MisII is available in the /home/cs/handin/cs3710/bin directory in the CADE lab and /services/classes/cs3710/bin directory on the CS machines as a script called mII which sets things up for you. The misII help command will give you information about all the individual commands. Make sure you use mII to start up the program or the environment variables will not get set correctly.

Use "read_pla" to read a network in espresso format (note that you should use the .ilb and .ob directives to name the input and output literals). An example is:

```
.i 4  
.o 1  
.ilb a b c d  
.ob seg1  
0000 1  
0010 1  
0011 1  
0101 1  
0110 1  
0111 1  
1000 1  
1001 1  
101_  
11__ 
.e
```

You can also use EQN format to write expressions as boolean equations. The full syntax can be found by using "help read_eqn" in mII. If you do that, you get this:

The equations are of the form "<signal> = <expr> ;". For reference, the equation format uses the operators:

```
| ()       | grouping      |
| != (or ^) | exclusive-or |
| ==       | exclusive-nor|
| !         | complement   |
| & (or *)  | boolean-and  |
| _ (or +)  | boolean-or   |
```

As a simple extension to eqntott, juxtaposition of two operands stands for boolean-and, and ‘ used as a post-fix operator stands for complement.

Here's an example of a full adder circuit:
co = a b ci + a b ci' + a b' ci + a' b ci;
sum = a b ci + a b' ci' + a' b ci' + a' b' ci;

You can use the “simplify” and “gd” commands to decompose the circuit, and “source script” to use the predefined reduction script. “pf” will print the factored form of the modified expression and you can draw the circuit based on that factored form. In fact, the general way that I use mII most effectively is:

read_pla (or read_eqn) filename
gd *
source script
pf

If you like, you can map the resulting circuit using the “map” function. There are two libraries of parts similar to the parts in your lab kit that you can use: 3700.genlib uses only gates, and 3700-mux.genlib uses gates and 2-way muxes. Use the “rlib 3700.genlib” command (for example) to read a library into misII, and then “map” to map the function to that library. “pf” and “pg” can be used to see the resulting gate network, and “plot” even makes a crude graph to show what’s connected to what. Check it out.

For CS3710 users, note that these parts are not exactly what’s available in the Xilinx libraries, but they are pretty similar. I’m not sure that mapping is all that useful, but if you do find it useful, the gates in the 3700.genlib library are certainly available in the Xilinx library for your use. Of course, there are more parts in the Xilinx library than the 3700.genlib library will use. You can almost certainly optimize further from the map output.