Block Diagrams

Introduction

Solving problems is often difficult in each branch of human knowledge and sometimes it is impossible without an appropriate strategy that helps us in this heavy work.

Scientists and electronic designers have tried to create many models able to help electronic technicians in this difficult task, without gaining any apparent advantage.

Actually, the best strategy to face a scientific or technical problem seems to be the old *divide et impera* strategy. It consists of splitting the whole problem into several interrelated sub-problems which may be easier to solve, if taken one by one.

When we apply this technique in Electronics, we usually draw a *block diagram.*

Block diagrams

An example of block diagram is shown in fig.1.

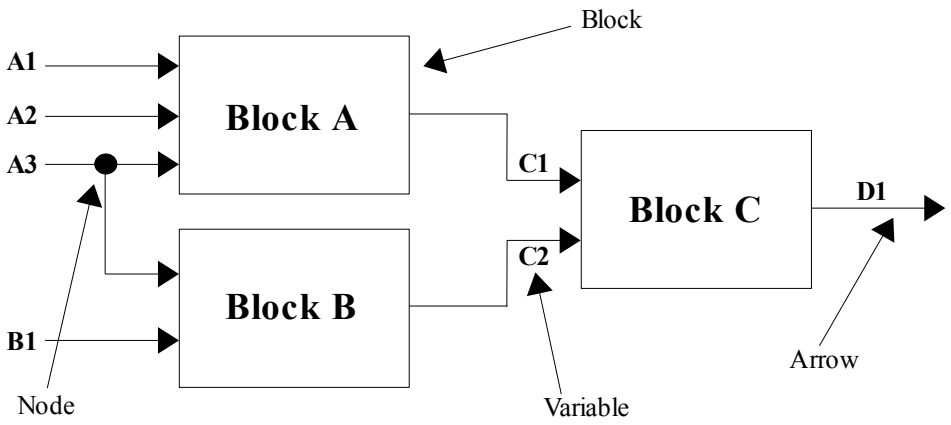


fig. 1

There we can see three blocks (block A, B and C) connected by arrows; some capital letters followed by numbers (A1, B2, etc.) representing electrical variables; nodes that let the signal flow into different blocks.

The blocks represent the solution of the sub-problem and could be an electrical analogue circuit or an electrical digital one. Each block accepts *n* input variables (represented by the letters followed by the number, e.g. A1, B2, C1, etc.) and puts *m* variables into its outputs. The arrows indicate the flow of the signal. That one indicated by the variable C2, for example, flows from block B to block C. This means that C2 is an output for block B and an input for block C.

The node lets the signal flow (like in fig. 1) both into block A and block B. It must be clear that the node doesn't interfere with the signal in any way. That means that the signal remains the same before and after the node, either at the block A input or at the block B input.

Blocks and variables

Each block is completely and not ambiguously identified when we know:

- all the input variables expressed in terms of name, function and active state (if it is a digital variable) or range (if it is an analogue variable);
- all the output variables expressed in terms of name, function and active state (if it is a digital variable) or range (if it is an analogue variable);
- the law that links each output to the related inputs. If the block is formed by a combinatorial circuit, that law is expressed by the truth table: if the block is formed by a sequential circuit, the law is expressed by the state diagram and if the block is formed by an analogue circuit, usually it can be substituted by one or more fundamental analogue circuits. This last case will be discussed in a further lesson.

The most difficult thing to do is to identify which type of block we must use: combinatorial, sequential or analogue. Typically, the greatest difficulties are met when we have to choose between a combinatorial or sequential circuit.

Before we can correctly choose one of the two solutions, we must try to write the truth table. If we obtain, for each combination of inputs and for each output, only one possible output state, the circuit is a combinatorial one, otherwise it must be a sequential one.

Another little problem which the student usually struggles on is represented by the "splitting depth": when can we stop splitting each block into other sub-blocks? Usually this action continues until each block is solvable with one of the methods that we know (Karnaugh maps, Moore or Mealy models, etc.).

In the next section we will try to face a typical problem of combinatorial logic.

An example

There are two crossing streets controlled by four traffic lights, as shown in fig. 2.

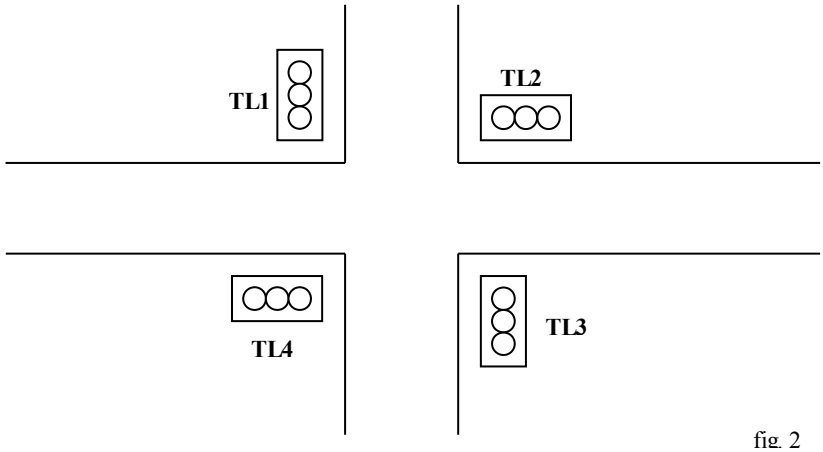


fig. 2

We must design a system that activates a line called Err (active high) whenever there is a fault in one or more of the traffic lights.

First of all, we must specify what we mean by the word “fault”.We have three types of faults: one type related to the single traffic light (single fault); one type related to traffic lights positioned on the same direction (for example TL1 and TL3: same direction fault) and one type related to traffic lights positioned on different directions (for example TL1 and TL2: different direction fault).

Single fault

We have a *single fault* in a traffic light when no lights are on or when two or more lights are on at the same time. Fault type 1.

Same direction fault

We have a *same direction fault* in two traffic lights positioned on the same direction when the two traffic lights have different lights on (for example, red in one traffic light and green in the other). Fault type 2.

Different direction fault

We have a different direction fault in two traffic lights positioned on different directions when they are not coherent each other (for example, TL1 green and TL2 green). Fault type 3.

Now that we have cleared what we mean by Single fault, Same direction fault and Different direction fault, we can try to face the problem. Its block diagram is the following:

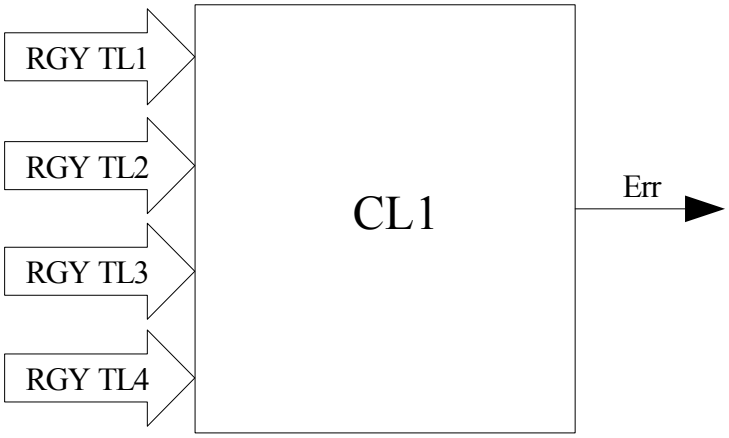


fig. 3

Each arrow in the input is formed by three different signals: Red, Green and Yellow (active high) and represents the activation signals of the traffic light. Since the number of input lines is very high (4x3=12), it is not possible to apply the Karnaugh maps to solve the problem. First we must split the single block CL1 into several, but easier to solve, blocks.

For instance, we could create, for each traffic light, a block which analyses the Single fault, as shown in figure 4. The block has to be identified according to the rules previously defined:

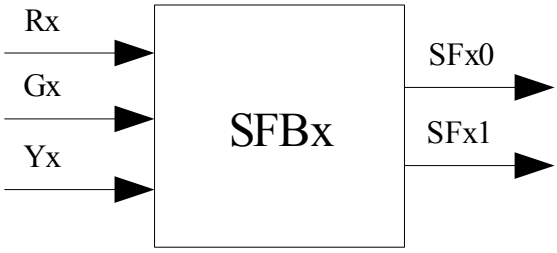


fig. 4

Inputs

Rx: Indicates that the Red light of the TL x is on. Active high;
Gx: Indicates that the Green light of the TL x is on. Active high;
Yx: Indicates that the Yellow light of the TL x is on. Active high.

Outputs

SFx0 and SFx1 (with SFx0 LSB) form together a code explained by table 1:

SFx1	SFx0	Meaning
0	0	An error on the TL x occurred
0	1	The Red light is on
1	0	The Green light is on
1	1	The Yellow light is on

tab. 1

Now we can write the truth table of the block (table 2):

Rx	Gx	Yx	SFx1	SFx0	Meaning
0	0	0	0	0	An error on the TL x occurred
0	0	1	1	1	The Yellow light is on
0	1	0	1	0	The Green light is on
0	1	1	0	0	An error on the TL x occurred
1	0	0	0	1	The Red light is on
1	0	1	0	0	An error on the TL x occurred
1	1	0	0	0	An error on the TL x occurred
1	1	1	0	0	An error on the TL x occurred

tab. 2

It is very important to focus the attention to SFx1 and SFx0. Two bits are absolutely sufficient to define without ambiguity the state of the traffic light: using two bits it is possible to know when the red light is on, when the green light is on, when the yellow light is on or when an error occurred. Actually, it is not important that the output code indicates *which kind* of error occurred, but only *if* it occurred.

The Karnaugh's map are the following:

SFx1

Rx \ Gx/Yx	00	01	11	10
0		**1**		**1**
1				

$$SFx1 = \overline{Rx}\cdot\overline{Gx}\cdot Yx + \overline{Rx}\cdot Gx\cdot\overline{Yx} \qquad (1)$$

SFx0

Rx \ Gx/Yx	00	01	11	10
0		**1**		
1	**1**			

$$SFx0 = Rx\cdot\overline{Gx}\cdot\overline{Yx} + \overline{Rx}\cdot\overline{Gx}\cdot Yx \qquad (2)$$

No reduction is possible.
Now we can draw a new block diagram, as the following:

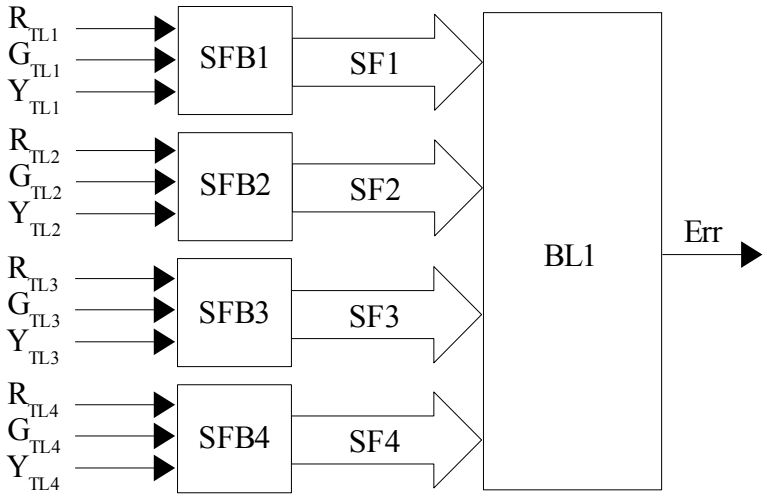


fig. 5

Actually, the block diagram shown in fig. 5 is not a good one. Each SFx output is formed by 2 wires. That means that BL1 has 8 inputs and that Karnaugh is not usable. So we must look for a different solution, even though the block SFBx represents a good solution.

We can try to continue in the same way as we started: as for SFBx, we can create a block (SDFBx) that evaluates a unique fault, for instance, the same direction fault. Proceeding in this way means applying the *divide et impera* method, introduced at the beginning of the paper.

The same direction fault block could appear as the following:

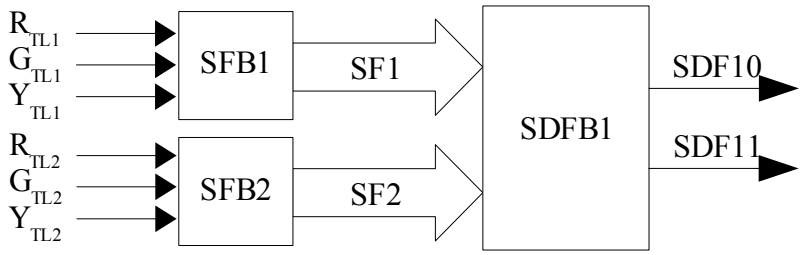


fig. 6

The block SDFB1 evaluates the same direction fault on only one direction. We need a second block, similar or equal to the first one, which evaluates the same direction fault on the orthogonal direction. As we did for the block SFBx, we put in the output not only a single error line, that goes high if a fault is detected, but a couple of lines that form together a code which identifies the state of the traffic light. This is absolutely sufficient, because, if it all works well, the traffic light that lay on the same direction must light on the same lights and not different ones.

This means that the block SDFB1 must evaluate the output signals of the blocks SFB1 and SFB2 (which must be the same) and if it is so, replicate the SF1 or SF2 code in its output. This can be easily done using the Karnaugh maps, because the input number is 2+2=4 and the maps are usable.

Now the block can be identified following the previously defined rules:

Inputs
SFx0: Low bit of the output code of block SFBx;
SFx1: High bit of the output code of block SFBx;
SFy0: Low bit of the output code of block SFBy;
SFy1: High bit of the output code of block SFBy;

Outputs
SDFx0 and SDFx1 (with SDFx0 LSB) form together a code as explained by table 3:

SDFx1	SDFx0	Meaning
0	0	An error on the TL x occurred
0	1	The Red light is on
1	0	The Green light is on
1	1	The Yellow light is on

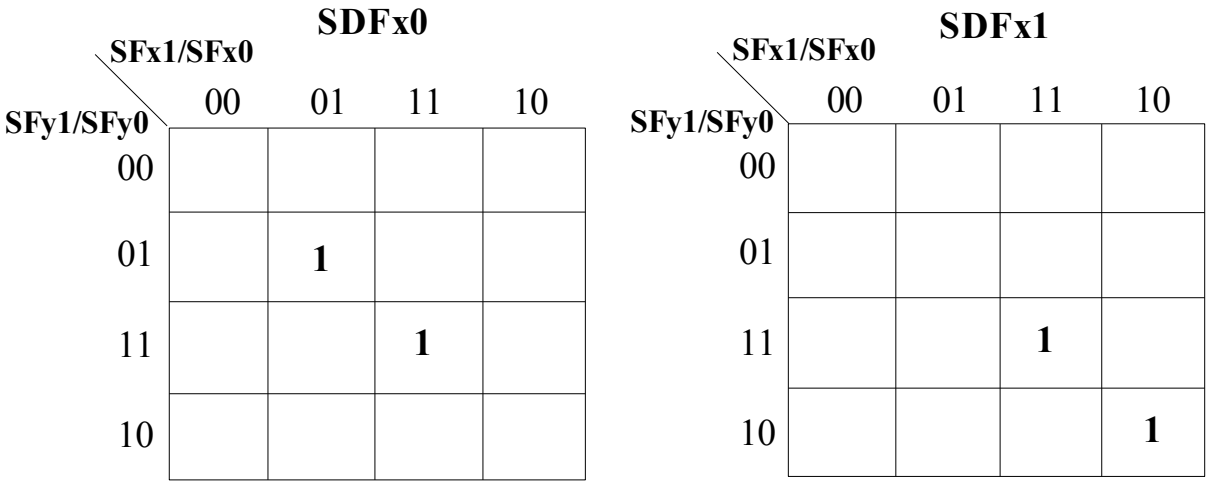
tab. 3

Now we can write the truth table of the block (table 4):

SFy1	SFy0	SFx1	SFx0	SDFx1	SDFx0	Meaning
0	0	0	0	0	0	A fault of type 1 or 2 occurred
0	0	0	1	0	0	A fault of type 1 or 2 occurred
0	0	1	0	0	0	A fault of type 1 or 2 occurred
0	0	1	1	0	0	A fault of type 1 or 2 occurred
0	1	0	0	0	0	A fault of type 1 or 2 occurred
0	1	0	1	0	1	The Red light is on
0	1	1	0	0	0	A fault of type 1 or 2 occurred
0	1	1	1	0	0	A fault of type 1 or 2 occurred
1	0	0	0	0	0	A fault of type 1 or 2 occurred
1	0	0	1	0	0	A fault of type 1 or 2 occurred
1	0	1	0	1	0	The Green light is on
1	0	1	1	0	0	A fault of type 1 or 2 occurred
1	1	0	0	0	0	A fault of type 1 or 2 occurred
1	1	0	1	0	0	A fault of type 1 or 2 occurred
1	1	1	0	0	0	A fault of type 1 or 2 occurred
1	1	1	1	1	1	The Yellow light is on

tab. 4

Now it is possible to fill in the Karnaugh's maps:



The equations are the following:

$$SDFx0 = \overline{SFy1}\cdot SFy0\cdot\overline{SFx1}\cdot SFx0 + SFy1\cdot SFy0\cdot SFx1\cdot SFx0 \quad (3)$$

$$SDFx1 = SFy1\cdot\overline{SFy0}\cdot SFx1\cdot\overline{SFx0} + SFy1\cdot SFy0\cdot SFx1\cdot SFx0 \quad (4)$$

Now we can face the last problem: reading the two codes put in the output of the blocks SDFB1 and SDFB2 and producing an error signal depending on them and the different direction fault rule.

The whole block diagram could be the following:

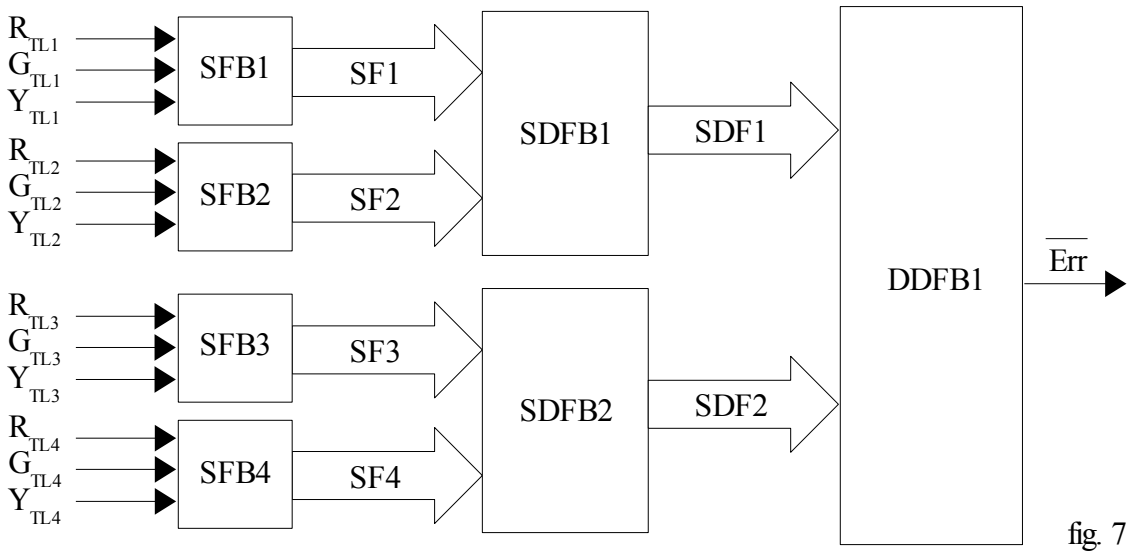


fig. 7

The last block (DDFB1) can now be identified following the previously defined (and well known) rules:

Inputs

SDF10: Low bit of the output code of block SDFB1;
SDF11: High bit of the output code of block SDFB1;
SDF20: Low bit of the output code of block SDFB2;
SDF21: High bit of the output code of block SDFB2;

Outputs

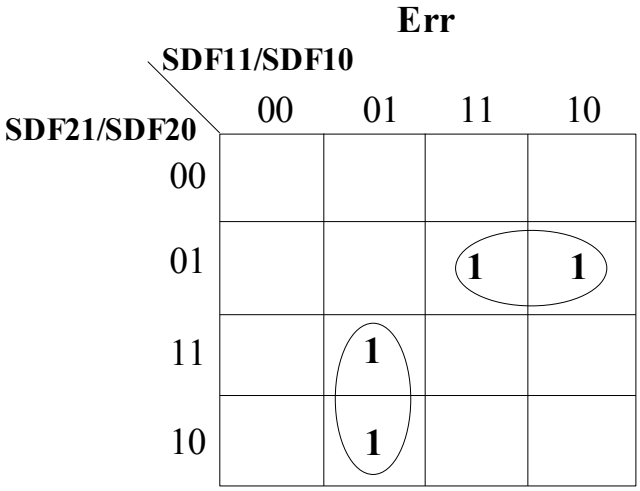
Err: Output error line. Active low.

The truth table of the block is shown in tab. 5

tab. 5

SDF21	SDF20	SDF11	SDF10	Err	Meaning
0	0	0	0	0	A fault of type 1, 2 or 3 occurred
0	0	0	1	0	A fault of type 1, 2 or 3 occurred
0	0	1	0	0	A fault of type 1, 2 or 3 occurred
0	0	1	1	0	A fault of type 1, 2 or 3 occurred
0	1	0	0	0	A fault of type 1, 2 or 3 occurred
0	1	0	1	0	A fault of type 1, 2 or 3 occurred
0	1	1	0	1	Valid situation (red + green)
0	1	1	1	1	Valid situation (red + yellow)
1	0	0	0	0	A fault of type 1, 2 or 3 occurred
1	0	0	1	1	Valid situation (green + red)
1	0	1	0	0	A fault of type 1, 2 or 3 occurred
1	0	1	1	0	A fault of type 1, 2 or 3 occurred
1	1	0	0	0	A fault of type 1, 2 or 3 occurred
1	1	0	1	1	Valid situation (yellow + red)
1	1	1	0	0	A fault of type 1, 2 or 3 occurred
1	1	1	1	0	A fault of type 1, 2 or 3 occurred

Now it is possible to fill in the Karnaugh's map:



$$Err = \overline{SDF21}\cdot SDF20\cdot SDF11 + SDF21\cdot\overline{SDF11}\cdot SDF10$$

Now that we have found the functions of the blocks SFBx, SDFBx and DDFB1, we can try to draw the electrical circuit using the gates instead of the logical functions.

The three solutions related to the blocks are shown in fig. 8, 9 and 10, while the whole circuit is shown in fig. 11.

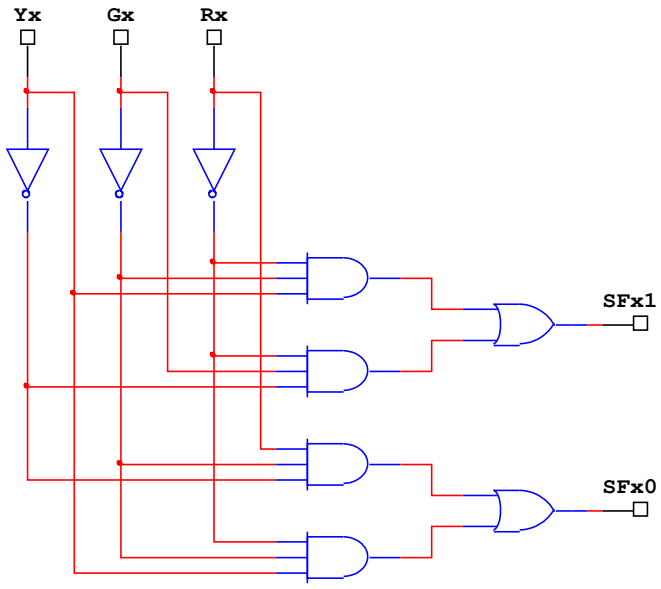


fig. 8

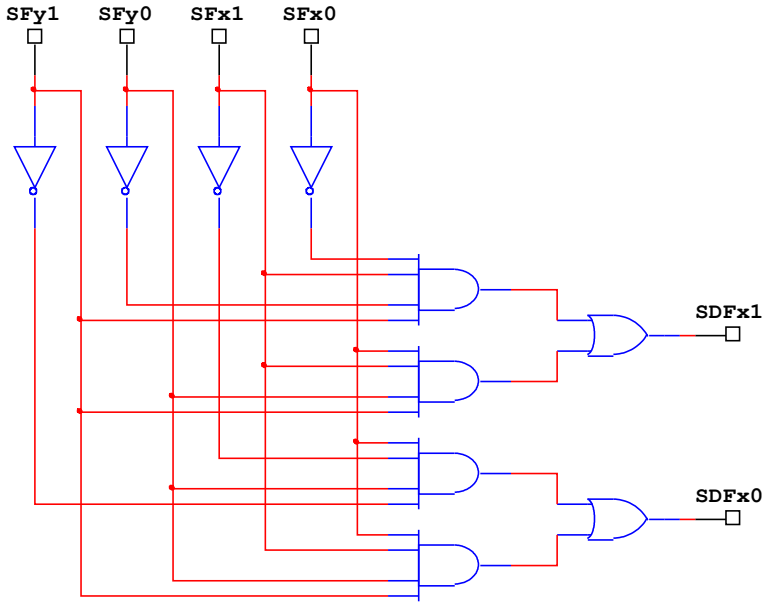


fig. 9

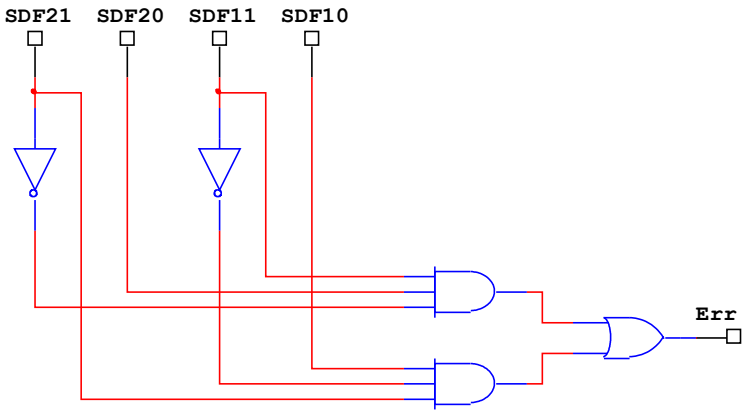


fig. 10

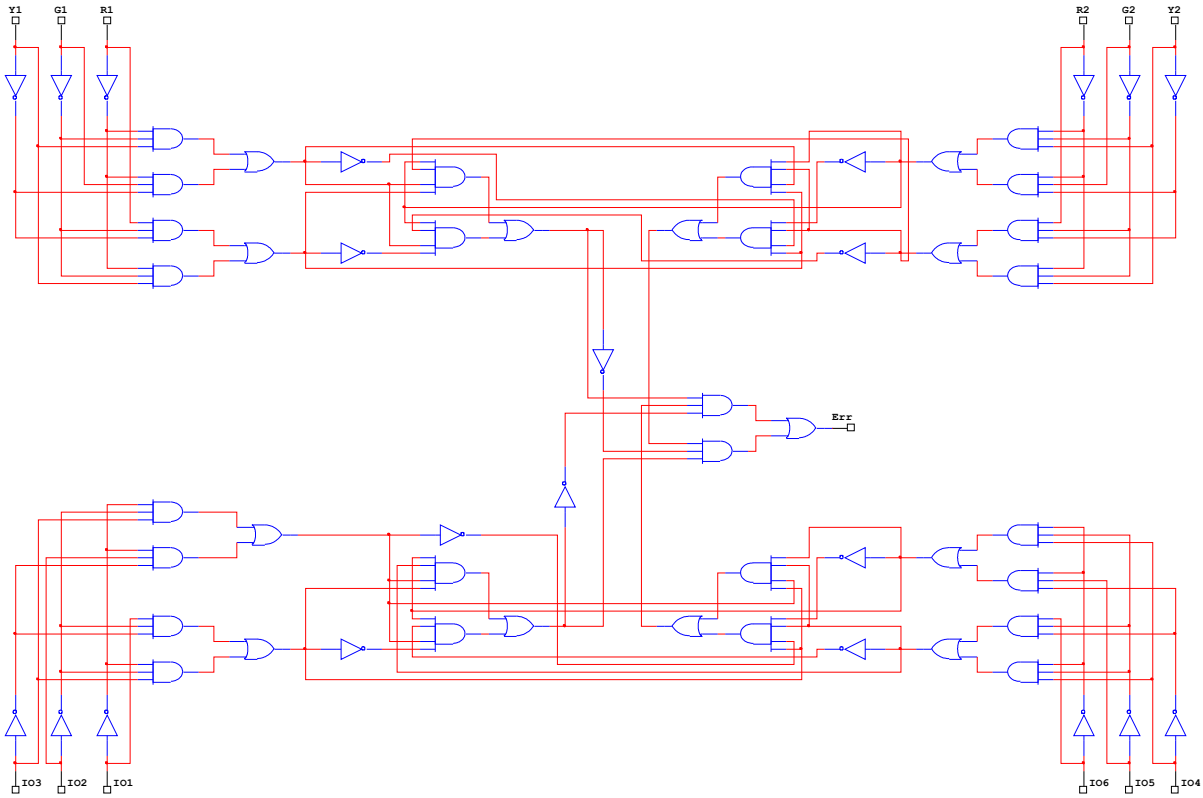


fig. 11

Conclusions

This paper has tried to explain how we can face a problem of medium complexity, by splitting it into several problems of low complexity, applying the *divide et impera* strategy, and solving each one of them using canonical synthesis methods.

We have also learned how a block must be identified and how it should be solved and, at the end of the paper, we have tried to face and solve a real problem.

What this paper presents is not the only way to solve a problem of combinatorial logic. There are several other techniques and strategies, but this one should be carefully studied by the student, because it represents a simple, neat and practical method.