For this assignment, when you are asked to use SIS for optimization, restrict yourself to the following operations:\footnote{Help is available for all these commands; you can save yourself a lot of typing by making use of aliases and paths. Run \texttt{alias} at the prompt, also take a look at \texttt{adnan/.sisrc}.}

\textbf{sweep} (performs constant propagation and inverter minimization)

\textbf{el} (performs elimination)

\textbf{fx} (performs kernel–based common logic extraction)

\textbf{decomp} (performs a decomposition of the network nodes)

For you own edification, you may want to read about/play with the following commands: \texttt{simplify, full_simplify, print_delay, reduce_depth, tech_decomp, tech_map}.

You can see the results of each operation by typing \texttt{print_stats} at the prompt; you may want to \texttt{set auto_exec print_stats} initially, which will run \texttt{ps} automatically after each command. The network can be printed using \texttt{write_blif, write_eqn, write_pla, print_factor}.

Report the best result (i.e., least number of literals) and the steps you took to get there.

1. (Problem 15, Chapter 10 Hachtel & Somenzi) Perform weak division of $F = abrs + abrt + abd + abc + abu + ghrs + ghrw + ghg + ghe + ghu + dp + ep + rstuw$ by $D = ab + gh$.

   \textbf{10 marks}

2. (Problem 12, Chapter 10 Hachtel & Somenzi) Run SIS on the function $F$ given above. You may input the function in either \texttt{eqn} or \texttt{blif} format, but the output should be in \texttt{blif}. (The optimum result contains 23 literals; show your steps.)

   \textbf{10 marks}

3. For the function $F = uwxy + uvxy + tx + tz + rsw + rsr + vwxy$, compute all the level-0 kernels.

   \textbf{10 marks}
4. (Problem 4, Chapter 8 deMicheli’s book.) Consider the logic network defined by the following expressions.

\[ x = ad' + a'b' + a'd' + bc + bd' + ac \]
\[ y = a + b \]
\[ z = a'c' + a'd' + b'd' + e \]
\[ u = a'c + a'd + b'd + e' \]

Draw the logic network graph. Outputs are \{x, y, z, u\}. Perform the weak division \( f_x/f_y \) and show all steps. Substitute \( y \) into \( f_x \) and redraw the network graph. Compute all kernels and co-kernels of \( z \) and \( u \). Extract a multiple-cube subexpression common to \( f_z \) and \( f_u \). Show all steps. Redraw the network graph.

25 marks

5. In this problem, you are to devise an 8-bit ALU in Verilog and run it through algebraic optimizations available in SIS.

The ALU should take two 8-bit vectors data arguments, and a two bit control argument, corresponding to the operation to be performed. The output is a single 8-bit vector. The functionality is given in Table 1.

<table>
<thead>
<tr>
<th>ALU Operation Mode</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>( in1 ) AND ( in2 )</td>
</tr>
<tr>
<td>01</td>
<td>( in1 ) XOR ( in2 )</td>
</tr>
<tr>
<td>10</td>
<td>( in1 ) OR ( in2 )</td>
</tr>
<tr>
<td>11</td>
<td>( in1 ) PLUS ( in2 )</td>
</tr>
</tbody>
</table>

Write this design in the Verilog HDL. Compile it to the \texttt{blif\_mv} format using the \texttt{vl2mv} compiler. Read it in \texttt{VIS} using the \texttt{read\_blif\_mv} command, and write it out in the \texttt{blif} format using the \texttt{write\_blif} command. (You may want to verify your design works by using the \texttt{sim} command in \texttt{VIS}.)

SUN binaries for SIS, \texttt{vl2mv} and \texttt{VIS} are available at

(a) \texttt{/home/projects/ece/verif/vis-1.2/vis}
(b) \texttt{/home/projects/ece/verif/vis-1.2/vl2mv}
(c) \texttt{/home/projects/logic_synthesis/solaris/bin/sis}

50 marks