Combinatorial Nets: analysis and synthesis

Prof. Daniele Gorla

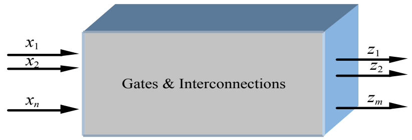
Combinatorial Net

A *combinatorial net* is a digital electronic circuit that can automatically compute a BF.

An *electronic circuit* is a system made up by elementary gates acyclically interconnected.

Lines entering into gates that do not exit from other gates are the *inputs*.

Exits of gates that do not enter in other gates are the *outputs*.



$$z_i = f_i(x_1 \ldots x_n) \quad (i=1\ldots m)$$

REMARK: output values in every moment univoquely depend from the input values.

Circuital Schema

A circuital schema is the interconnection between gates (represented in a graphical way) through lines.

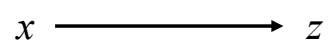
We have 3 kinds of lines:
1. Input lines, each labeled by one of the *n* boolean variables;
2. Output lines, each labeled by one of the *m* output variables (one for every computed BF);
3. Inner lines, each connecting the output of a gate with the input of other gates.

Constraints:
- Every input of every gate must be either an input line or an inner line;
- Every output of every gate must be either an output line or an inner line;
- Gates interconnections must NOT generate cycles.

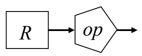
Inductive Definition of Combinatorial Net

A **combinatorial net with one output** (CN1) is inductively defined as:

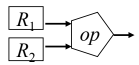
- a line connecting input *x* to output *z*



  is a CN1;
- if *R* is a CN1 and *op* is a unary gate (NOT), then also



  is a CN1;
- if R_1 and R_2 are CN1 and *op* is a binary gate (AND, OR, NOR, NAND, XOR, XNOR), then also



  is a CN1, once inputs with the same label are joint together.

A **combinatorial net with *m* outputs** (simply, **combinatorial net**, CN) is the justapposition of *m* CN1 obtained by joining inputs with the same label.

Relation between Bes and CNs

For every BE there exists a unique CN (actually, CN1) made up by NOT, AND & OR gates that computes it.

Procedure:

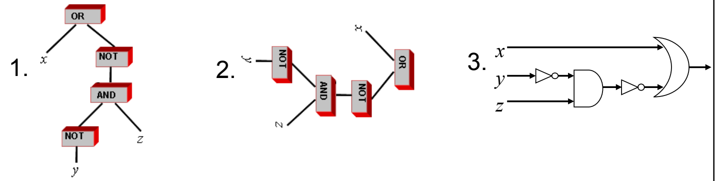
1. Given a BE, write it in a hierarchical way, by putting at the top level the last operators (out-most); variables will be at the bottom level
   → pay attention to the precedence rules for operators
   (¯ > · > +)

2. Clockwise turn this representation by 90°; this gives you the structure of the CN, with inputs at left and outputs at right

3. Turn every operator in the corresponding gate and link every gate of every level to a gate of the next level by a line

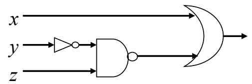
Example

$$x + \overline{\overline{y} \cdot z}$$



REMARK: by using NAND, NOR, XOR and XNOR gates, we loose unicity!

Ex.:



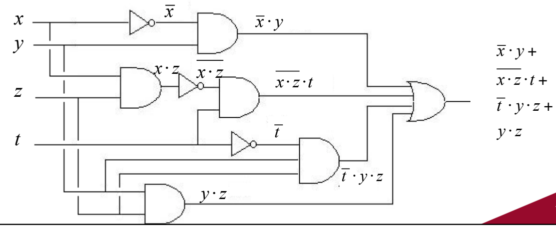
Relation between CNs and BEs

For every CN with *m* outputs there exists a *m*-tupla of BEs that describes it; such an *m*-tupla is unique up-to variables renaming (associated to the inputs of the CN)

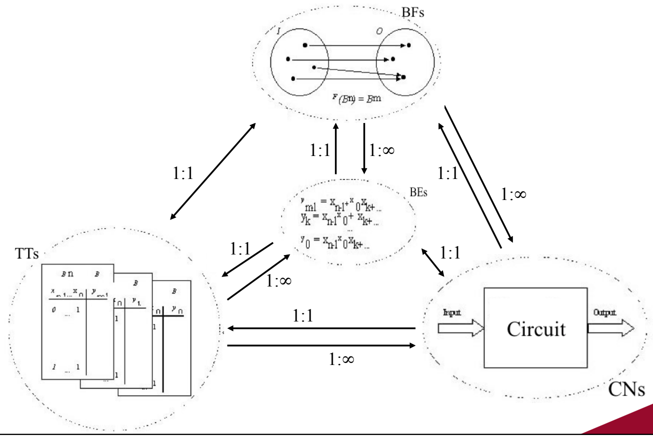
Procedure:

1. Label every input with a different variable
2. For every gate (with inputs already labeled by a BE), label the output with the BE resulting from the operation of the gate over the inputs
3. Iterate until all outputs are labeled with a BE

Ex.:



Summing up: relations between BFs, TTs, BEs and CNs



Analysis of CNs

AIM: given a CN, find the computed BF

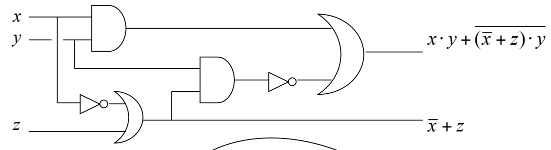
STEPS: CN $\rightarrow^1\rightarrow$ BE $\rightarrow^2\rightarrow$ TT (= BF)

Step 1 has been just considered in this class

Step 2 can be done by using one of the many methods for passing from a BE to a TT:

- Perfect induction: consider all the possible truth values to the variables and incrementally build the TT
- By using DNFs, DCFs, CNFs or CCFs

Example



DNF

$$x \cdot y + \overline{(\overline{x} + z) \cdot y} = xy + x\overline{z} + \overline{y}$$

$$\overline{x} + z = \overline{x} + z + y\overline{y} = (\overline{x} + y + z)(\overline{x} + \overline{y} + z) = M_4 \cdot M_6$$

CCF

x	y	z	f_1	f_2
0	0	0	1	1
0	0	1	1	1
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

Synthesis of CNs

AIM: given a verbal description of a BF, find a CN that computes it

STEPS: Verbal description $\rightarrow^1\rightarrow$ BF (= TT) $\rightarrow^2\rightarrow$ BE $\rightarrow^3\rightarrow$ CN

Step 1 is the only one for which we cannot give a “mechanical” procedure (it depends on the verbal description: we shall see many examples!)

Step 2 is done by using one of the ways for passing from a TT to a BE. Since we’d like to have the smallest possible CN, we shall usally rely on the KMs for finding a minimal DNF/CNF and then we shall simply as much as possible, by using Boole axioms and/or NAND/NOR/XOR/XNOR gates

Step 3 is the way we saw in this class for having a CN starting from a BE

Example (1)

Design the control circuit of a tap that controls the amount of water in a chemical solution. The tap must be closed if:

1. The tank is full; or
2. The drain valve is closed, the reagents concentration is enough and the water level is not low.

SOLUTION:

We associate a (input) variable to every event:

Tk = 1 if the tank is full, 0 otherwise;
V = 1 if the drain valve is open, 0 if closed;
C = 1 if the reagents concentration is enough, 0 otherwise;
L = 1 if the water level is low, 0 otherwise.

Also the function to be computed is an event, hence it is a (output) variable:

T = 1 if the tap is open, 0 otherwise.

Example (2)

SAPIENZA Università di Roma Dipartimento di Informatica

From the verbal description to the TT:

Tk V C L	T
0 0 0 0	1
0 0 0 1	1
0 0 1 0	0
0 0 1 1	1
0 1 0 0	1
0 1 0 1	1
0 1 1 0	1
0 1 1 1	1
1 0 0 0	0
1 0 0 1	0
1 0 1 0	0
1 0 1 1	0
1 1 0 0	0
1 1 0 1	0
1 1 1 0	0
1 1 1 1	0

$T = 0$ if $V = 0$, $C = 1$ and $L = 0$

$T = 0$ if $Tk = 1$

The tap must be closed if:
1. the tank is full; or
2. the drain valve is closed, the reagents concentration is enough and the water level is not low.

$Tk = 1$ iff the tank is full;
$V = 1$ iff the valve is open;
$C = 1$ iff the concentration is enough;
$L = 1$ iff the water level is low;
$T = 1$ iff the tap is open.

Example (3)

SAPIENZA Università di Roma Dipartimento di Informatica

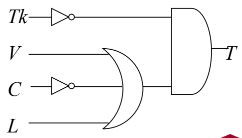
Karnaugh Map:

C L \ Tk V	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	1	1	0	0
10	0	1	0	0

Minimal DNF: $T = \overline{Tk}V + \overline{Tk}\bar{C} + \overline{Tk}L$

Minimal BE: $T = \overline{Tk}(V + \bar{C} + L)$

Circuit:



OBS.: this is a minimal POS that we'd Have obtained from the KM by covering the 0s

Example (4)

SAPIENZA Università di Roma Dipartimento di Informatica

OBS.: in this example, we could have derived the minimal BE directly from the verbal description, without passing through the TT and the KM:

Once defined the boolean variables, we have that

"The tap must be closed if: (1) the tank is full; or (2) the drain valve is closed, the reagents concentration is enough and the water level is not low"

can be directly translated to $\bar{T} = Tk + \bar{V}C\bar{L}$

from which $T = \overline{Tk + \bar{V}C\bar{L}} = \overline{Tk}(V + \bar{C} + L)$

Example (5)

SAPIENZA Università di Roma Dipartimento di Informatica

REMARK: with a different choice of the variables associated to the events, we'd have had a different TT (and hence also the BE)!!

Ex.: $Tk = 0$ if the tank is full, 1 otherwise;
$L = 0$ if the water level is low, 1 otherwise.

Would lead to $\bar{T} = \overline{Tk} + \bar{V}CL$

and hence $T = Tk(V + \bar{C} + \bar{L})$

A second example (1)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Design a combinatorial circuit that receives in input bit quadruples with an even number of "1s" and returns 1 if the number of "1s" equals the number of "0s".

SOLUTION:

In this case, we have 4 input variables, one for every bit of the quadruple;

moreover, configurations with an odd number of "1s" will never appear; hence, we shall use *don't care* symbols for such quadruples.

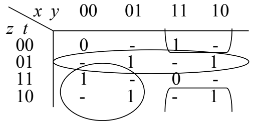
x y z t	*f*
0 0 0 0	0
0 0 0 1	-
0 0 1 0	-
0 0 1 1	1
0 1 0 0	-
0 1 0 1	1
0 1 1 0	1
0 1 1 1	-
1 0 0 0	-
1 0 0 1	1
1 0 1 0	1
1 0 1 1	-
1 1 0 0	1
1 1 0 1	-
1 1 1 0	-
1 1 1 1	0

17

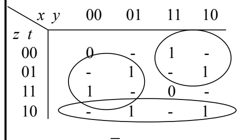
A second example (2)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

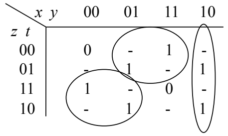
KM:



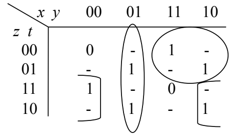
Minimal DNF: $\overline{z}t + \overline{x}z + x\overline{t}$
(8 gates)



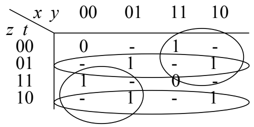
$z\overline{t} + \overline{x}t + x\overline{z}$



$\overline{x}z + y\overline{z} + x\overline{y}$



$\overline{y}z + \overline{x}y + x\overline{z}$



Minimal BE: $\overline{z}t + z\overline{t} + \overline{x}z + x\overline{z} =$

$= (z \oplus t) + (x \oplus z)$

(3 gates!!!)

↓

Not always the minimal DNF leads to the minimal BE

18