Digital Logic Gates (in one easy lesson)

by: Robert A. Rafuse

How many of you out there in the human race wonder where all these people that design and tell us about all those modern electronic circuits get all the brains? We know though, don't we! They stole 99.9% of ours and expect us to know what they are talking about. Are we content to stand back twiddling our thumbs and sing Yankee Doodle while the sun goes down? Heck no! We're going to start catching up in the knowledge department, and since we are getting interested in digital processing of synthesized music, then we night as well start with basic digital theory.

But where to start? I guess that you've all had that nightmare where a 14 legged IC, with the head of a dragon spewing hot solder, attacked your "two transistor-super-duper-do-nothing-circuit". Fear not. Just sharpen your sword...I mean pencil, put on your thinking cap, and we'll conquer that foe.

Let's first talk about the how and why of logic thinking. Even before the age of electronic computers, way back in the eighteen hundreds, an English gentleman by the name of George Boole was working with logic that is today the basis of all digital circuitry. George developed a mathematical system by which a logic statement could be proved to be either true of false. In logic thinking there are only two valid states: right or wrong, true or false, yes or no, etc. The result is that today Mr. Boole's system is with us in the form of Boolean algebra.

Wait now, don't stop reading! Just because I mentioned algebra don't tune me out. If your younger brother or sister is studying Boolean algebra in Jr. High School, and better yet understanding it, then there is still hope for you. The Boolean system relates directly to a mathematical system called binary. In the binary system there are only two possible states, a one or a zero. This system adapted very well to Boole's work and today is used in digital design.

The analogy of a switch seems to fit this idea very well. We can say that if a switch is closed we have a "1" and if it is open we have a "0". It doesn't really matter what type of switch it is as long as it can simulate the same principle of operation. It can be a mechanical one such as a toggle switch or an electronic one using such discrete components as resistors, diodes, and transistors. Let's look at the three basic building blocks found in digital logic: the AND gate, the OR gate, and the Inverter. With these gates, any circuits such as adders, counters, multipliers, dividers, and an entire computer can be constructed. Knowing their operation will give us a solid background from which we can build our digital knowledge.

The AND gate is a device whose output is a logic 1 if all of its inputs are a logic 1. Figure 1(a) shows the switch analogy of a two input AND gate. In this drawing we see two switches A and B which are in series. You can probably figure out that the only way current will flow through the load, and develop a voltage across it, is when both switches are closed. Seems too simple, doesn't it? Don't worry though, the equivalent logic circuit is just as easy. The accepted symbol for the AND gate is shown in figure 1(b). If we apply a value of voltage equal to +v with respect to ground to both inputs A and B, then the output of the gate (X) will have a value of V with respect to ground. The value of +V is

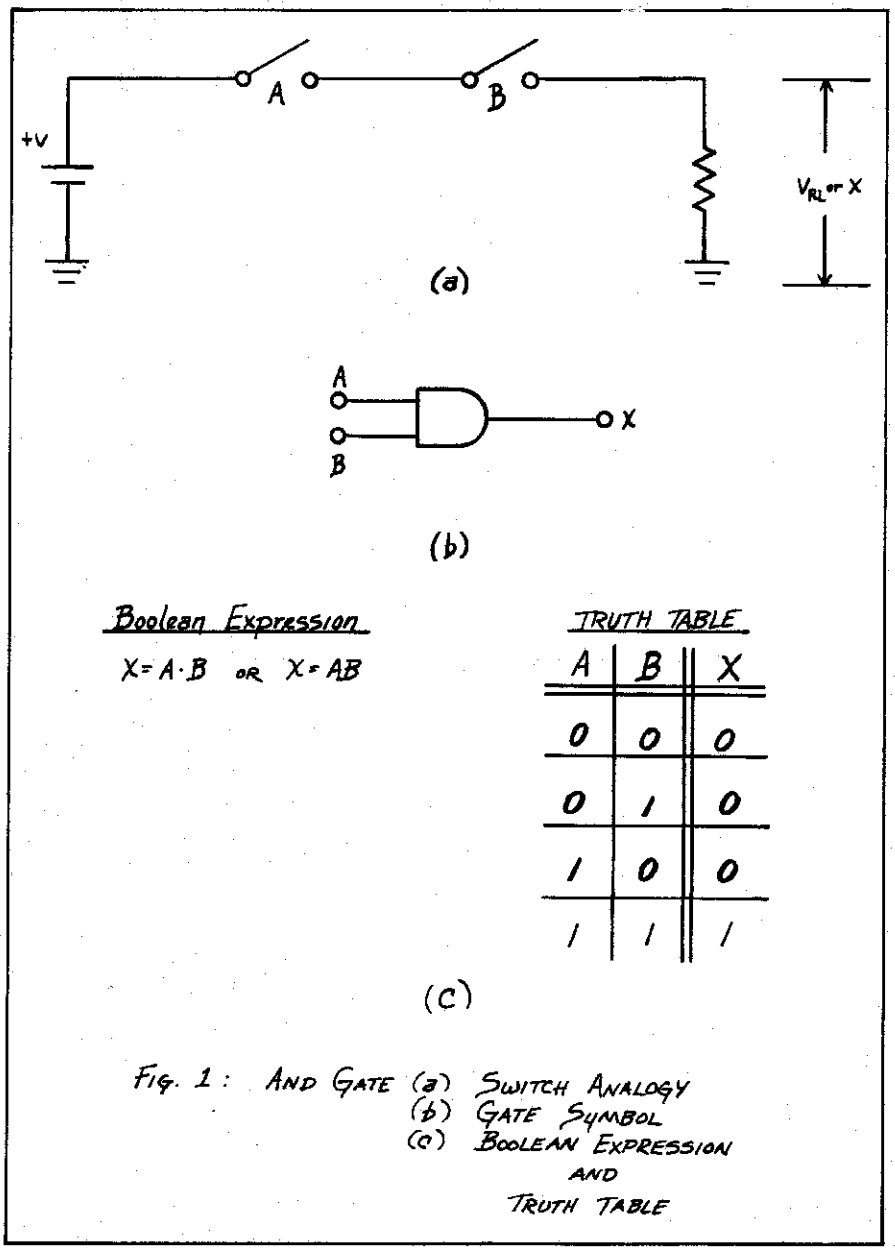


Fig. 1: And Gate (a) Switch Analogy (b) Gate Symbol (c) Boolean Expression and Truth Table

Where's Walter?

TTL logic which operates at a supply voltage of 5 volts. The value of the low or 0 level is between 0 and .8 volts. The 4000 CMOS series logic is the same when operated at a supply voltage of 5 volts. CMOS logic can be operated from a supply voltage between 3 and 15 volts. Any value of voltage between the accepted 1 or 0 states is invalid and undesirable.

For the AND gate we can write a Boolean expression and a "truth table". Both are shown in figure 1(c). The expression is usually read as "X equals A and B" or "X equals AB'. The truth table is just a tablature form of stating the value of X for all possible combinations of A and B. The two input AND gate is available in various DIP packages with different switching speeds and power consumption. (Figure 2) shows a 7408 quad 2 input AND gate package. The

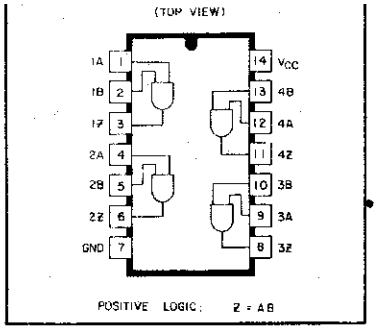
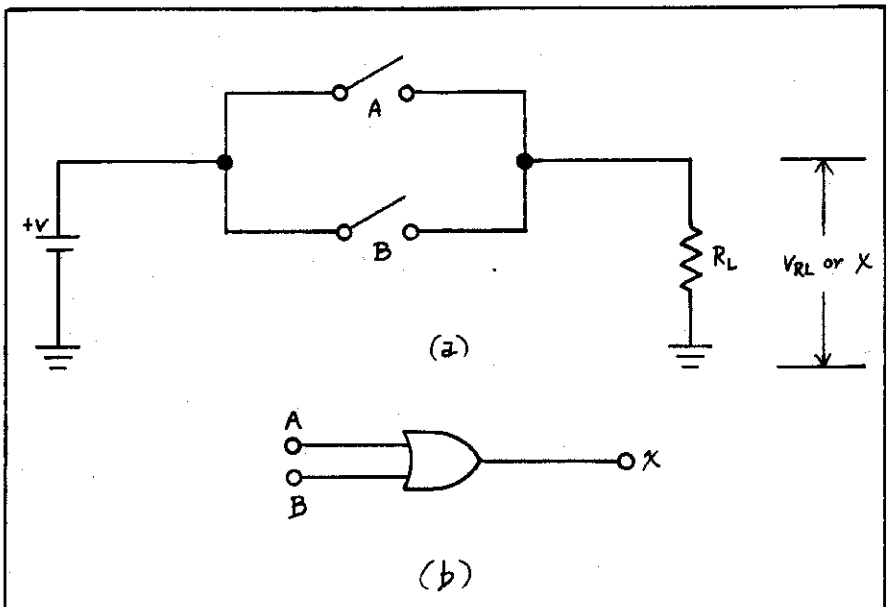


FIG: 2

7408 only costs abouth 25 cents from any surplus dealer, so why not buy one and experiment with it to see if I'm telling the truth.

The next gate I would like to talk about is the OR gate. Continuing on with



Boolean Expression

$X = A + B$

Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

(c)

FIG. 3: OR GATE (a) SWITCH ANALOGY
(b) GATE SYMBOL
(c) BOOLEAN EXPRESSION AND TRUTH TABLE

A and B in parallel. We know that if we close either or both switches, current will flow through the load resistor and the voltage across it will be +V or a logic 1. Figure 3(b) shows the accepted symbol for a 2 input OR gate while figure 3(c) contains the Boolean expression and truth table. The expression is usually read as "X equals A or B". The + symbol is read as OR and should not be confused with the mathematical addition symbol. The 2 input OR gate is available in a quad package in both CMOS and TTL with the TTL 7432 shown in figure 4.

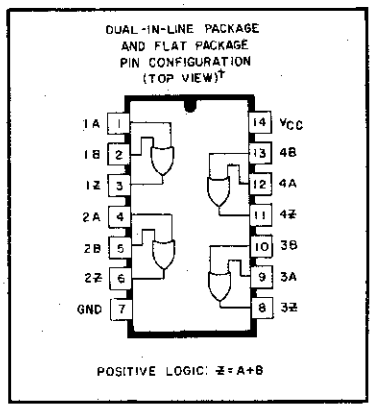


FIG: 4

The third and simplest gate so far is the Inverter or "Not" gate. Two accepted symbols for the Inverter are shown in figure 5(a). The Inverter doesn't really make any decision as do the AND and OR gates. The inverter has only one input and the output X is always the inverse of A. If we apply a logic 1 and A, then the output X will be a 0, while a 0 at A will cause X to be a 1. This is shown in the truth table of figure 5(b) along with its Boolean expression. The equation is read as "X equals A not" or "X equals not A". The Inverter is available as a "six pack" in both CMOS and TTL, with the TTL 7404 shown in figure 6.

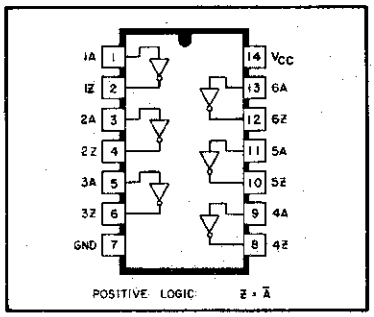
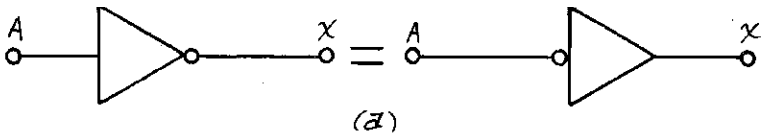


FIG: 6

Well there you have it, the three digital gates that form the basis for any logic function. I bet you're saying "I don't believe it!". I am sorry, but that is all there is to these gates. Let's see



(a)

BOOLEAN EXPRESSION

$X = \overline{A}$

TRUTH TABLE

A	X
0	1
1	0

(b)

FIG. 5: INVERTER (a) SYMBOLS (b) BOOLEAN EXPRESSION AND TRUTH TABLE

if you have absorbed some of the things I have said, so take a look at the gates in figure 7. Try writing down the name

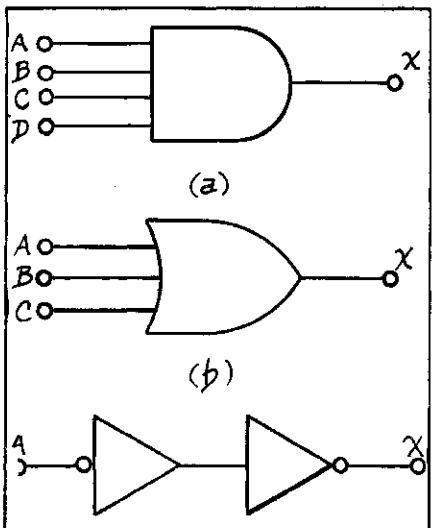
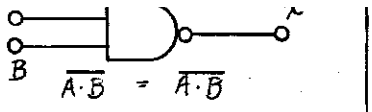


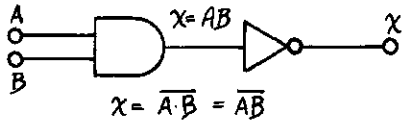
FIG. 7: QUIZ

and Boolean expression for each drawing without reading any further. If you have written for (a) 4 input AND and X=ABCD, (b) 3 input OR and X=A+B+C, (c) two inverters and X=A, then you are doing great. If you had any right for that matter, congratulations! See, even you can understand some digital logic. Tomorrow, the world. You see that no matter how many inputs there are, you just expand the expression to fit the extra inputs. For the inverter circuit you can see that by inverting something twice, you arrive right back where you started! This is a very important point and you should try to remember it.

I suppose you have been wondering about the result of combining these gates together. Well, the next two gates I'll talk about, you probably have seen in most logic circuits you've looked at. The first combination we'll look at, shown in figure 8(a) is called a NAND gate. If you think the English language is taking a beating don't worry, because for the sake of brevity, it is an abbreviation for "Not And". Look at figure 8(b) and you will soon understand. The NAND gate consists of an AND gate whose output is inverted, with inversion indicated by the small circle at the output of the gate. The Boolean expression shown has a bar across the top of the equation to represent inversion of the AND function. Looking at the truth table of figure 8(c) we see that the output is similar to the AND gate but everything is inverted. Remember what I said about the inverter?



(a)



(b)

TRUTH TABLE

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

(c)

FIG. 8: NAND GATE (a) SYMBOL AND BOOLEAN EXPRESSION (b) GATE EQUIVALENT (c) TRUTH TABLE

Put an Inverter on the output of a NAND gate as in figure 9(a) and we obtain an output equivalent to the AND gate. Pretty neat, eh! Figure 9(b) shows how to make an Inverter using a NAND gate. With both inputs tied together, the result is that whatever is applied at A is inverted at the output X.

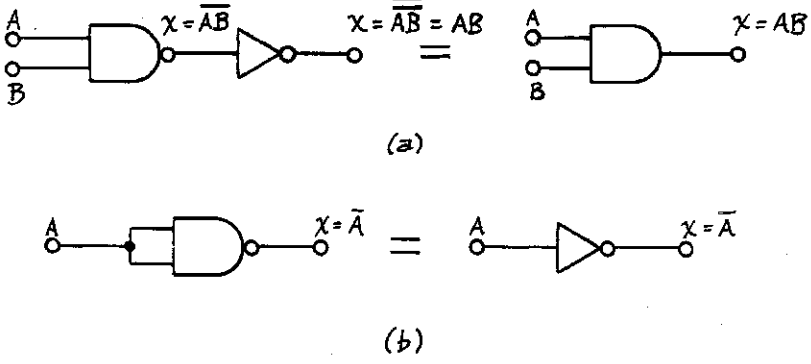
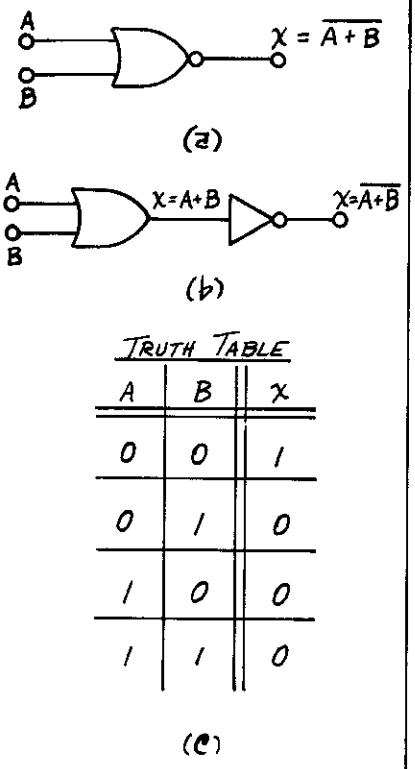


FIG. 9: NAND GATE (a) CONVERTED TO AN AND GATE (b) USED AS AN INVERTER

reasons you probably already have deduced. The NOR consists of an OR gate



TRUTH TABLE

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

(c)

FIG. 10: NOR GATE
(a) SYMBOL AND BOOLEAN EXPRESSION
(b) GATE EQUIVALENT
(c) TRUTH TABLE

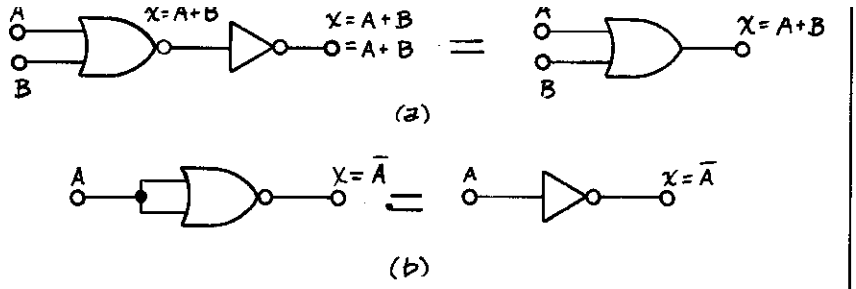


FIG. 11: NOR GATE (a) CONVERTED TO AN OR GATE
(b) USED AS AN INVERTER

whose output is inverted, this being shown in figure 10(b). The small circle again indicates inversion and is also depicted by the bar in the Boolean expression. The truth table is shown in figure 10(c). As you probably have already guessed the NOR can also be converted to an OR gate by inverting the output (figure 11(a)) and an Inverter with the two inputs tied together (figure 11(b)).

Both the NAND and NOR gates are available in various number of inputs and gates in DIP packages. The reason for making these gates with the NAND and NOR is that when building logic circuits, the inverter is often used. Here comes the big clincher. We can draw the NAND and NOR two different ways without changing the operation of the gate. Figure 12 (a) shows the NAND gate and you'll find if you run through a set of input conditions the result for both will be the same.

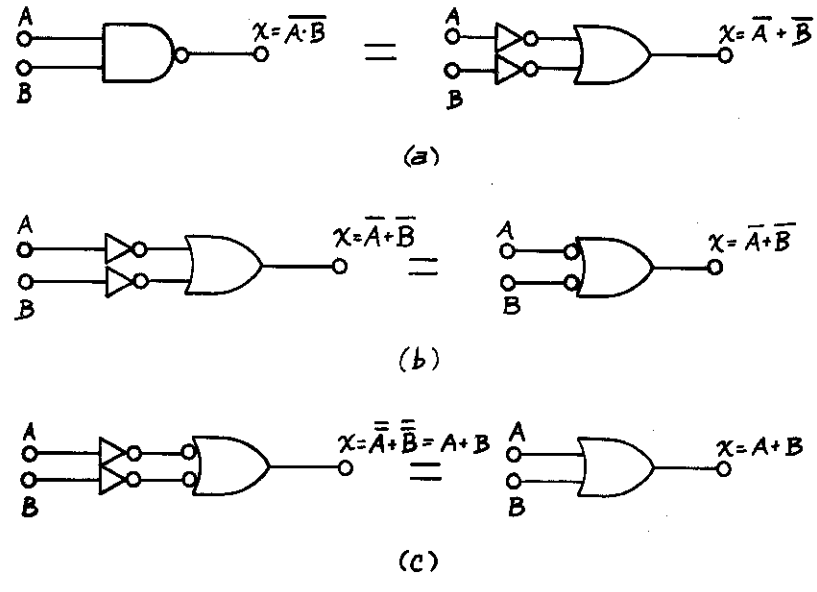


FIG. 12: NAND GATE (a)(b) EQUIVALENT CIRCUITS
(c) DOUBLE INVERSION EQUIVALENCY

Also, I should mention that the inputs that are inverted can also be drawn as shown in figure 12(b). I hate to be repetetive, but remember what I said about inverters? Why not invert both inputs as I do in figure 12(c) and presto! We have created an OR gate, and you thought it could never be done. The same is true for the NOR gate with the two equivalent drawings shown in figure 13(a). By inverting the two inputs we have now created an AND gate as seen in figure 13(b). Look at the gates of figure 14 and you will see all the gates described and their equivalent symbols.

Last but not least, before you take your commercial break, I should mention one more gate called the "Exclusive OR" which is used in many logic circuits. The symbol for the Exclusive OR is shown in figure 15(a) and its truth table in figure 15(b). We can immediately see that the only time we have an output X is when the inputs are opposite each other. The + symbol in the Boolean expression indicates the Exclusive OR operation. The gate functions like an OR gate but excludes the A=1, B=1 state where the output is inverted.

So there you have it. The layman's guide to digital gates in one easy lesson. The reason why the NAND and NOR are used is that any digital logic circuit, no matter how complex, can be constructed using AND and Inverter gates or OR and Inverter gates. So if you have to convert a NOR to an OR, NAND to an OR, etc., you now have some idea how to do it. Some of the terminology might be a little strange at first, but after reading this article over a few times, "no sweat". If you're interested in learning more, maybe you can pick up a book on the subject at you local library, or better still buy one of those available from digital IC manufacturers such as RCA or National Semiconductor. These latter books usually have something on theory, specifications, and applications. Who knows! Maybe the next "million-dollar-IC-do-nothing-circuit" will be your own.

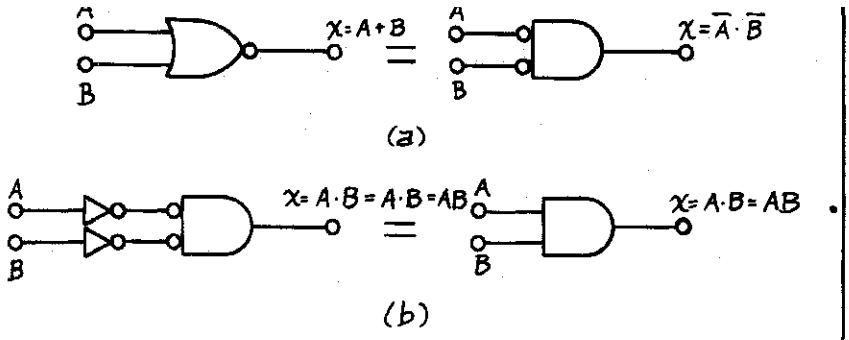


FIG. 13: NOR GATE (a) EQUIVALENT CIRCUITS (b) DOUBLE INVERSION

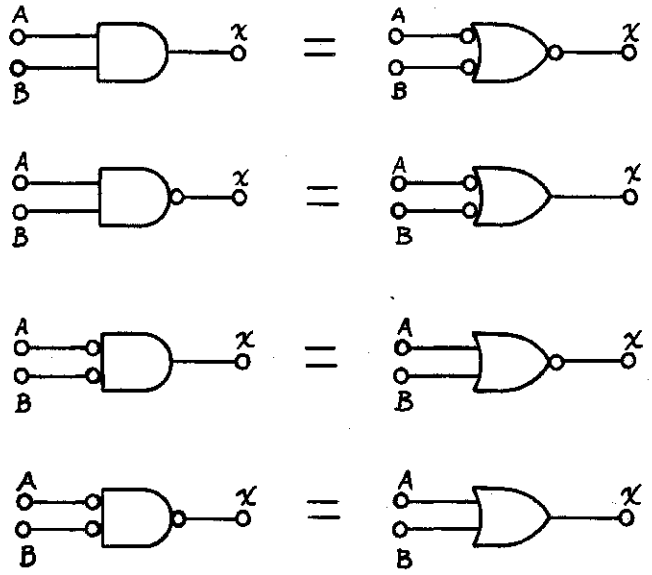
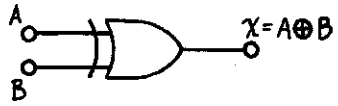


FIG. 14: EQUIVALENCY CHART



(a)

TRUTH TABLE

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

(b)

FIG. 15: EXCLUSIVE OR; (a) SYMBOL AND BOOLEAN EXPRESSION (b) TRUTH TABLE

Oops! It seems that the gremlin struck the "Random Tone Generator" article, Polyphony, July 1977. We offer our apology and the following corrections supplied by the author...

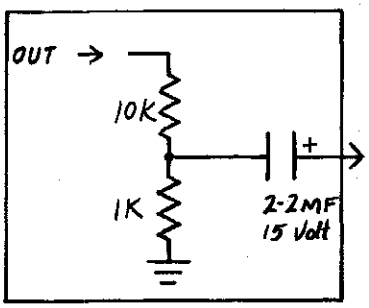
There are a few corrections to the schematic diagram I think you (or your readers) may like to know about.

1) The .22pf capacitor should be .22uf

2) The 4.7pf capacitor should be .47uf.

3) The unlabelled fixed resistor in series with the "rate" pot should be marked 100K ohm.

Another item I should have mentioned in the article (but neglected to) is the following: the auxiliary output (labelled 'to synthesizer') is a 9 volt square wave which is a bit too high for input to PAIA modules. Though it will still work, I'd recommend lowering the amplitude with a resistive divider something like this:



AUX. OUTPUT

Otherwise, no doubt, the module would probably "clip it"... it will function, but distortion would no doubt result.

-Kenny Winograd-