Unit 17 Espresso minimization algorithm.

- The ESPRESSO program is an example of a HEURISTIC algorithm.
- The espresso algorithm consists of three basic steps called:
  - Expand
  - Irredundant cover.
  - Reduce.
- A program calles SIS is available on the net which implements the ESPRESSO algorithm,
- Examples of simple input files for SIS are presented.

In this unit we encounter two problems which are becoming significant with the increasing use of computers.

In the minimization of simple Boolean logic systems, it is, in principle, possible to test all possible combinations of inputs and verify that the optimum solution has been obtained. It is also possible to follow all of the steps by which the optimum solution has been obtained.

When the number of inputs and outputs of a Boolean minimization problem is large, say 50 or more inputs, it is no longer possible to determine, by enumeration, that the solution obtained is the optimum solution. The problem is said to be NP-complete. An example of this type of problem is the traveling salesman problem in which a salesman has to visit n cities and the distance traveled is to be minimized. There are $n!$ (factorial n) routes. For 50 cities, there are 3×10^{64} possible routes so it is evident that even for this small number of cities, it is no longer possible to calculate the total distance for all possible routes and be sure that the optimum route has been obtained. A good algorithm for the solution of this type of problem will give what is called a *near optimal solution.*

The second difficulty with large problems is that they can no longer be solved analytically by hand and that near optimal solutions can only be obtained by the use of computer programs operating with a set of algorithms which have been shown to give good solutions by extensive trials on other similar problems. The solution of the problem therefore depends on the application of a program which can not be proved to be correct but which must be accepted with the assumption that if it gave good solutions on similar problems it will give good solutions on the present problem.

So we are now faced with the situation where large Boolean minimization problems cannot, because of their size, be fully solved and we have to compromise on a near optimal computer solution. There have been many programs written which obtain such near optimal solutions

One of the programs which is most open to analysis is the SIS (Sequential Interactive Synthesis) package of programs from The Department of Electrical Engineering and Computer Science of the University of California, Berkeley. The algorithms and the program structure are described in Logic Minimization algorithms for VLSI Synthesis by Brayton, Hachtel, McMullen and Sangiovanni-Vincentelli and the program itself can be down loaded from the http://www.mrc.uidaho.edu/vlsi/cad_free.html net site or from the ic.eecs.berkeley.edu site and is available in both both UNIX and DOS port formats. The version which has been used for the examples in this unit is the version SIS for DOS which is the DOS port of the SIS program which was carried out by Paul Stallard, Dept of Computer Science, University of Bristol. Email: paul@cs.bris.ac.uk and Dave Protheroe, Dept of Electrical and Electronic Engineering, South Bank University, Email: prothed@vax.sbu.ac.uk

The SIS program is really a suite of algorithms which can be used for solving many problems. In this unit we are concerned with the ESPRESSO algorithm which is included as a command in the SIS program.

We will give a brief description of the ESPRESSO algorithm here as an illustration of the types of of algorithms that are used in such computerized minimization problems but is should be borne in mind that this is a public domain program and that there are many confidential, proprietary programs in use which are not subject to public scrutiny and testing but which are being used to design logic systems which are sold to the public. It is left to the reader to consider whether lack of public scrutiny of design methods is a healthy situation and whether a manufacturer's assertion that a particular product is not qualified for life critical applications is a valid defense in the event of an accident caused by a program containing a bug. It is also worth considering whether it is valid for a manufacturer to exclude the use of advanced integrated circuits from use in life critical applications on the grounds that the manufacturer cannot guarantee the design.

The ESPRESSO algorithm is an example of a HEURISTIC program, that is a program which uses approximate methods based on previous experience to obtain a near optimal solution. The word heuristic comes from the Greek root "heurein", to find, and heuristic programs and algorithms usually have the characteristic that they are fast at finding a solution and easily modified if a better method is found. In simple terms these heuristic programs can be described as a suite of sophisticated rules of thumb.

The ESPRESSO algorithm is based on the cube notation used for representing Boolean functions. Consider a Boolean function having n inputs. The function can be considered to exist in n dimensional space with one axis for each of the n inputs.

The values of the Boolean function can be considered to occupy points in space corresponding to the values of 0 or 1 for each of the inputs.

A (trivial) Boolean expression of one variable can occupy either of the points 0 or 1 in a 1 dimensional space as shown in Figure 17.1 (a). A Boolean function of two inputs could occupy any of the four points as shown in Figure 17.1 (b). By extension, a Boolean expression of n inputs could occupy any of the 2^n vertices of an n dimensional cube. A real Boolean expression will also normally occupy a number of vertices of the n dimensional cube.

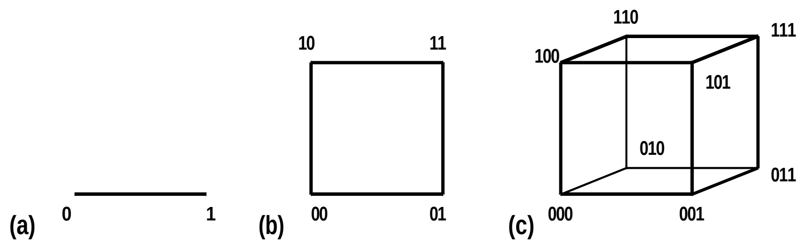
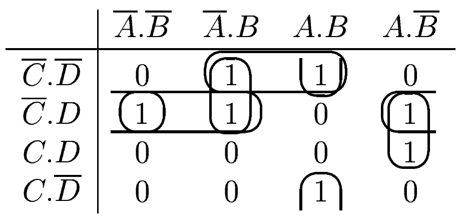


Figure 17.1: This is a repeat of figure 7.6.

It is now possible to visualize a number of lower dimensional sub cubes which will cover the occupied vertices of the n cube. some of these sub cubes will be contained within other sub cubes of greater dimension. Two sub cubes are considered to intersect if they have at least one common term or vertex. A prime cube is a cube that cannot be expanded without including a vertex for which the Boolean expression is off—a vertex corresponding to a member of the off-set of minterms.

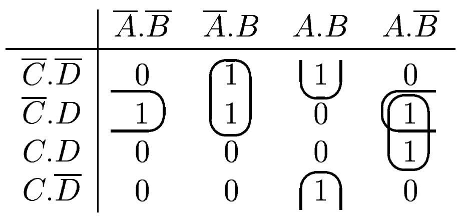
The operation of the ESPRESSO algorithm then consists of three fundamental steps.

1. Expand. A cube is expanded until no further expansion is possible without including a vertex of the off-set, that is, the cube is expanded until it is prime. This operation involves complementing each input, in turn, to test if the new vertex is a member of the off-set or on-set of the Boolean expression. At the end of this expansion process, we have a prime cover of the function such that no prime cube contains any other prime cube. However, a proper subset of these prime cubes may also provide a cover. This is illustrated on the Karnaugh map shown below. Each of the pairs is prime but it is possible to select a smaller number of pairs which will give a full coverage.



	$\overline{A}.\overline{B}$	$\overline{A}.B$	$A.B$	$A.\overline{B}$
$\overline{C}.\overline{D}$	0	1	1	0
$\overline{C}.D$	1	1	0	1
$C.D$	0	0	0	1
$C.\overline{D}$	0	0	1	0

A smaller subset of primes which would give full coverage is shown in the Karnaugh map below. It is worth noting that, even in this simple system, it is possible to select the set of primes which will give full cover in a number of different ways.



	$\overline{A}.\overline{B}$	$\overline{A}.B$	$A.B$	$A.\overline{B}$
$\overline{C}.\overline{D}$	0	1	1	0
$\overline{C}.D$	1	1	0	1
$C.D$	0	0	0	1
$C.\overline{D}$	0	0	1	0

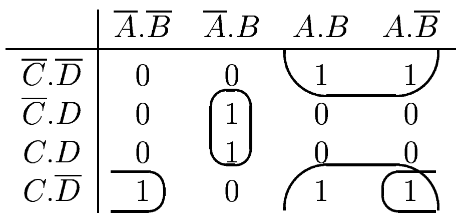
2. Irredundant_Cover. This procedure within ESPRESSO attempts to reduce the number of prime cubes to the minimum so that there are no redundant prime cubes covering the Boolean function. This is essentially the step carried out in moving between the two Karnaugh maps shown above.

The primes are classified into *relatively_essential* cubes which cannot be omitted without destroying the covering property and *redundant* cubes which can be removed without destroying the covering property. This redundant set can be partitioned into a totally redundant set and a partially redundant set and a minimal_irredundant procedure attempts to select a minimal coverage.

3. Reduce. The Expand and Irredundant_Cover procedures will give a locally optimal solution which may not be the global optimum solution. Remember that this is an NP-Complete problem where the number of solutions is so large that it is not possible to test all solutions for optimality. The Reduce procedure transforms a prime cover into a new cover by replacing each cube by a smaller cube contained within it.

These three procedures of Expand, Irredundant_Cover and Reduce are iterated with different starting points until there is no further improvement in the optimality of the reduction. The ESPRESSO command is then completed by the preparation of a file which contains the optimum output configuration for the Boolean function.

Now let us consider the actual operation of the ESPRESSO command in the SIS program and take as an example, the minimization of the logic system for which the Karnaugh map is shown below.



	$\overline{A}.\overline{B}$	$\overline{A}.B$	$A.B$	$A.\overline{B}$
$\overline{C}.\overline{D}$	0	0	1	1
$\overline{C}.D$	0	1	0	0
$C.D$	0	1	0	0
$C.\overline{D}$	1	0	1	1

This Karnaugh map can be expressed as an input file for the SIS program by putting it into the PLA format (Programmable Logic Array) as shown in the file below which has been called blexpt1. Note that the comments within the file have been added to the listing afterwards and do not form part of the input data to the program.

```
.i  4                   ;There are 4 inputs to the system.
.o  1                   ;There is one output from the system.
1100 1                  ;These are the values of ABCD for output 1.
1000 1
0101 1
0111 1
0010 1
1110 1
1010 1
```

The SIS system is then started and this input file is read in to the computer by using the read_pla instruction. The ESPRESSO command is given to minimize the logic system. The result of the minimization is read out using the write_blif command to write the output to a Berkeley Logic Interchange Format (blif) file which is called blexpt1o. The sequence of computer instructions is shown below.

```
sis
read_pla blexpt1
espresso
write_blif blexpt1o
quit
```

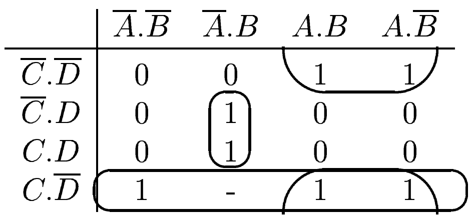
The minimized logic output is then available in the file called blexpt1o which is shown below. Again, the comments were inserted into the file afterwards.

```
.model blexpt1              ;Name of the input file
.inputs v0 v1 v2 v3         ;SIS names for A, B, C, D
.outputs v4.0               ;The first and only output
.names v0 v1 v3 [1]         ;First Product term, ABD
011 1                       ;ABD = 011  gives output 1
.names v0 v3 [2]            ;Second Product term, AD
10 1                        ;AD = 10  gives output 1
.names v1 v2 v3 [3]         ;Third Product term BCD
010 1                       ;BCD = 010 gives output 1
.names [1] [2] [3] [4]      ;Output is product of 3 complements
000 1                       ;having value 000 = 1
.names [4] v4.0             ;Complemented to give Sum of Products
0 1                         ;by use of De Morgan's Theorem
.end
```

We can now interpret this file as follows:
$\overline{A}.B.D$ is represented by lines 4 and 5
$A.\overline{D}$ is represented by lines 6 and 7
$\overline{B}.C.\overline{D}$ is represented by lines 8 and 9
which gives the Boolean exprssion in lines 10 to 13:

$$Q = \overline{\overline{(\overline{A}.B.D)}\;\overline{(A.\overline{D})}\;\overline{(\overline{B}.C.\overline{D})}} = \overline{A}.B.D + A.\overline{D} + \overline{B}.C.\overline{D}$$

It frequently occurs that a logic system contains inputs or outputs which do not affect the operation of the system. These inputs and outputs are called dont-care states and are indicated by - instead of the 0 or 1. A Karnaugh map which contains a dont-care output state is shown below and the corresponding PLA format input file is also shown as blexpt2.


$\overline{A}.\overline{B}$ $\overline{A}.B$ $A.B$ $A.\overline{B}$
$\overline{C}.\overline{D}$ 0 0 1 1
$\overline{C}.D$ 0 1 0 0
$C.D$ 0 1 0 0
$C.\overline{D}$ 1 - 1 1

The dont-care state in the bottom row of the Karnaugh map allows us to have a quad across the bottom row.

```
.i  4
.o  1
1100 1
1000 1
0101 1
0111 1
0010 1
1110 1
1010 1
0110 -                        ;This is the extra dont-care output.
```

When this input file is read into the SIS program and the ESPRESSO instruction executed, the output file blexpt2o is obtained.

```
sis
read_pla blexpt2            ;New input file
espresso
write_blif blexpt2o         ;New output file
quit
```

The commented listing of the output file, blexpt2o, is then:

```
.model blexpt1
.inputs v0 v1 v2 v3
.outputs v4.0
.names v0 v1 v3 [2]        ;Pair  ABD = 011
011 1
.names v0 v3 [3]           ;Quad   AD = 10
10 1
.names v2 v3 [4]           ;Quad   CD = 10
10 1
.names [2] [3] [4] [5]     ;Product of Complements of Products
000 1
.names [5] v4.0            ;when complemented
0 1                        ;gives Sum of Products
.exdc
.inputs v0 v1 v2 v3
.outputs v4.0
.names v0 v1 v2 v3 v4.0
0110 1
.end
```

These examples are trivial but have the advantage that it is possible to see the parallel operation of the Karnaugh map method and the ESPRESSO system. It is not possible to use the Karnaugh map method on larger problems containing many inputs and many rows in the truth table statement of the problem and also on problems which have a large number of simultaneous outputs from the logic system. A moderately large system might have 100 inputs and 30 outputs and would be easily handled by the SIS program. The example below is smaller than this (because of space constraints on this page) but it does illustrate the real power of the ESPRESSO instruction on the SIS computer package. This example file is included with the SIS package and is called alu1.

The input file has 12 inputs and 8 outputs and the truth table has 19 rows which contain many dont-care inputs.

```
.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
```

When this file is processed in SIS the output file below is obtained in less than 1 second of processing time on a Pentium PC.

```
.model alu1
.inputs v0 v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11
.outputs v12.0 v12.1 v12.2 v12.3 v12.4 v12.5 v12.6 v12.7
.names v3 v7 v8 [8]
010 1
.names v0 v4 v9 [9]
000 1
.names v1 v5 v9 [10]
000 1
.names v2 v6 v9 [11]
000 1
.names v0 v4 v8 [12]
010 1
.names v1 v5 v8 [13]
010 1
.names v2 v6 v8 [14]
010 1
.names v7 v10 [15]
00 1
.names v4 v11 [16]
10 1
.names v4 v10 [17]
00 1
```

```
.names v5 v11 [18]
10 1
.names v5 v10 [19]
00 1
.names v6 v11 [20]
10 1
.names v6 v10 [21]
00 1
.names v7 v11 [22]
10 1
.names v3 [23]
0 1
.names v0 [24]
0 1
.names v1 [25]
0 1
.names v2 [26]
0 1
.names [16] [17] [24] [27]
000 1
.names [27] v12.0
0 1
.names [18] [19] [25] [29]
000 1
.names [29] v12.1
0 1
.names [20] [21] [26] [31]
000 1
.names [31] v12.2
0 1
.names [15] [22] [23] [33]
000 1
.names [33] v12.3
0 1
.names [9] [12] [35]
00 1
.names [35] v12.4
0 1
.names [10] [13] [37]
00 1
.names [37] v12.5
```

```
0 1
.names [11] [14] [39]
00 1
.names [39] v12.6
0 1
.names [8] [41]
0 1
.names [41] v12.7
0 1
.end
```

In this listing, the 8 outputs are named as v12.0 to v12.7. It is left as an exercise for the reader to examine the listing and to identify the function or meaning of each of the lines of output. It is also left as an exercise for the reader to interpret each of the lines in terms of logic AND gates and OR gates and to draw up a circuit diagram for the system. In this and larger systems, the output BLIF file would be normally be used as the input file to a program which would prepare the masks for the manufacture of the logic array integrated circuit.

The version of the SIS program which was used in these examples is the DOS port of the UNIX version of the SIS program and is to be found at the net site http://www.mrc.uidaho.edu/vlsi/cad_free.html under the selection SIS for DOS or at the net site

If you have any difficulty due to changes at these sites you could carry out a net search for the terms UNIX and SIS and locate the new site.

17.1 References

Brayton, R.K., Hachtel, G.D., McMullen, C.T., Sangiovanni-Vincentelli, A.L. (1984)
Logic Minimization Algorithms for VLSI Synthesis
Kluwer Academic Publishers

Rudell, R.L, Sangiovanni-Vincentelli, A. (1987)
Multiple-Valued Minimization for PLA Optimization
IEEE Transactions on Computer Aided Design, CAD-8, (5) 727-750, 1987

http://www.mrc.uidaho.edu/vlsi/cad_free.html

http://www.ic.berkeley.edu

17.2 Problems

17.1 Write a PLA format input data file for the logic system detailed in Problem 14.1. Write the BLIF format output file which you would expect to obtain from running the ESPRESSO command in SIS on this input file. Then run the SIS program (if it is available) and compare your expected results with the actual results. Account for any differences.

17.2 Write the PLA input file which corresponds to the minterm expression:

$$Q = \Sigma m(1, 4, 5, 8, 11, 13, 17)$$

17.3 Write the PLA format input file which corresponds to the minterm expression:

$$Q = \Sigma m(0, 3, 7, 11, 13) + d(2, 10, 15)$$

where the $d(...)$ denote the dont-care input states.

17.4 Distinguish between the effect of dont-care states in the input and dont-care states in the output of a PLA input file for SIS.

17.5 A seven segment display contains seven light emitting segments labelled a to g as shown in Figure 17.2. Design the truth table for the logic system, having 7 outputs, for a BCD to seven segment decoder which has as its inputs the four outputs of a BCD counter and which will display the numbers on the seven segment diplay. Prepare the PLA input file suitable for submission to the SIS program.

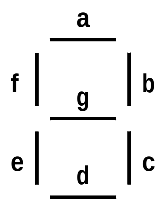


Figure 17.2: