ESD Electronic Systems Design
Synthesis Verification Testing Power Communication

Espresso
Two-level Boolean minimization

Luigi Di Guglielmo
Davide Bresolin
Tiziano Villa

University of Verona
Dep. Computer Science
Italy

Agenda

- Introduction
- *espresso* – two-level Boolean minimization
- *espresso* - Input file
  - description format
  - keywords
- *espresso* - Options
- Exercises

Introduction

- A Boolean function can be described providing:
  - ON-set
    - OFF-set is the complement of the ON-set.
    - The DC-set is empty
  - ON-set and DC-set
    - OFF-set is the complement of the union of ON-set and DC-set
  - ON-set and OFF-set
    - DC-set is the complement of the union of ON-set and OFF-set
- A Boolean function is completely described by providing its **ON-set**, **OFF-set** and **DC-set**.

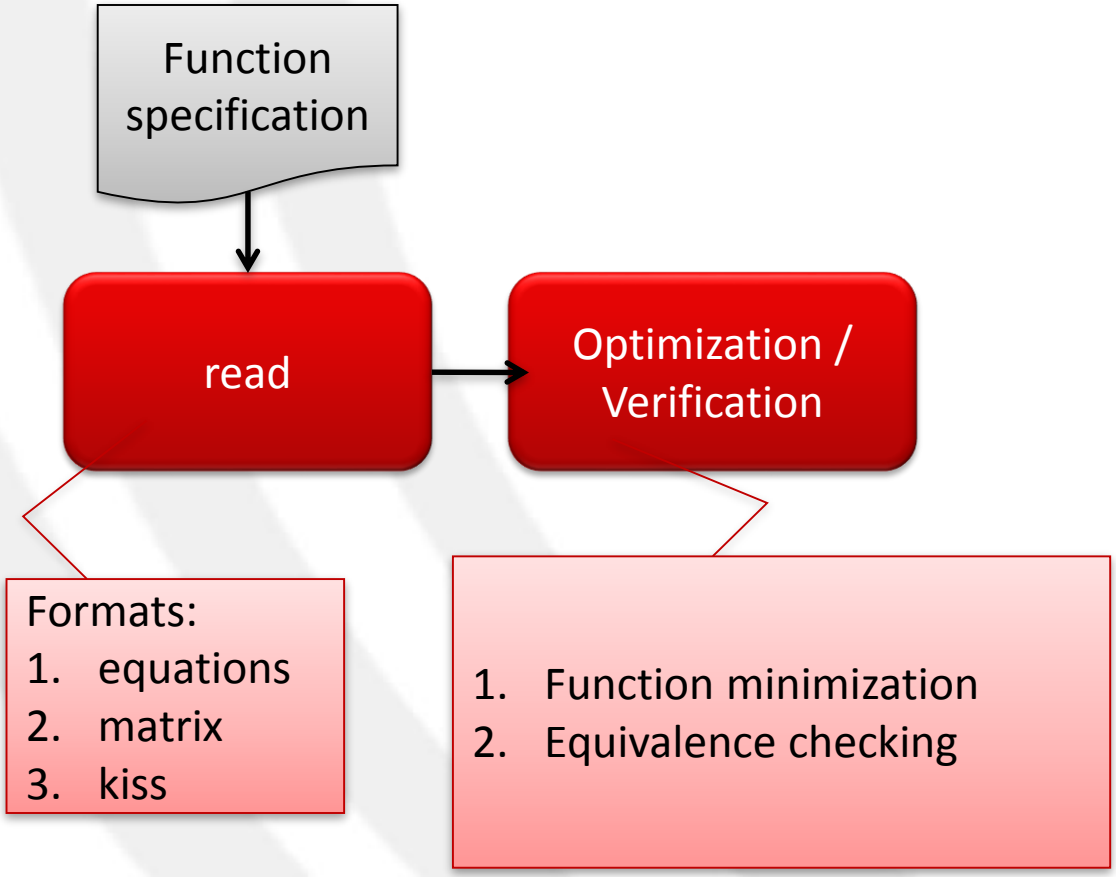
espresso – U.C. Berkeley

- *espresso* is a tool developed by the CAD group at U.C. Berkeley (software developer: Richard L. Rudell)
- Current release is the #2.3
  - Release date 01/31/1988
- *espresso* is a program for **two-level Boolean minimization**

espresso – Boolean Minimization

- *espresso* takes as input:
  - A sum-of-product (SOP) representation of a two-valued (or multi-valued) Boolean function
- and produces:
  - a minimal equivalent SOP representation

How to use *espresso*



espresso – Basic usage

$>*espresso* [options] [in_file] [> out_file]

- Reads the *in_file* provided
  - Or the standard input if no file is specified
- Writes the minimized results in *out_file*
  - Or to the standard output if no redirection to file is specified

Example - Adder

$$sum = \overline{ain} * \overline{bin} * cin + \overline{ain} * bin * \overline{cin} + ain * \overline{bin} * \overline{cin} + ain * bin * cin$$

$$cout = ain * bin * \overline{cin} + \overline{ain} * bin * cin + ain * bin * cin + ain * \overline{bin} * cin$$

ain	bin	cin	sum	cout
0	0	1	1	0
0	1	0	1	0
1	0	0	1	0
1	1	1	1	1
1	1	0	0	1
0	1	1	0	1
1	0	1	0	1

espresso – Input file format (I)

- *espresso* accepts specifications described as a character *matrix* with *keywords* embedded
  - keywords specify:
    - the size of the matrix
    - the format of the function
  - comments:
    - allowed using **#**
  - whitespaces:
    - Blanks, tabs … are ignored

espresso – Input file format (II)

- Semantics of input part
  - The format of the function
    - each position in the input matrix corresponds to an input variable where:
      - "0" implies the corresponding input literal appears complemented in the product term
      - "1" implies the input literal appears uncomplemented in the product term
      - "-" implies the input literal does not appear in the product term

espresso – Input file format (III)

- Semantics of output part
  - Specifying the format of the function
    - type *f*:
      - for each output, a **1** means this product term belongs to the *ON-set*, and **0** or **–** means this product term has *no meaning* for the value of this function
    - type *fd*:
      - for each output, a **1** means this product term belongs to the *ON-set*, **–** implies this product term belongs to the *DC-set* and a **0** means this product term has *no meaning* for the value of this function
      - it is the default type

espresso – Input file format (IV)

- type *fr*:
  - for each output, a **1** means this product term belongs to the *ON-set*, a **0** means this product term belongs to the *OFF-set*, and a **–** means this product term has *no meaning* for the value of this function
- type *fdr*:
  - for each output, a **1** means this product term belongs to the *ON-set*, a **0** means this product term belongs to the *OFF-set*, a **–** means this product term belongs to the *DC-set*, and a **~** implies this product term has *no meaning* for the value of this function

espresso – Input file keywords (I)

- The following keywords are recognized by *espresso*:
  - **.i** [d]
    - specifies the number "d" of input variables
  - **.o** [d]
    - specifies the number "d" of output variables
  - **.type** [s]
    - specifies the logical interpretation of the output part of the character matrix
    - this keyword must come before any product term
    - [s] is one of "*f*" "*fd*" "*fr*" "*fdr*"
  - **.e**
    - optionally marks the end of the description

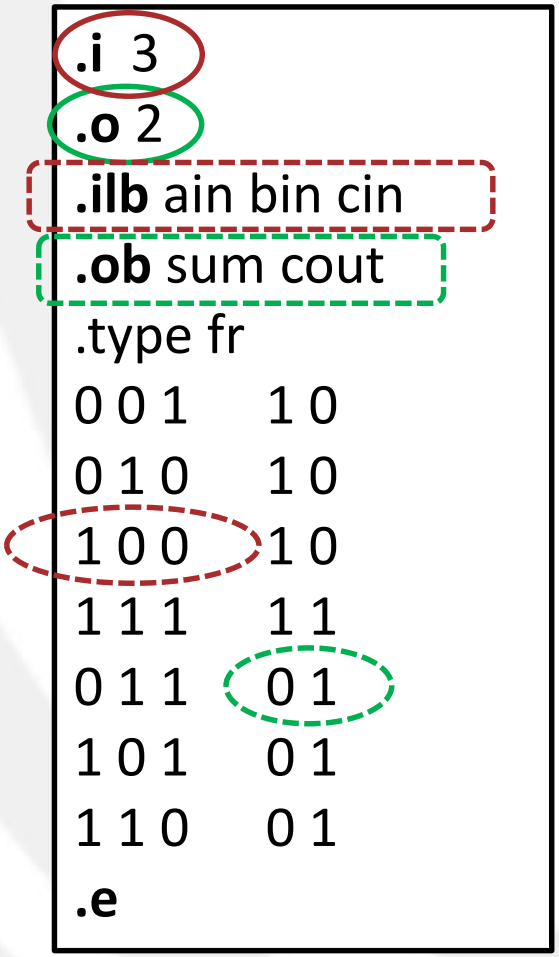
espresso – Input file keywords (II)

```
# num of input vars
# e.g., ain, bin, cin
.i 3
# num of output functions
# e.g., sum, cout
.o 2
.type fr
0 0 1    1 0
0 1 0    1 0
1 0 0    1 0
1 1 1    1 1
0 1 1    0 1
1 0 1    0 1
1 1 0    0 1
.e
```

espresso – Input file keywords (III)

- **.ilb** [s1] [s2] .. [sn]
  - gives the names of the binary-valued variables
  - must come after **.i** and **.o**
  - as many tokens as input variables
- **.ob** [s1] [s2] .. [sn]
  - gives the names of the output function
  - must come after **.i** and **.o**
  - as many tokens as output variables

espresso – Input file keywords (IV)



espresso – Input file keywords (V)

- **.phase** [b1] [b2] .. [bn]
  - specifies the phase of each output
    - positive (1) or negative (0)
  - must come after **.i** and **.o**
  - as many tokens as output variables
- **.p** [d];
  - specifies the number [d] of products
  - optional

espresso – Input file keywords (VI)

- **.symbolic** [s0]..[sN] ; [t0] .. [tM] ;
  - the binary variables named [s0] thru [sN] must be considered as a single multiple-valued variable
    - variable with 2^N parts corresponding to the decodes of the binary-valued variables
  - [s0] is the most significant bit, [sN] is the least significant bit
  - [t0] .. [tm] provide the labels for each decode of [s0] thru [sN]
- **.mv** [num_var] [num_bin_var] [d1] [dN]
  - specifies the number *num_var* of variables, the number *num_bin_var* of binary variables and the size of each of the multiple-valued variables (*d1* through *dN*)

espresso – Input file keywords (VII)

```
.i 4
.o 3
.ilb ain<1> ain<0> bin<1> bin<0>
.ob sum<1> sum<0> cout

.symbolic ain<1> ain<0>
.symbolic bin<1> bin<0>
.symbolic sum<1> sum<0>

0 0 0 0  0 0  0
0 0 0 1  0 0  1
0 0 1 0  0 1  0
0 0 1 1  0 1  1
0 1 0 0  0 0  1
 ...
```

```
 ...

0 1 0 1  0 1 0
0 1 1 0  0 1 1
0 1 1 1  1 0 0
1 0 0 0  0 1 0
1 0 0 1  0 1 1
1 0 1 0  1 0 0
1 0 1 1  1 0 1
1 1 0 0  0 1 1
1 1 0 1  1 0 0
1 1 1 0  1 0 1
1 1 1 1  1 1 0
.e
```

espresso – Options (I)

- Interesting options for running *espresso* are:
  - **-D***check*
    - checks that ON-set, OFF-set, DC-set are disjoint
  - **-D***exact*
    - performs exact minimization (potentially expensive)
  - **-D***many*
    - reads and minimizes all PLA defined into the input file
  - **-D***opo*
    - performs output phase optimization, i.e., reduce the number of terms needed to implement the function or its complement

espresso – Options (II)

- **-D*verify***
  - checks for Boolean equivalence of two functions
  - requires two filenames from command line
- **-D*equiv***
  - identifies output variables which are equivalent
- **-D*so***
  - minimizes each function one at time as a single-output function
- **-epos**
  - swaps the ON-set and OFF-set of the function after reading the function
  - useful for minimizing the OFF-set of a function

espresso – Options (II)

- **-v ‘’**
  - verbose debugging details
  - ‘’ activates all details
- **-d**
  - enables debugging
- **-o** [type]
  - selects the output format
  - type can be:
    - *f*: only On-set
    - *fd*: ON-set and DC-set
    - *fr*: ON-set and OFF-set
    - *fdr*: ON-set, OFF-set and DC-set

U.C. Berkeley – Official release

- Official *espresso* release is available at http://embedded.eecs.berkeley.edu/pubs/downloads/espresso/index.htm
  - Source code
  - Examples
  - Man pages for *espresso*

Lab configuration for Espresso

- The latest version of the tool is installed:
  - $> espresso --help
- Man pages are available
  - http://bear.cwru.edu/eecs_cad/man_octtools_espresso.html
  - http://user.engineering.uiowa.edu/~switchin/OldSwitching/espresso.5.html

Man pages

- PLA format manual (espresso.5)
  - see examples
    - #1, a two bit adder
    - #2, multi-valued function
    - #3, multi-valued function setup for *kiss*-style minimization
- *espresso* usage manual (espresso.1)
  - List options by espresso -h

Exercise 1 (I)

- The Indian society of Natchez, who lived in North America, was divided into four groups: *Suns*, *Nobles*, *Honorables*, *Stinkards*.
  In this society, marriages were allowed according to specific rules, and the corresponding progeny belongs to a particular group as described in the following table:

Mother	Father	Progeny
Sun	Stinkard	Sun
Noble	Stinkard	Noble
Honorable	Stinkard	Honorable
Stinkard	Sun	Noble
Stinkard	Noble	Honorable
Stinkard	Honorable	Stinkard
Stinkard	Stinkard	Stinkard

- Other combinations are not allowed.

Exercise 1 (II)

1. Represent the condition that characterizes the progeny of type Stinkard using a multi-valued single product.
2. Represent, using the minimum number of multi-valued products, the illegal marriages.
3. Represent using the minimum number of multi-valued products the illegal marriages and progeny group.

Exercise 2 (I)

- Formulate the minimum map coloring problem (coloring a map with the minimum number of colors such that adjacent regions don't have the same color) as a logic minimization problem.
- Apply your formulation to the following map and use *espresso* to find a minimum coloring for the map.

Exercise 2 (II)