

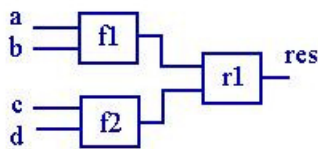
Title: Logic circuits with gates and inverters in ‘chained’ expressions.

Author: Peter Lablans, Ternarylogic LLC.

Date: July 12, 2005

Composite binary expressions.

Assume that the following logical expression needs to be realized in a binary switching circuit: $res \rightarrow (a \ f1 \ b) \ r1 \ (c \ f2 \ d)$. Assuming that f1, f2 and r1 are binary switching functions, the following diagram provides a possible solution for realizing the expression.



This diagram requires two levels of circuitry it has to realize. First the expressions (a f1 b) and (c f2 d) have to be executed. Once the results of these expressions are available then they can be used as inputs to the circuit that executes function r1.

It should be clear, but perhaps not readily apparent, that the order of the expression and the realization of this expression depend on a pre-conceived notion how binary logic is realized in circuitry. A preferred way would be a technology executes the expression as a 4-inputs/single output device, as is shown in the following figure.



The expression $res \rightarrow (a \ f1 \ b) \ r1 \ (c \ f2 \ d)$ is then merely an equivalent for the function ‘logfun’.

There is at least one known way to realize such composite circuits in a ‘single-step’ method. This is by way of memory or look-up-table circuitry. The inputs of a circuit may therein be considered to be an address. One of the disadvantages of this method is that all required output states have to be individually realized.

Another way of looking at digital logic circuitry is by ‘reconstructing’ the realization of all 2 inputs/single output binary devices. This reconstruction is the focus of this article.

There are 16 truth tables describing the possible binary devices. The following table shows 14 of the 16 truth tables.

1		0		1	2		0		1	3		0		1	4		0		1	5		0		1	6		0		1	7		0		1
0		0		0	0		0		0	0		0		0	0		0		1	0		0		1	0		0		1	0		0		1
1		0		1	1		1		0	1		1		1	1		0		0	1		0		1	1		1		0	1		1		1

8		0		1	9		0		1	10		0		1	11		0		1	12		0		1	13		0		1	14		0		1
0		1		0	0		1		0	0		1		0	0		1		0	0		1		1	0		1		1	0		1		1
1		0		0	1		0		1	1		1		0	1		1		1	1		0		0	1		0		1	1		1		0

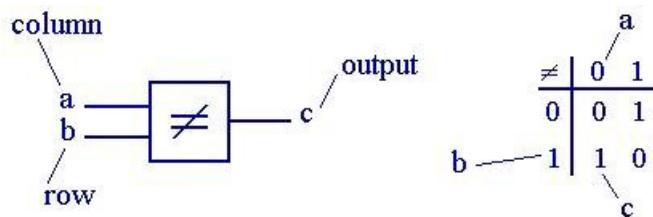
The all 0 and all 1 truth tables are not shown.

Other important binary functions are one-place functions (single input/single output functions) or inverters. The basic form of the binary inverter is (a b): [0 1] → [a b]. Or input 0 is transformed into ‘a’ and input 1 is transformed into ‘b’. The states ‘a’ and ‘b’ can assume ‘0’ or ‘1’. The inverter (0 0) outputs a 0 no matter what the input is. The function (1 1) generates a 1, no matter what the input is.

The inverters (0 1) and (1 0) are reversible inverters, wherein (0 1) is the Identity inverter I: (0 1): [0 1] → [0 1].

The inverter R= (1 0) reverses the input state: R: (1 0): [0 1] → [1 0].

Suppose the truth table of the XOR function (truth table 6 is the above table) needs to be realized. The following figure shows a two-inputs/single output diagram realizing the expression $c \rightarrow (a \neq b)$.



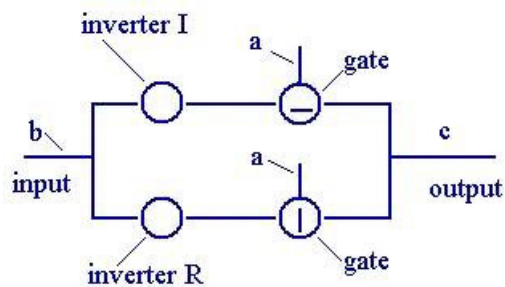
This diagram means the following: when input a=0 then b ‘sees’ the column under a=0 in the truth table. Or, when a=0 b ‘sees’ the inverter (0 1) or inverter I. When a=1 then b ‘sees’ the inverter (1 0) or the inverter R.

Assuming that the inverters I and R can be realized, then all that is required is a method to make the inputs ‘see’ either of the inverters.

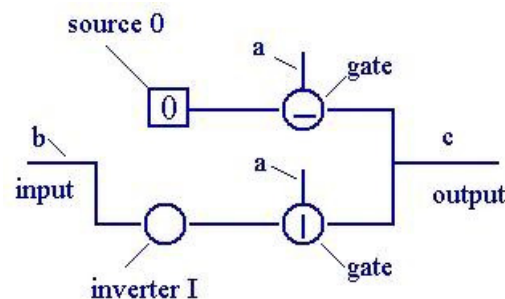
This can be realized by individually controlled gates. Each gate has a transmission part (from input to output) which is conducting under the condition of a control input.

In the following diagram one individually controlled gate is drawn as a circle with a horizontal line inside and a vertical line on top. The vertical line on top signifies a control input. The horizontal line indicates that the gate is conducting when the control input provides a signal related to state 0.

Another individually controlled gate is drawn in a similar fashion but with a vertical line inside, indicating that it is conducting when the control input provides a signal related to state 1. When a gate is not conducting it is non-conducting and no signal will appear on its output.



The following diagram shows the realization of the AND gate with inverters and gates.

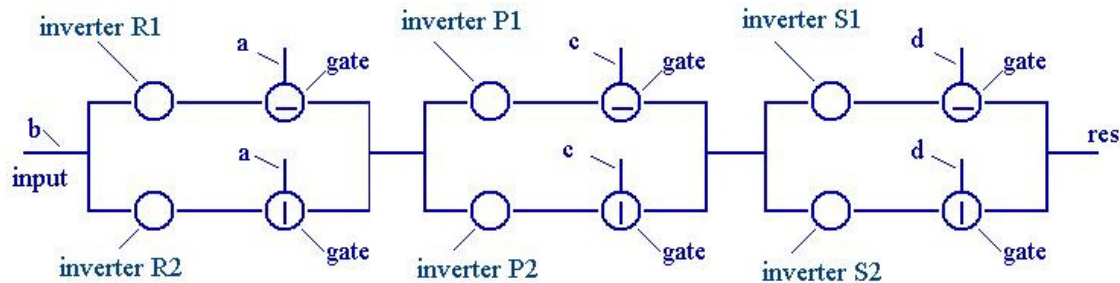


When $a=0$ then b ‘sees’ the inverter (0 0) in the truth table of the AND function. In the diagram it is assumed that ‘0’ is a state represented by a signal not equal to ‘absence of signal’. So a source needs to be inserted that always generates a signal representing state 0. Inverter I is of course a straight-through connection as it does not change its input.

The present set up will be able to create a composite expression. By re-formulating the example composite expression it can be executed in a simpler way.

Equivalent reformulation of composite logic expressions.

Assume that the circuit as shown in the following figure is available:



The following inverters are used:

R1: (r10 r11)

R2: (r20 r21)

P1: (p10 p11)

P2: (p20 p21)

S1: (s10 s11)

S2: (s20 s21)

The binary functions R, P and S are thus realized, having the following truth tables:

R	0	1	P	0	1	S	0	1
0	r10	r20	0	p10	p20	0	s10	s20
1	r11	r21	1	p11	p21	1	s11	s21

The circuit can be described by the following 'chained' expression:

$res \rightarrow \{[(b R a) P c] S d\}$.

Because input signals are all available at the same time, the circuit is switched on completely when the inputs are available.

For instance the expression: $res \rightarrow (a \neq b) \wedge (c \wedge d)$ which determines a carry bit in a 2-bit word by 2-bit word ripple adder is equivalent to the 'chained' expression:

$res \rightarrow \{[(a \neq b) \wedge c] \wedge d\}$.

The equivalent expression in this example is relatively simple. In general the transformation of 'traditional' expression to 'chained' expression is more complicated.

Comments.

The above method has been developed to enable n-valued 2-input/single output logic functions using gates and inverters. Different variants of this method have been developed and each has its advantages and disadvantages.

Switching theory is originally based on applying electromechanical switches. In those switching systems one of the states 0 or 1 is represented by 'absence of signal'. Switching theory by itself is well known. Parallel execution of circuitry is also known and was used by Kilburn in what is presently known as the Manchester Carry Chain approach.

Logic design using on/off switches in essence apply AND, OR and NOT functions. That type of switching theory is known as 'relay logic' or 'ladder logic' and is applied in Control Engineering and PLC circuit design.

The combination of gates and inverters and re-formulation of composite expressions by way of equivalent 'chained' expressions is a novel approach.

About Ternarylogic LLC: Ternarylogic LLC is a New Jersey based R&D company. Its mission is to create novel MVL technology. The company owns a portfolio of inventions related to scramblers/descramblers, sequence generators and sequence detectors, sequence correlators, gates and inverters based circuitry, non-binary multipliers, latches and other non-volatile memory elements, optical disk applications and MC-DSSS technology.