Assignment 4 - Logic Design

Due October 10, 1995

These exercises will give you practice in designing microcomputer glue logic using PALs.

Notes

Use an 18CV8 PAL to solve each of the following problems. For each question you should hand in the following:

- a truth table showing the input and output conditions. For sequential devices also show the current state and next state. You may use 'X' to indicate a "don't care" condition and thereby reduce the size of the table.
- a listing of your CUPL `.pld` file with pin assignments and the required equations for all outputs and state variables (the "`.D`" variables). Use the names given in the questions below as the pin names and provide an active-low `RESET*` input for sequential devices.
- a listing of the `.doc` file output from CUPL showing the minimized terms
- the `.si` simulation input file and `.so` simulation output showing that you have tested your design. A minimum set of test vectors are given for each question but you may include more complete tests if you wish.

You may find it useful to define intermediate variables to make the design easier to understand and reduce typing. Intermediate variables can be introduced simply by writing logic equations like the ones that define output variables. Intermediate variables defined on the left hand sides of these equations can be used in subsequent equations.

You tell CUPL that an input is active-low by using '!' before the signal name in the `Pin` statement. *CUPL logic equations are always written using truth values, not the signal levels!*

Note that in the case of the 18CV8 the sums (OR gates) for each output contain a maximum of 8 terms (inputs). If you need to compute an expression with more than 8 terms you can define an intermediate output and then use it as an input in other expressions (although this doubles the propagation delay).

For circuits that require a reset function, use the `.AR` qualifier on the appropriate signals (see examples in lecture notes).

Please include spaces in your ORDER statement between inputs and outputs and between unrelated variables so that the simulation output is easy to read.

Question 1

Design an address decoder using the 18CV8 PAL for a 68000 system with 64 kBytes of EPROM at $20000 and 32kBytes RAM at $0. The addresses should be fully decoded. The inputs will be the address lines A23 to A15, and AS*. The two active-low outputs will be RAM* and ROM*.

Your test vectors should include addresses inside, at the boundaries, and outside both address spaces. They should also include combinations with and without AS* enabled.

Question 2

Modify the address decoder of Question 1 so that initially (after a RESET* input is asserted) the EPROM appears at *both* $0 and $20000 and the RAM is not accessible. After the first memory access at or above $20000, the memory configuration should return to that in Question 1. Your PAL will now also need RESET* and CLK inputs. For this exercise you may consider the address on the address bus valid if and only if AS* is asserted. See Section 6.5 if you are curious why someone might need a circuit like this.

Your test vectors should include a reset operation (with and without clock edges), and memory accesses to addresses at $0, $4, $1FFFF, and $20000 and $4 (in that order).

Question 3

Design a DTACK* (wait state) generator. Figure 4.41 shows a couple of implementations using standard SSI chips. You will design a simple version using a PAL. It's inputs are the 68000's 8 MHz CLK, and one chip select (CS*). The output is DTACK*. Your design should generate DTACK* three clock cycles after CS* goes active. DTACK* should go false at the clock edge where CS* goes inactive.

Your test vectors should include a sequence of clock cycles where the RESET* line is asserted and released and a sequence showing the generation of the delayed DTACK* in at least two successive memory accesses.