for the following reason: Since the decoding of the instruction takes a small amount of time the $\overline{\text{WE}}$ will only go high *after* the new instruction is on the instruction bus, and the data memory address here is directly connected to the instruction bus. Thus, there appears a new address while $\overline{\text{WE}}$ is still low, overwriting the memory at a wrong address.

Note that the same problem does not at all occure when writing into the register 'R' because *that* writing operation is edge triggered and one can conveniently use the falling edge of the clock and thus be very sure that all inputs to the register are stable at that very moment, since they change with the rising edge of the clock, i.e. half a clock cycle before.

Find a Patch

This problem can be patched by adding extra combinational logic between the decoded write instruction signal (wr_inst in figure 1) and the actual $\overline{\text{WE}}$ signal, making sure that $\overline{\text{WE}}$ is withdrawn before the new address appears. (Hint: the

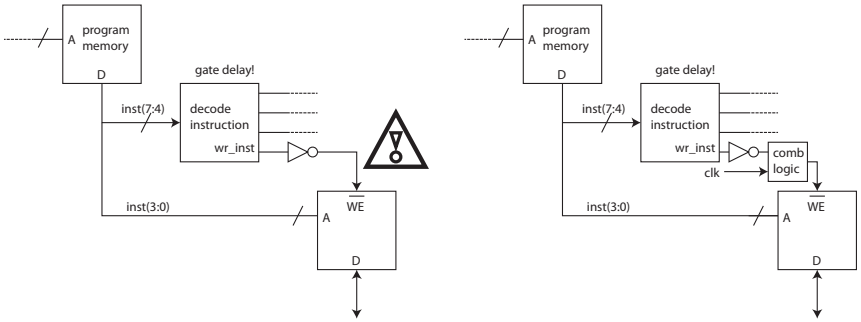

Figure 1: Illustration of the problem (left) and outline of a solution (right)

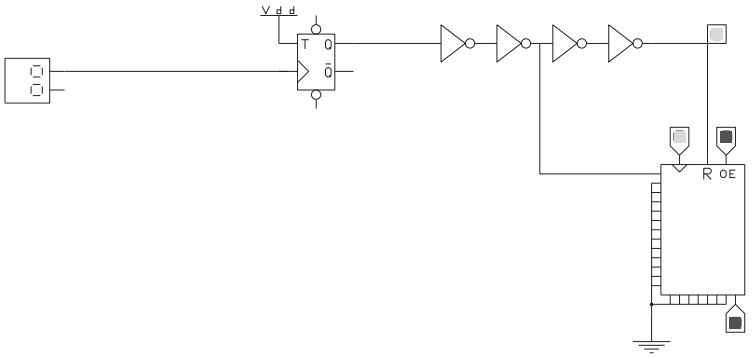

Figure 2: Diglog file to simulate the problem: this circuit *should* only write the byte 1 to memory address 0, but it actually also writes it to memory address 1

program memory does also have a gate delay which is longer than that of a single logic gate. That's an important consideration here!) Try to find that (simple) combinational logic block indicated on the right hand side of the figure 1.

You may use the Diglog schematics in figure 2 to test your solution. Here the toggle flipflop simulates both the write instruction (when its output is zero) and the address (also zero when its supposed to write to the memory). It simulates another instruction that is not supposed to touch the memory (when the toggle flipflop's output is one), where also the instruction operand is one. The first two inverters simulate the gate delay of the program memory and the next the delay of the decoder. Since the decoder's output is delayed with respect to the program memory the data memory is also overwritten at address 1. Try to solve this!