

## Espresso Tutorial

The Espresso logic minimization software can help in the design of large digital logic designs, particularly those with Boolean functions that have more than 4 inputs or with multiple Boolean functions. It is available on the linuxpool as the command `espresso`.

### Input File Format

Espresso uses the PLA file format. A file in PLA format lists all the inputs and their corresponding outputs.

#### *Example 1*

In this example we define a function of three inputs.

```
# this is a comment
# three inputs
.i 3
# one output
.o 1
.p 8
000 1
001 0
010 1
011 1
100 0
101 0
110 -
111 -
.e
```

Lines that begin with `#` are comments and are ignored by espresso. The line `.i 3` tells espresso that there are 3 inputs to our function and the line `.o 1` tells espresso that there is one output. The line `.p 8` tells espresso that there are 8 product terms in the following truth table and the line `.e` indicates the end of the file.

The lines between the `.p` and `.e` contain the truth table for the function. In this example, the first 3 numbers contain the input values and the last number (after the space) contains the value of the function for these inputs. Dashes indicate don't cares and may appear in inputs or outputs. In the above example, both the 110 and 111 input cases are don't care cases.

### **Example 2**

We can also give names to our inputs and outputs with the `.ilb` and `.ob` lines as shown below.

```
# three inputs
.i 3
# one output
.o 1
# input variable names
.ilb A B C
# output variable names
.ob Y
.p 8
000 1
001 0
010 1
011 1
100 0
101 0
110 -
111 -
.e
```

This can be useful as shown in the next section.

### **Example 3**

We can also minimize more than one function at a time. The following input file contains two output variables `Y` and `Z`.

```
# three inputs
.i 3
# two outputs
.o 2
.ilb A B C
.ob Y Z
.p 8
000 11
001 00
010 10
011 10
100 00
101 01
110 -1
111 -1
.e
```

Each line of the truth table contains two output values, one for each of the output functions.

## Output Format

You run the Espresso program by typing *espresso filename* at the prompt. This version of espresso writes its output to the standard output, but this can be redirected to a file, if desired.

### Example 1

When you run espresso on the input file from Example 1, the software will display:

```
# this is a comment
# three inputs
# one output
.i 3
.o 1
.p 2
0-0 1
-1- 1
.e
```

The output file format is also in PLA format. The .i, .o, .p, and .e lines have the same meaning as in the input file. However, the truth table has now been minimized. This output represents two product terms. Assuming that the input variables are named A, B, and C, the first product term, 0-0, represents the product  $A' \cdot C'$  and the second product term, -1-, represents the product term B. Thus the final minimized function is the sum of these two product terms,  $A' \cdot C' + B$ .

### Example 2

Running Espresso on the example file from Example 2 produces the following output.

```
# three inputs
# one output
.i 3
.o 1
.ilb A B C
.ob Y
.p 2
0-0 1
-1- 1
.e
```

This output is little different than the first example. However, if we use the “*eqntott*” output option by typing *espresso -oeqntott filename* at the prompt, we get the following output.

```
# this is a comment
# three inputs
# one output
Y = (!A&!C) | (B);
```

This is the eqntott format expression for the Boolean expression  $A' \cdot C' + B$ . (In this format, & is the AND operator, !A means NOT A, and | is the OR operator.)

### Example 3

Running Espresso on the input from Example 3 produces the following output.

```
# three inputs
# two outputs
.i 3
.o 2
.ilb A B C
.ob Y Z
.p 4
000 11
1-1 01
11- 01
-1- 10
.e
```

The two output functions contain a total of 4 terms, listed in the first column. A 1 in the column for the first output variable, Y, indicates that the corresponding term is included in the minimized expression for Y. Similarly, a 1 in the column for the second output variable, Z, indicates that the corresponding term is included in the minimized expression for Z. A 0 indicates that the term is not present.

The final expressions for Y and Z are.

$$Y = A' \cdot B' \cdot C' + B$$

$$Z = A' \cdot B' \cdot C' + A \cdot C + A \cdot B$$

This is easily confirmed by using the “*eqntott*” output option.

```
# three inputs
# two outputs
Y = (!A&!B&!C) | (B);
```

$$Z = (!A \& !B \& !C) \mid (A \& C) \mid (A \& B);$$

Note that we get a different function for Y in this example than we got in Example 2, even though the inputs are the same!

This is because Espresso tries to share common terms between the output functions. Since  $A' \cdot C' + B$  is logically equivalent to  $A' \cdot B' \cdot C' + B$ , the expression for Y in Example 2 produces the same output as the expression for Y in this example. However, since the term  $A' \cdot B' \cdot C'$  is also used in expression for Z, this term can be used to calculate both Y and Z, reducing the overall size of the circuits of Y and Z.

If you don't want to share common terms you can use the "Dso" option by typing *espresso -oeqntott -Dso filename* at the prompt. In this example, you would get the following output.

```
# three inputs
# two outputs
Y = (!A&!C) | (B);

Z = (!A&!B&!C) | (A&B) | (A&C);
```

This produces the same function for Y as in Example 2. This is the best expression for Y, taken by itself, but adds an additional term if both Y and Z are to be implemented in the same circuit. Use your best judgment as to which option to use to minimize your functions.