INF2270, exercise L5a: example solution

P. Häfliger

March 3, 2011

Task 1

The top of figure 1 shows the schematic to simulate writing and reading from the memory cell included in Diglog. Figure 3 shows a correct signal sequence. For simplicity sake we show only two address bits (A0,A1, all others are zero), two data bits (D0,D1) and two bits of the input lines (I0,I1).

'_CS' needs to be low for any action to affect the memory and for it to be able to drive the data lines at all. For writting to the memory '_OE' needs to be kept high and '_IE' needs to be low for the inputs to drive the data lines (D7 to D0)). Then as '_WE' goes low the signals on D are stored into the memory. Note that it is unwise to change the address lines A while '_WE' is low! Thus, the signals shown here save the value '??????10' into the memory at address '00000000' and then the value '??????01' onto address '00000011'.

Now by setting '_IE' high, no-one is driving the data lines any longer which briefly show an intermediate value 0.5 on the scope, until '_OE' is set low. Now the data lines D are driven from the memory, first from address 00000011 to the stored '??????01', then very briefly from address '00000010', at which point the data lines go to '??????00' since we have not stored anything at this address and that memory cell is initially at '00000000', and then from address '00000000' where we have stored '??????10'.

Task 2

The bottom of figure 1 and figure 2 show one possible implementation of a 4-bit comparator.

The 'equal' comparison per bit is rather straight forward: the XNOR checks if the bits at this position are equal and the output AND requires that also all more significant bits be equal.

The a>b comparison (written as 'agb') checks if a is greater than b for just the local position (first AND gate), then verifies that the more significant bits have not determined that a<b (second AND), or if a>b has already been determined for a higher order bit (rightmost OR). Note in figure 1 that the MSB input is simply: the MSB is the sign bit and a number is greater which has a

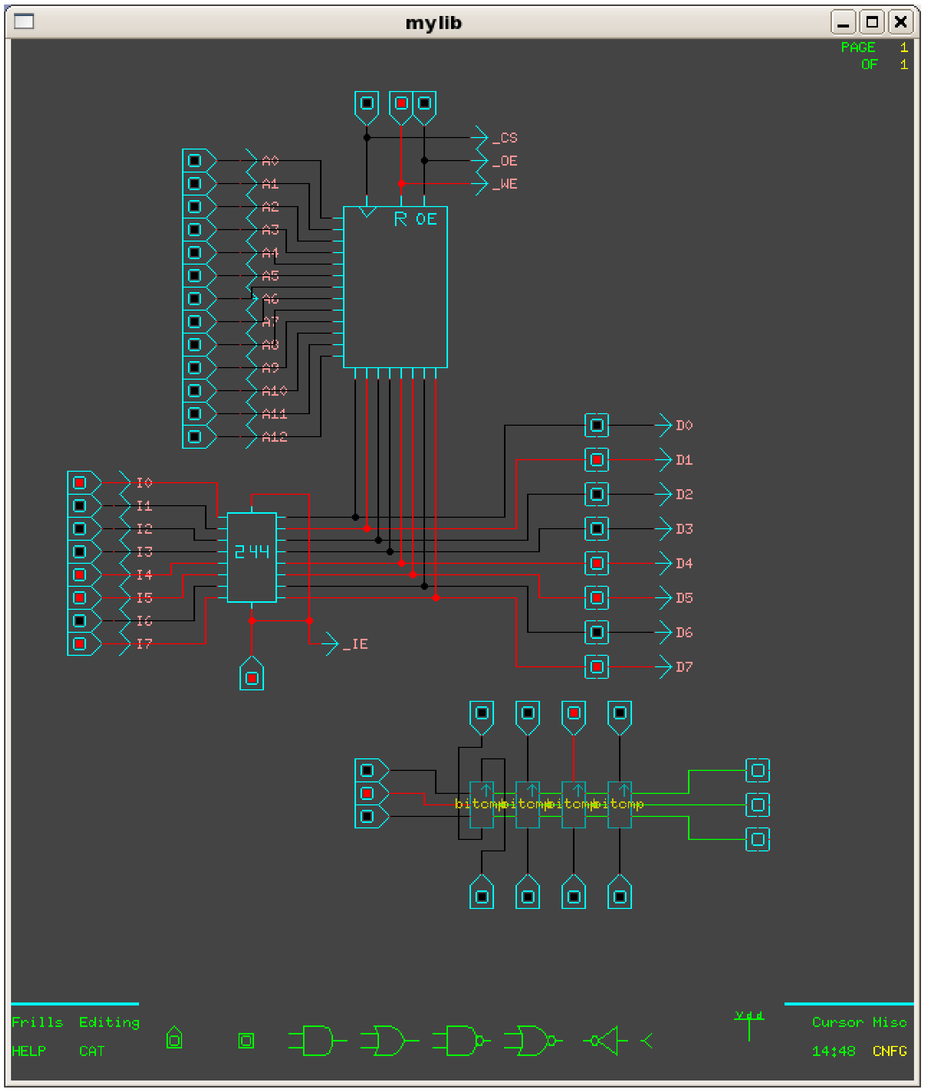

Figure 1: Schematic drawn in Diglog for Task 1 and Task 2. Task two uses hierarchical design, i.e. you need to draw/load the schematic of figure 2 onto another page of Diglog. (Pages are selected by pressing the number buttons '1' to '9'.)

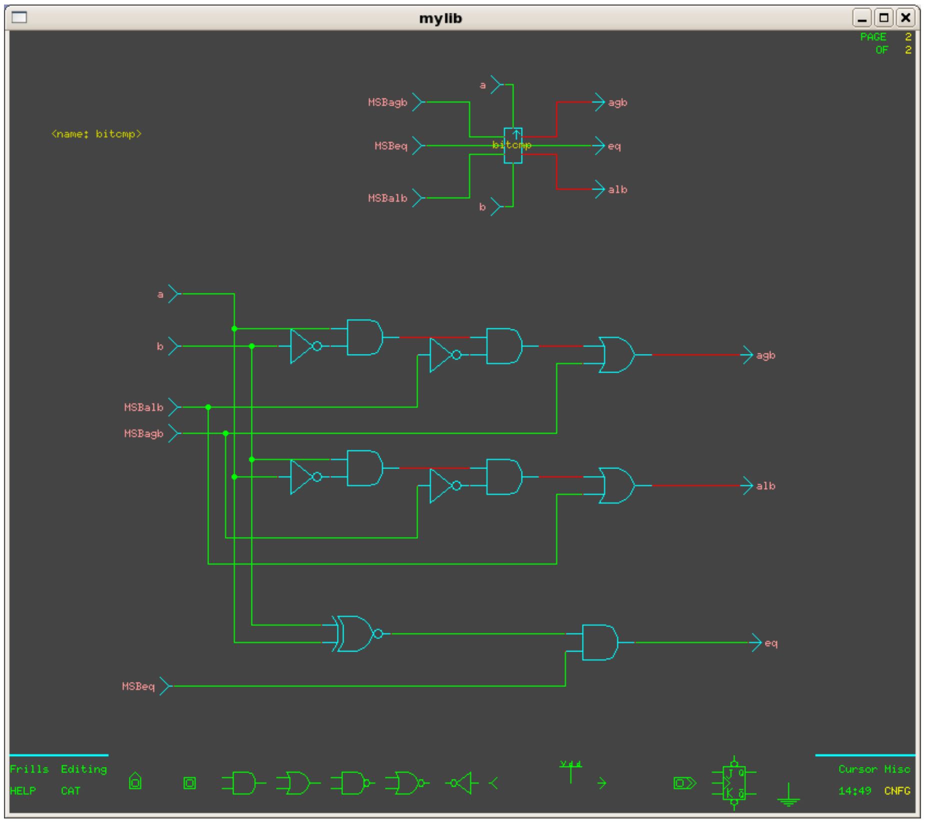

Figure 2: Single bit comparator cascaded to a four bit comparator in figure 1. Note that redundancy is not exploited here: each of the three outputs could simply be 'none of the other two'.

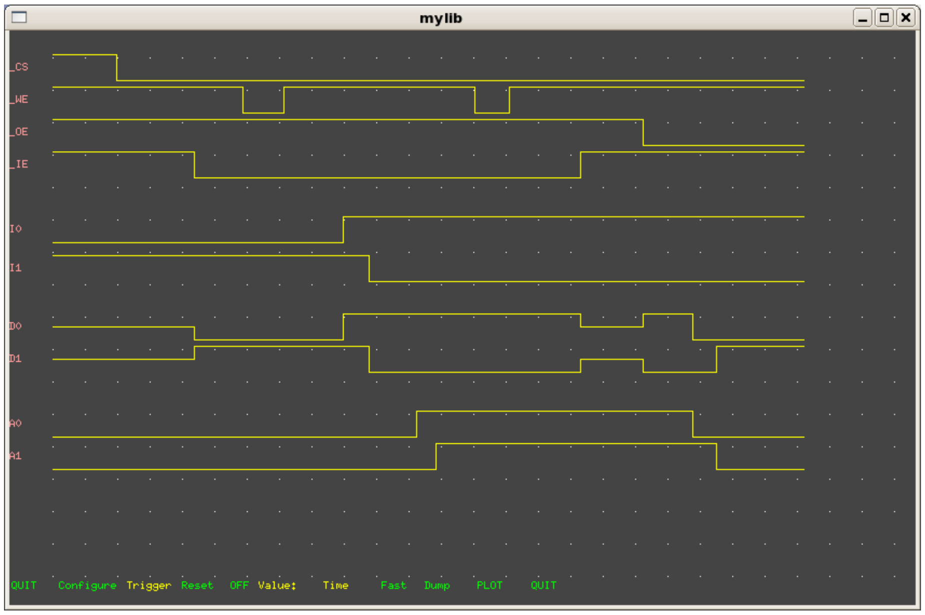

Figure 3: Scope output for writing values into the memory and then reading them. Signal names refere to figure 1.

'0' as MSB if the other has a '1' as MSB. Thus, the MSB 1-bit comparison is reversed as compared to the less significant bits.

Figure 4 shows an implementation that uses the redundancy. The notation is from the ISE tools that have been used before in the course, where bus lines are labelled seperated with commas or indices to letters. The gates are representing n gates in parallel where n is the width of the input bus.

The a>b comparison (written as 'agb') is also implemented slightly differently: it checks if a is greater than b for just the local position (first AND gate), then verifies that all more significant bits were equal (second AND), or if a>b has already been determined for a higher order bit (rightmost OR). Note the difference between signed and unsigned here: the MSB is the sign bit. Therefore that number is greater which has a '0' as MSB if the other has a '1' as MSB. Thus, the MSB 1-bit comparison is reversed as compared to the less significant bits. In figure 4, the MSBs on the input bus are swapped as a consequence. The least siginficant output bits (eq(0) and agb(0)) are the final result.

The a<b (bga) does not have to be computed explicitly but can be derived from the other two since it is then simply equivalent to NOT agb(0) AND NOT eq(0).

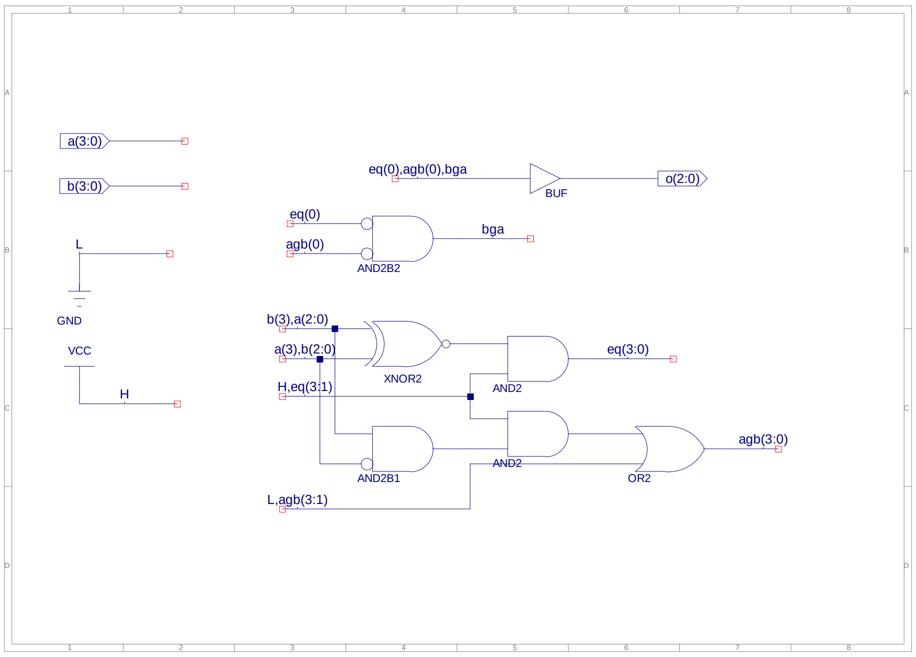

Figure 4: A 4-bit comparator for two's complement binary numbers