University of Toronto, Faculty of Applied Science and Engineering
Department of Electrical and Computer Engineering

ECE 1387S - CAD for Digital Circuit Synthesis and Layout   Handout #12

Exercise #2
Using SIS - The Berkeley Sequential/Combinational Logic Synthesis System

November 1999   J. Rose

Assignment Date: November 3
Due Date: November 17 (before lecture begins)
Late Penalty: **-1 mark per day late, with total marks available = 10**

The purpose of this exercise is to gain familiarity with a very powerful and wide-ranging logic synthesis system called SIS. SIS is a framework that contains code from much of the sequential and combinational logic synthesis research that has been done at Berkeley and elsewhere. We will use it to focus mainly on the lecture material dealing with multi-level logic synthesis, but it also contains sequential synthesis such as Finite State Machine optimization and encoding, re-timing, as well as two-level logic optimization, technology mapping into a library of gates, technology mapping into lookup tables/Xilinx FPGAs, and Actel FPGAs. If you ever do work in logic synthesis, it is not very difficult to put your code inside the SIS framework and thus save the pain of a lot of I/O bother and be able to leverage the optimization code that is already there.

SIS is available on EECG in /usr/src/local/sis-1.2/bin/sis and on ECF in ~jayar/1387/e2/sis-1.2/bin/sis. It will only run on the ECF sparcstations, not skule.ecf.

This exercise consists of the following:

1. Read the manual for SIS (handout #13), and focus particularly on the multi-level logic optimization commands. You should read the entire manual to get a sense of all the different kinds of logic synthesis that SIS covers.

2. Give the truth table for the following five-input (inputs: $\mathbf{n_4}$,$\mathbf{n_3}$,$\mathbf{n_2}$,$\mathbf{n_1}$,$\mathbf{n_0}$,) four-output (outputs: $\mathbf{f_1}$, $\mathbf{f_2}$, $\mathbf{f_3}$, $\mathbf{f_4}$) logic function. Assume that the five inputs together make up a base 10 number, **N**, which ranges from 0 to 31.

   Output $\mathbf{f_1}$ is 1 when **N** is even, except for the cases x= 6 and x=26 (assume 0 is even).
   Output $\mathbf{f_2}$ is 1 when **N** is odd, except for the cases x= 9 and x=23.
   Output $\mathbf{f_3}$ is 1 when **N** is evenly divisible by 3 (0 is not divisible by 3).
   Output $\mathbf{f_4}$ is 1 when **N** is evenly divisible by 4 (0 is not divisible by 4).

   Create an input file in the Berkeley **pla** format which is is basically a multi-output truth table. You can find an example file in the **pla** format in the file ~jayar/1387/e2/alu4.pla. The “-” entries mean “don’t care”, for both input and output.

3. **Digression:** This part of the exercise is about two-level logic optimization. If you were to write the sum of products expression for all four logic functions, how many distinct minterms would you use? That is, count the number of rows of your truth table which have a least one "1" in the four outputs.

   i. Run the SIS program, and read in your file using the **read_pla** command.

   ii. Run the **espresso** command to optimize this two-level logic.

   iii. Write out the resulting optimized pla using the **write_pla** command.

   How many distinct product terms are needed for the four functions after optimization by espresso?

4. Turn your logic function into a multi-level logic network - read in your file using the **read_pla** command.

   For each of the following optimized forms of the network, use the **print_stats** command to determine the number of nodes, the number of literals in some_of_products form, the number of literals in factored form. Show these literal counts in the form of a table. Use the **print_factor** command to obtain the actual factored equations.

   i. The un-optimized two-level expression.

   i. The espresso-optimized two-level expression.

   ii. The multi-level network optimized using only the script script.algebraic. (This and other scripts below can be found in the directory /usr/src/local/sis-1.2/sis/sis_lib on the eecg and in ~jayar/1387/e2/sis-1.2/sis/sis_lib on the ECF system. You invoke a script using the **source** command.)

   iii. The multi-level network optimized using only the script script.boolean.

   iv. The multi-level network optimized using only the script script.rugged.

   Inspect the factored form of the equations and comment briefly on the results.

5. Create a script of your own that does better than the best factored-form literal count result of question 4.