VLSI CAD: LOGIC TO LAYOUT:
SIS Software Package

Rob A. Rutenbar
University of Illinois at Champaign-Urbana

SIS[1] is a multi-level logic optimization tool developed by researchers at the University of California at Berkeley. An earlier version of the tool called **MIS** was hugely successful and influential in the VLSI CAD domain. SIS extended the functions of MIS and is the most recently available version of the tool.

As we noted in the lectures, all of multi-level logic optimization is based on scripts that heuristically optimize a Boolean network model of your design. Thus, SIS has many commands and many options. For simplicity, we are using a standard, default synthesis script called the RUGGED script to optimize your designs.

SIS can also read logic design in many different file formats. But conveniently for us, it can read the same format that the ESPRESSO 2-level optimizer tool uses, the so-called PLA format. So, we will let you edit files in the ESPRESSO truth table format, and those can be uploaded and optimized by SIS.

The new information you need is how to read a SIS output result, since the result is an optimized Boolean network model, each of whose nodes is an optimized 2-level SOP form. Let's look at a few small examples to see how to read a SIS result.

EXAMPLE 1: 2 functions of 4 variables
Here are two functions named **s1** and **s0**, each a function of input variables **a1 a0 b1 b0**. This is specified in the **ESPRESSO** PLA format:

```
.i 4
.o 2
.ilb a1 a0 b1 b0
.ob s0 s1
0000 00
0001 00
0010 00
0011 00
0100 00
0101 01
0110 10
0111 11
1000 00
1001 10
1010 01
1011 00
1100 00
1101 11
1110 00
1111 00
.e
```

This is the result of optimizing this design with **SIS**, as it will appear in your Coursera SIS assignment window:

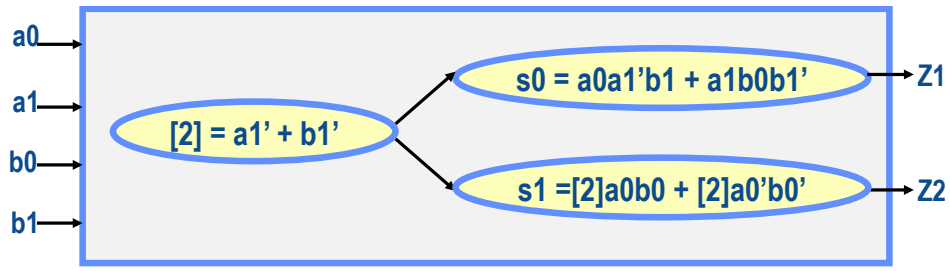
Submission	
Submission time	Wed-20-Mar 14:17:09
Raw Score	1.00 / 1.00

Feedback	**SIS Standard output**
	`{s0} = a0 a1' b1 + a1 b0 b1'`
	`{s1} = [2] a0 b0 + [2]' a0' b0'`
	`[2] = a1' + b1'`
	SIS Standard error

Two important things to understand about SIS output:

- Primary outputs of the Boolean logic network appear in **{ }** braces. So the outputs are written as **{s0}** and **{s1}** here.
- SIS can create new, intermediate nodes in this network (e.g., by factoring! which you know how to do!). Those nodes appear in **[ ]** brackets with arbitrary numbers labeling them. So the new intermediate node extracted in this optimization is **[2]**.

If we draw this network as we did in our lectures, it looks like this:



For "measuring" the complexity of the network, we would count as follows:

- This network has 3 nodes in its Boolean logic network.
- This network has 14 literals. Count *every variable* on the right hand side of an "=" in *every* node.
- This network has 4 AND gate product terms in it. (2 SOP forms that each have products with 2 or more literals).

EXAMPLE 2: Simple 2-bit multiplier

Suppose we have input variables **a1 a0** and **b1 b0**, and we regard these as unsigned 2-bit integers. We multiply them, and compute the 4-bit product **p3 p2 p1 p0**.
This is specified in the **ESPRESSO** PLA format:

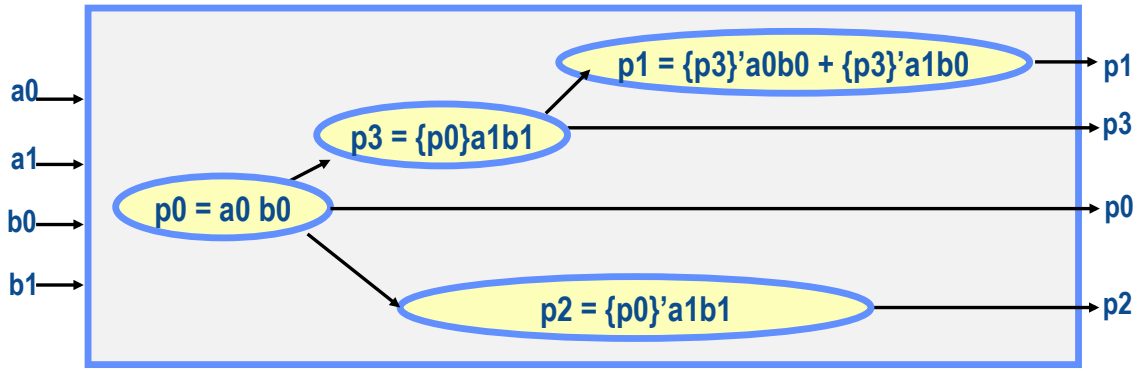
```
.i 4
.o 4
.ilb a1 a0 b1 b0
.ob  p3 p2 p1 p0
0000 0000
0001 0000
0010 0000
0011 0000
0100 0000
0101 0001
0110 0010
0111 0011
1000 0000
1001 0010
1010 0100
1011 0110
1100 0000
1101 0011
1110 0110
1111 1001
.e
```

Here is what the SIS output looks like for this multiplier example:

Submission	
Submission time	Wed-20-Mar 14:24:05
Raw Score	1.00 / 1.00

Feedback	**SIS Standard output**
	`{p3} = {p0} a1 b1`
	`{p2} = {p0}' a1 b1`
	`{p1} = {p3}' a0 b1 + {p3}' a1 b0`
	`{p0} = a0 b0`
	SIS Standard error

The result drawn as a Boolean network diagram is this:



Observe that SIS is free to use any node, as an input to any other node. In this case, even the nodes that make the primary outputs, get used as inputs to other nodes.

For "measuring" the complexity of this network, we would count as follows:

- This network has 4 nodes in its Boolean logic network.
- This network has 14 literals. Count *every variable* on the right hand side of an "=" in *every* node.
- This network has 5 AND gate product terms in it. (2 SOP forms that each have products with 2 or more literals).