CAD Design Project 3 – Boolean Optimization with Espresso

M11152013 張棨揚

Background

The Espresso is a software package developed at IBM for two-level Boolean function minimization in 1982. The Espresso adopts positional cube notation – a binary encoding technique to process Boolean cubes, a.k.a. implicants and product terms. Combined with the unate recursive paradigm (URP), we are capable of implementing primitive Boolean operations efficiently for 2-level and multilevel logic synthesis.

Rough Comparison of Minimizers

- MINI

  Iterate EXPAND, REDUCE, RESHAPE

- Espresso

  Iterate EXPAND, IRREDUNDANT, REDUCE

- Espresso guarantees an irredundant cover

  Because of the irredundant operator

- MINI may return irredundant covers, but can guarantee only minimality w.r.t. single implicant containment

Espresso

It accepts as input a two-level representation of a two-valued (or multiple-valued) Boolean function and algorithms using heuristic Boolean minimization, The default input, and output file formats are compatible with the Berkeley standard format for the physical description of a PLA.

PLA files are expressed in binary, the ON-set of a Boolean function, we mean those minterms which imply the function value is a 1. Likewise, the OFF-set are those terms that imply the function is a 0, and the DC- set (don't care set) are those terms for which the function is unspecified.

Common Keywords

- **.i [d]**

  Specifies the number of input variables.

- **.o [d]**

  Specifies the number of output functions.

- **.p [d]**

  Specifies the number of product terms

- **.e (.end)**

  Optionally marks the end of the PLA description.

Common Keywords

- **.ilb [s1] [s2] . . . [sn]**

  Gives the names of the binary valued variables. This must come after .i and .o (or after .mv). There must be as many tokens following the key- word as there are input variables.

- **.ob [s1] [s2] . . . [sn]**

  Gives the names of the output functions. This must come after .i and .o (or after .mv). There must be as many tokens following the keyword as there are output variables.

- **.pair [d]**

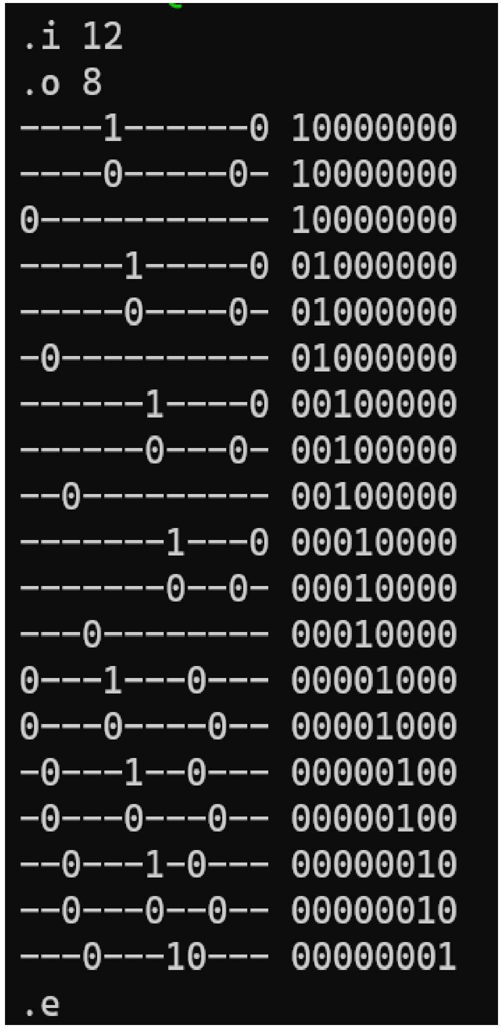
  Specifies the number of pairs of variables which will be paired together using two-bit decoders. The rest of the line contains pairs of numbers which specify the binary variables of the PLA which will be paired together.

- **.phase [s]**

  [s] is a string of as many 0's or 1's as there are output functions. It specifies which polar- ity of each output function should be used for the minimization

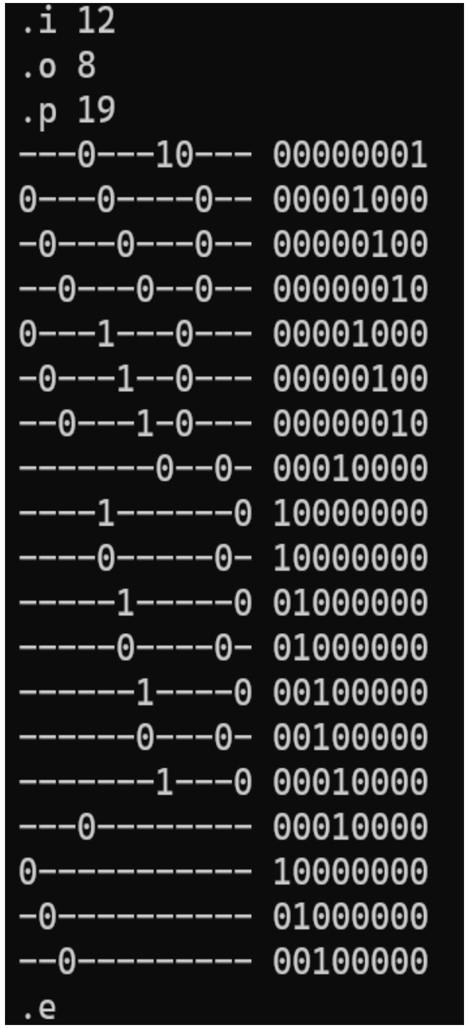
Example

Input 12 variables, output 8 variable and Boolean function , and then use espresso for Boolean Optimization



```
.i 12
.o 8
----1------0 10000000
----0-----0- 10000000
0----------- 10000000
-----1-----0 01000000
-----0----0- 01000000
-0---------- 01000000
------1----0 00100000
------0---0- 00100000
--0--------- 00100000
-------1---0 00010000
-------0--0- 00010000
---0-------- 00010000
0---1---0--- 00001000
0---0----0-- 00001000
-0---1--0--- 00000100
-0---0---0-- 00000100
--0---1-0--- 00000010
--0---0--0-- 00000010
---0---10--- 00000001
.e
```

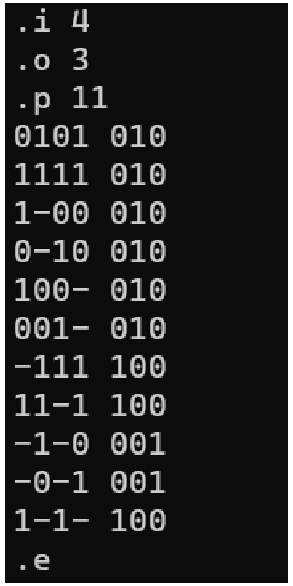
output



```
.i 12
.o 8
.p 19
---0---10--- 00000001
0---0----0-- 00001000
-0---0---0-- 00000100
--0---0--0-- 00000010
0---1---0--- 00001000
-0---1--0--- 00000100
--0---1-0--- 00000010
-------0--0- 00010000
----1------0 10000000
----0-----0- 10000000
-----1-----0 01000000
-----0----0- 01000000
------1----0 00100000
------0---0- 00100000
-------1---0 00010000
---0-------- 00010000
0----------- 10000000
-0---------- 01000000
--0--------- 00100000
.e
```

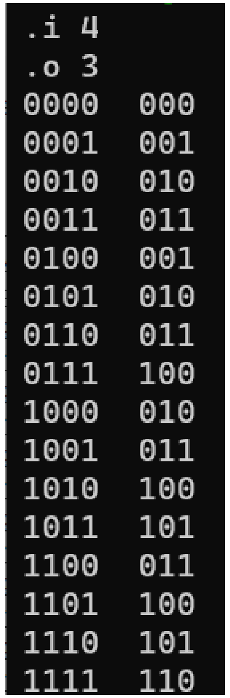
Example

A 2-bit by 2-bit binary adder can be rewritten via .pair and .phase progressions

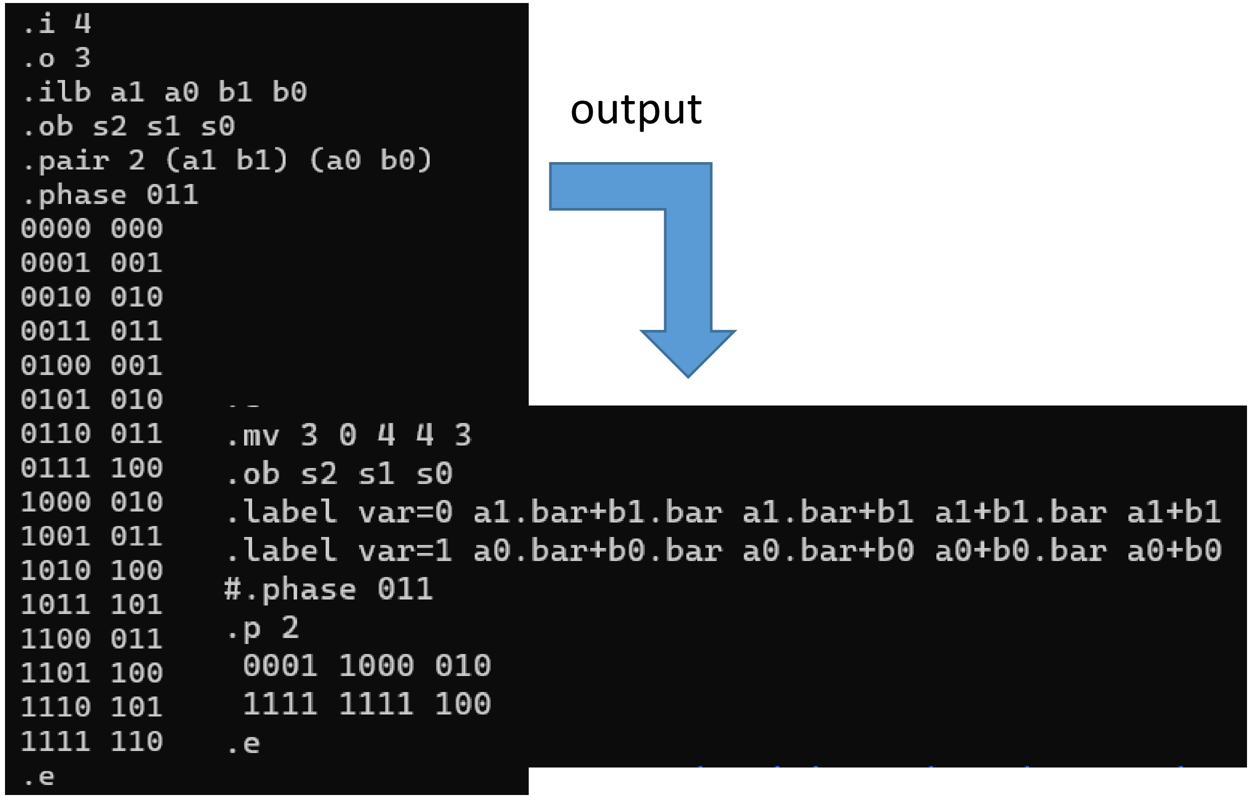


```
.i 4
.o 3
.p 11
0101 010
1111 010
1-00 010
0-10 010
100- 010
001- 010
-111 100
11-1 100
-1-0 001
-0-1 001
1-1- 100
.e
```

output



```
.i 4
.o 3
0000  000
0001  001
0010  010
0011  011
0100  001
0101  010
0110  011
0111  100
1000  010
1001  011
1010  100
1011  101
1100  011
1101  100
1110  101
1111  110
```



```
.i 4
.o 3
.ilb a1 a0 b1 b0
.ob s2 s1 s0
.pair 2 (a1 b1) (a0 b0)
.phase 011
0000 000
0001 001
0010 010
0011 011
0100 001
0101 010
0110 011
0111 100
1000 010
1001 011
1010 100
1011 101
1100 011
1101 100
1110 101
1111 110
.e
```

output

```
.mv 3 0 4 4 3
.ob s2 s1 s0
.label var=0 a1.bar+b1.bar a1.bar+b1 a1+b1.bar a1+b1
.label var=1 a0.bar+b0.bar a0.bar+b0 a0+b0.bar a0+b0
#.phase 011
.p 2
 0001 1000 010
 1111 1111 100
.e
```

Other Keywords

- **.mv [num_var] [num_binary_var] [d1] . . . [dn]**

  Specifies the number of variables (num_var), the number of binary variables (num_binary_var), and the size of each of the multiple-valued vari- ables (d1 through dn).

- **.label var=[d] [s1] [s2][…..]**

  Specifies the names of the parts of a multiple- valued variable. This must come after .mv. There must be as many tokens following the key- word as there are parts for this variable. Note that the variables are numbered starting from 0.

Other Keywords

- **.kiss**

  Sets up for a kiss-style minimization.

- **.type [s]**

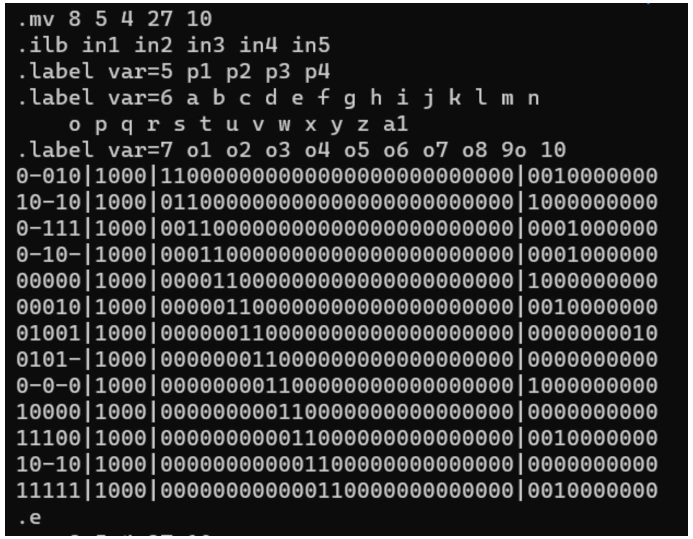
  Sets the logical interpretation of the character matrix as described below under "Logical Description of a PLA". This keyword must come before any product terms. [s] is one of f, r, fd, fr, dr, or fdr.

- **.symbolic [s0] [s1] . . . [sn] ; [t0] [t1] . . . [tm]**

- **.symbolic-output [s0] [s1] . . . [sn] ; [t0] [t1] . . . [tm] ;**

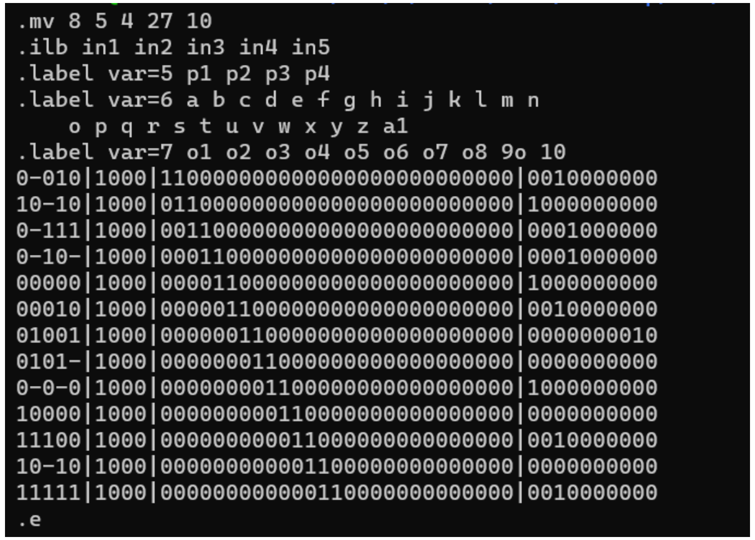
Example

- 5 binary variables and 3 multiple-valued variables (8 variables total) where the multiple-valued variables have sizes of 4 27 and 10 (note that the last multiple-valued variable is the "output" and also encodes the ON-set, DC-set and OFF-set information).



```
.mv 8 5 4 27 10
.ilb in1 in2 in3 in4 in5
.label var=5 p1 p2 p3 p4
.label var=6 a b c d e f g h i j k l m n
    o p q r s t u v w x y z a1
.label var=7 o1 o2 o3 o4 o5 o6 o7 o8 9o 10
0-010|1000|110000000000000000000000000|0010000000
10-10|1000|011000000000000000000000000|1000000000
0-111|1000|001100000000000000000000000|0001000000
0-10-|1000|000110000000000000000000000|0001000000
00000|1000|000011000000000000000000000|1000000000
00010|1000|000001100000000000000000000|0010000000
01001|1000|000000110000000000000000000|0000000010
0101-|1000|000000011000000000000000000|0000000000
0-0-0|1000|000000001100000000000000000|1000000000
10000|1000|000000000110000000000000000|0000000000
11100|1000|000000000011000000000000000|0010000000
10-10|1000|000000000001100000000000000|0000000000
11111|1000|000000000000110000000000000|0010000000
.e
```

output



```
.mv 8 5 4 27 10
.ilb in1 in2 in3 in4 in5
.label var=5 p1 p2 p3 p4
.label var=6 a b c d e f g h i j k l m n
    o p q r s t u v w x y z a1
.label var=7 o1 o2 o3 o4 o5 o6 o7 o8 9o 10
0-010|1000|110000000000000000000000000|0010000000
10-10|1000|011000000000000000000000000|1000000000
0-111|1000|001100000000000000000000000|0001000000
0-10-|1000|000110000000000000000000000|0001000000
00000|1000|000011000000000000000000000|1000000000
00010|1000|000001100000000000000000000|0010000000
01001|1000|000000110000000000000000000|0000000010
0101-|1000|000000011000000000000000000|0000000000
0-0-0|1000|000000001100000000000000000|1000000000
10000|1000|000000000110000000000000000|0000000000
11100|1000|000000000011000000000000000|0010000000
10-10|1000|000000000001100000000000000|0000000000
11111|1000|000000000000110000000000000|0010000000
.e
```

Discussion

Espresso is an application that has been developed for a long time and is still power today. In the past two weeks of testing and observation, we found out it can quickly generate the most appropriate simplification results in both complex instructions and the tedious boolean fuction, through which it can reduce unnecessary product tern and give us the solution after the boolean function is simplified.

Reference

[1] Espresso logic minimizer (Source code)

https://github.com/psksvp/espresso-ab-1.0

[2] Espresso: A Multi-valued PLA minimization (Documentation)

https://ptolemy.berkeley.edu/projects/embedded/pubs/downloads/espresso/index.htm

[3] PLA format description

https://ddd.fit.cvut.cz/www/prj/TT-Min/pla.html

[4] L07_comb_two2.pdf

https://moodle2.ntust.edu.tw/pluginfile.php/224729/mod_resource/content/1/L07_comb_two2.pdf