Chapter 3

Building Static Figures

For the purposes of this book, *static figures* are considered any figures, no matter how simple or complex, that do not move on the screen. Such figures, in the context of TV games, can represent game boundaries, obstacles and nonmoving targets.

A static figure, incidentally, does not have to be a visible one. Invisible figures can add an extra bit of challenge to an otherwise too-simple game; but more importantly, invisible lines and rectangles can serve as "windows" for confining other images to a field that is smaller than the TV screen.

Static figures, whether visible or not, play invaluable roles in TV-games technology; and it turns out that building static figures can be a fascinating and rewarding pastime in itself.

This chapter describes a number of basic techniques for building static lines, bars, and rectangles, while the following chapter takes up the special subject of building complex static figures (circles, figures with diagonal lines, images of people, rocket ships, cars, etc.).

Each technique is illustrated with several specific examples, and the reader will find the procedures used many times and in many different ways throughout the book. It is important to bear in mind, however, that the purpose of this book is to provide some guidelines for creating original TV games and displays. For that reason, alone, it is just as important to understand the essence of each figure-building technique as it is to see how the specific examples work.

None of the figure-building techniques is really any better than the others under all possible circumstances. The most appropriate technique depends on the size, position, general configuration, complexity of the figure, and the role the figure is to play in the display.

The selection of a figure-building technique thus calls for an intelligent decision on the part of the experimenter, so it is important to study this entire chapter and the one that follows before making a firm commitment to the circuit for an original TV-game display.

A prudent student of TV-game technology will breadboard the circuits as they are encountered in these chapters. Of course it takes longer to become acquainted with all the circuits this way, but this doing-while-reading approach makes the learning process much easier, more effective, and a whole lot more fun. Try building the specific examples first, then test your understanding by attempting to create a few static images of your own.

LINES AND BARS DIRECTLY FROM THE COUNT SOURCES

One of the simplest and most straightforward ways to create simple lines and bars on the screen is by combining the wave-forms already available from the Sourcebox unit. All of the horizontal-count outputs quite naturally generate alternate black and white vertical bars or lines on the screen, while the vertical-count outputs create horizontal bars.

Figure 3-1 summarizes five different horizontal- and vertical-count signals as they appear on the TV screen. To view the figures as shown here, simply connect a jumper wire between the designated horizontal- or vertical-count output and the GAME VID IN terminal on the Sourcebox unit. This procedure, incidentally, assumes the CVID and MOD terminals on the Sourcebox unit are jumpered together as described in the previous chapter.

Note that the 256H display shows a vertical black bar that almost reaches the center of the screen. This is a clear indication that the 256H signal is at logic 0 through the first half of each horizontal trace. (A logic-0 video signal always creates a black area on the screen, and a logic-1 signal creates a white area).

The figure for 128H shows twice as many vertical bars as the 256H signal does. The reason is rather easy to understand: the 128H frequency is twice that of the 256H count source. And in a similar manner, each lower-order H output shows about twice as many bars as the one preceding it. Outputs 8H, 4H, 2H, and 1H are not shown here because the lines are too fine and closely spaced to make a meaningful picture in the book. Look at them on your own TV screen, however.

A figure for 256V is not shown here because it does not make a very interesting picture. It is at logic 0 through all but the last four lines of the display. Figure 3-1 does show what you should expect to see from 128V, 64V, 32V, and 16V.

While inspecting these figures for yourself, take careful note of the fact that the bars for 64H and 64V are very close to the same size. The same is true for 32H and 32V, 16H and 16V, etc. From 64H and 64V downward, the lines are about the same size when comparing an H output with its vertical counterpart.

There is, however, an obvious difference between the sizes of the bars for 128H and 128V, and 256H and 256V. Keep these facts in mind for a time when you will be considering your own figure-building procedures.

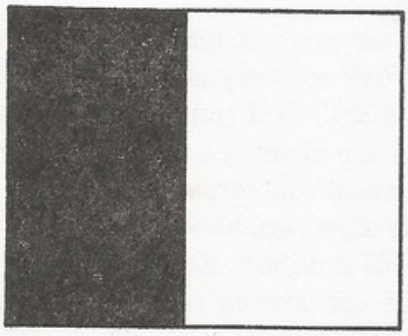
An experimenter can be justifiably proud of his system when seeing these basic horizontal- and vertical-count bars for the first time. A lot of work and money has gone into building the Sourcebox, and this is the first solid result of all that work. But looking at these bars and lines can become rather boring after a while. So now it is time to begin using these lines and bars as mere building blocks for creating more-useful and interesting figures.

THE LINE/BAR TINKERBOX

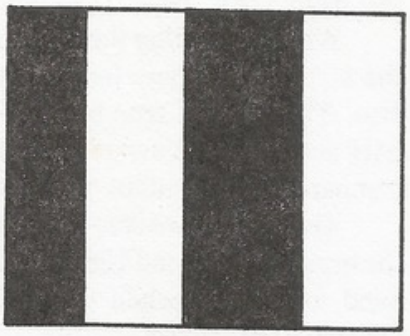
Figure 3-2 shows a schematic diagram, physical layout, and parts list for a breadboard system we shall call the Line/Bar Tinkerbox. It is simply a selection of ICs that are most useful for creating certain lines and bars at desired positions on the screen. The main purpose of this Tinkerbox is to let the experimenter find out exactly what IC devices are necessary for building a desired line or bar on the screen. After taking careful note of how it is done with the Tinkerbox, the experimenter can transfer the ideas to a permanent cirucit.

The ICs are plugged into a standard breadboard and bus-strip assembly. (See the parts list in Fig. 3-2 for catalog numbers.) The breadboard assembly is connected to the Sourcebox unit via a multiconductor cable or bundle of wires. These wires are connected to the Sourcebox output connector by means of a standard 22-pin edge-card PC board. Figure 3-3 shows a connection diagram that corresponds with the Sourcebox output terminal configuration described in Chapter 2.

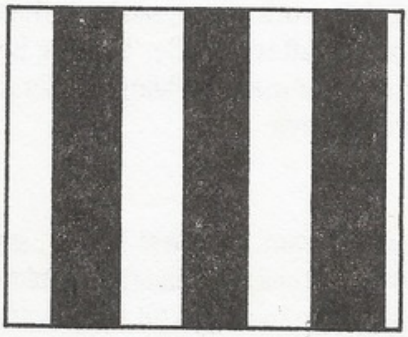
To get a feeling for what the Tinkerbox can do and how to use it, connect the circuit shown in Fig. 3-4a. In this case, 256H is connected to pin 1 of IC3, the 8-input NAND gate. If a test jumper is



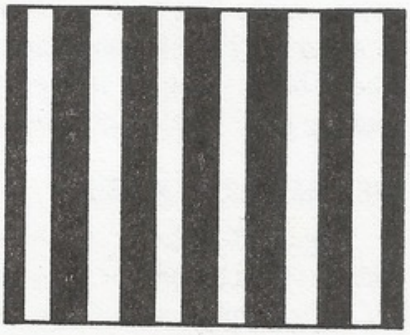
256H



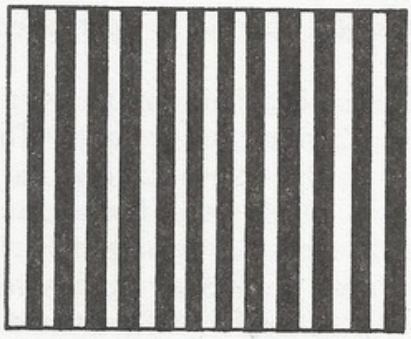
128H



64H



32H



16H

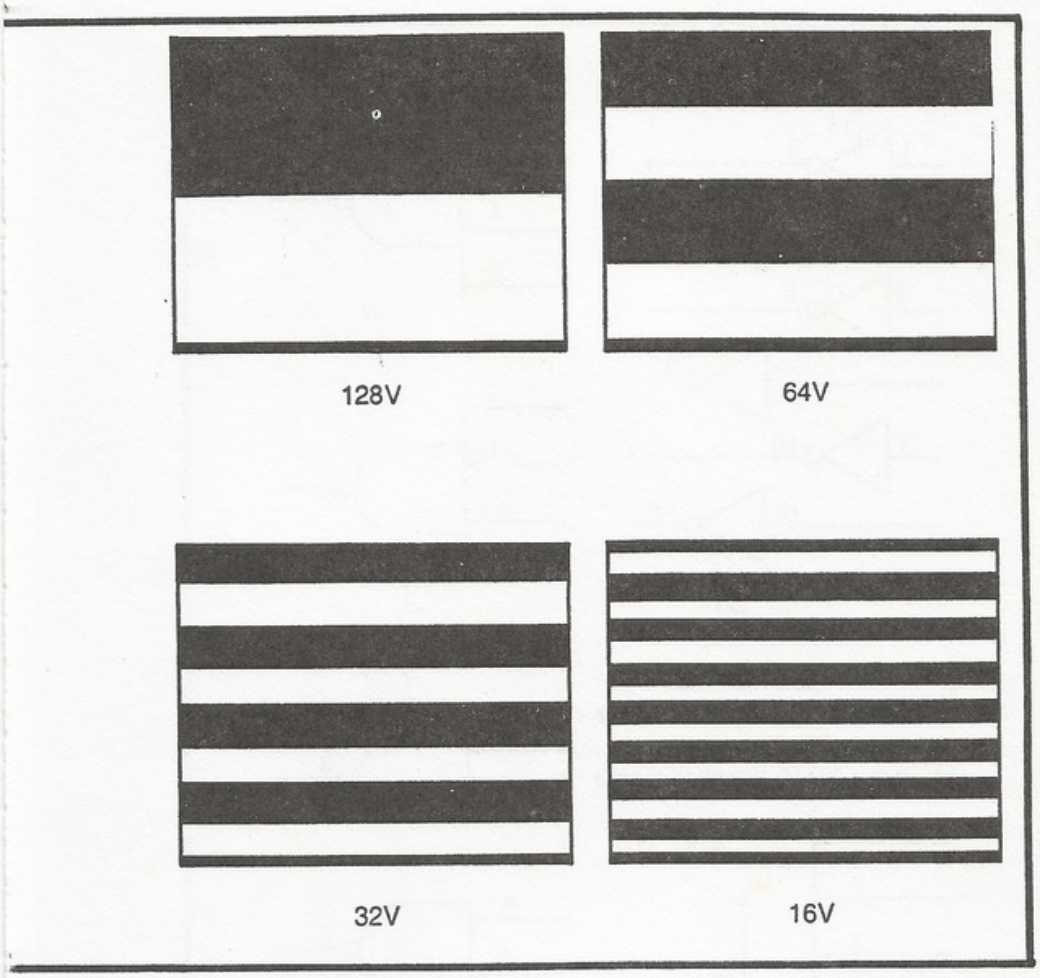


Fig. 3-1. Video images available directly from the horizontal and vertical-count sources.

connected from that input point and the GAME VID IN terminal on the Sourcebox, the screen will show the usual 256H pattern.

But once that same signal passes through the NAND gate, the blacks and whites are reversed. In digital terms, a NAND gate with a single input works like a logic inverter—it reverses the logic-1 and -0 levels.

The signal from the NAND gate is then passed through an inverter circuit where the waveform is reversed once again. This operation brings the signal back to its original phase, thus creating an image on the screen that is in all respects the same as that created by the original 256H waveform.

Any horizontal- or vertical-count waveform can be applied to the input of the circuit in Fig. 3-4a, and emerge in its original form from the output of the inverter circuit, IC1-A.

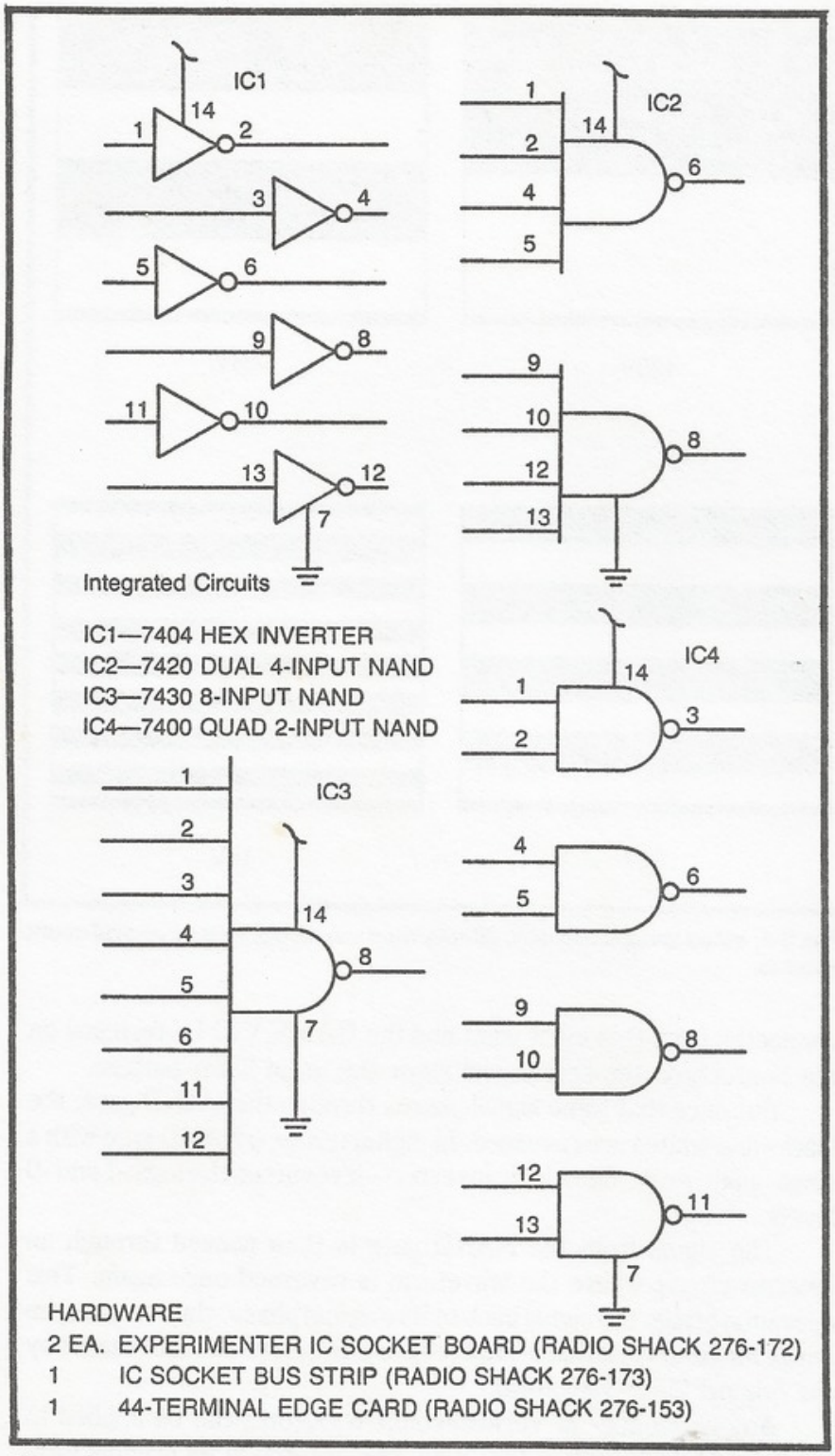


Fig. 3-2. Most-used ICs for the Line/Bar Tinkerbox assembly. Other useful IC's include a 7486 quad EXCLUSIVE-OR and a 7402 quad 2-input NOR gate.

So what is accomplished by the simple circuit in Fig. 3-4a? Not much in terms of building images for TV games. But the circuit does clearly demonstrate two important facts: a NAND gate inverts logic levels (and hence reverses blacks and whites), and doubly inverting a signal returns it ot its original phase.

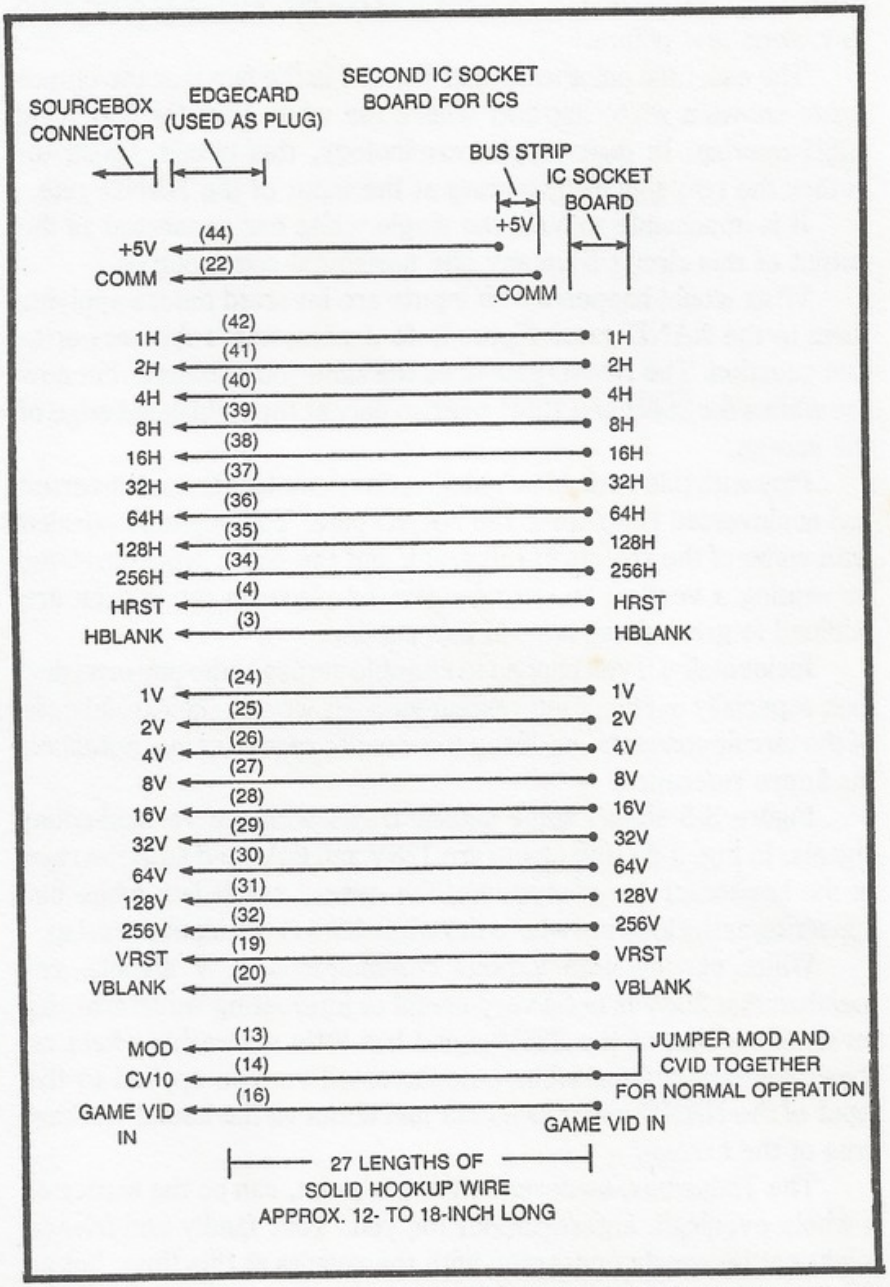


Fig. 3-3. Signal and power supply connections between the Sourcebox unit and Tinkerbox assembly. Note that the Sourcebox MOD and CVID terminals are connected together at the Tinkerbox.

Now wire the circuit in Fig. 3-4b. There are two inputs in this instance. The waveform from 256H is applied to an inverter circuit and then to one input of the 8-input NAND gate. The other input, 128H, goes directly to a second input of the NAND gate. The little screen figures accompanying the diagram in Fig. 3-4b show the patterns appearing on the TV screen as GAME VID IN is connected to various test points.

The essential point to note in Fig. 3-4 is the fact that the output figure shows a white bar only where the white bars for $\overline{256H}$ and 128H overlap. In digital logic terminology, this circuit ANDs together the two signals appearing at the input of the NAND gate.

It is impossible to build the single white bar generated as the output of this circuit from any one horizontal-count output.

What would happen if *both* inputs are inverted before applying them to the NAND gate? Figure 3-4c demonstrates the answer to that question. The NAND gate does the same job as before, but now the whites for $\overline{256H}$ and $\overline{128H}$ overlap only at the right-hand edge of the screen.

Play with this basic idea, using various combinations of inverted and noninverted H inputs to the NAND gate. You might be puzzled with some of the results at this point, but the exact procedures for generating a vertical line at one desired place on the screen are outlined in great detail later in this section.

Incidentally, if you happen to stumble across some patterns that look especially useful or interesting for later work, take careful note of the circuit connections. Keep the results recorded in a notebook for future reference.

Figure 3-5 shows some similar tricks with the vertical-count signals. In Fig. 3-5a, the inputs are 128V and 64V; and as in the case of the horizontal demonstrations, the overall result is a white bar appearing at the location where the white bars at the inputs overlap.

White playing with various combinations of V signals, remember that 256V is not a very useful or interesting waveform. An inverted version of the 256V signal has little noticeable effect on these demonstrations, while a noninverted version applied to the input of the NAND gate blanks out just about all the useful working area of the screen.

The Tinkerbox, as described to this point, can be the source of a whole evening's entertainment for you. Your family and friends might not be overly impressed with the results at this time, but as long as you are having fun with the system and learning things as you progress, that is all that really counts.

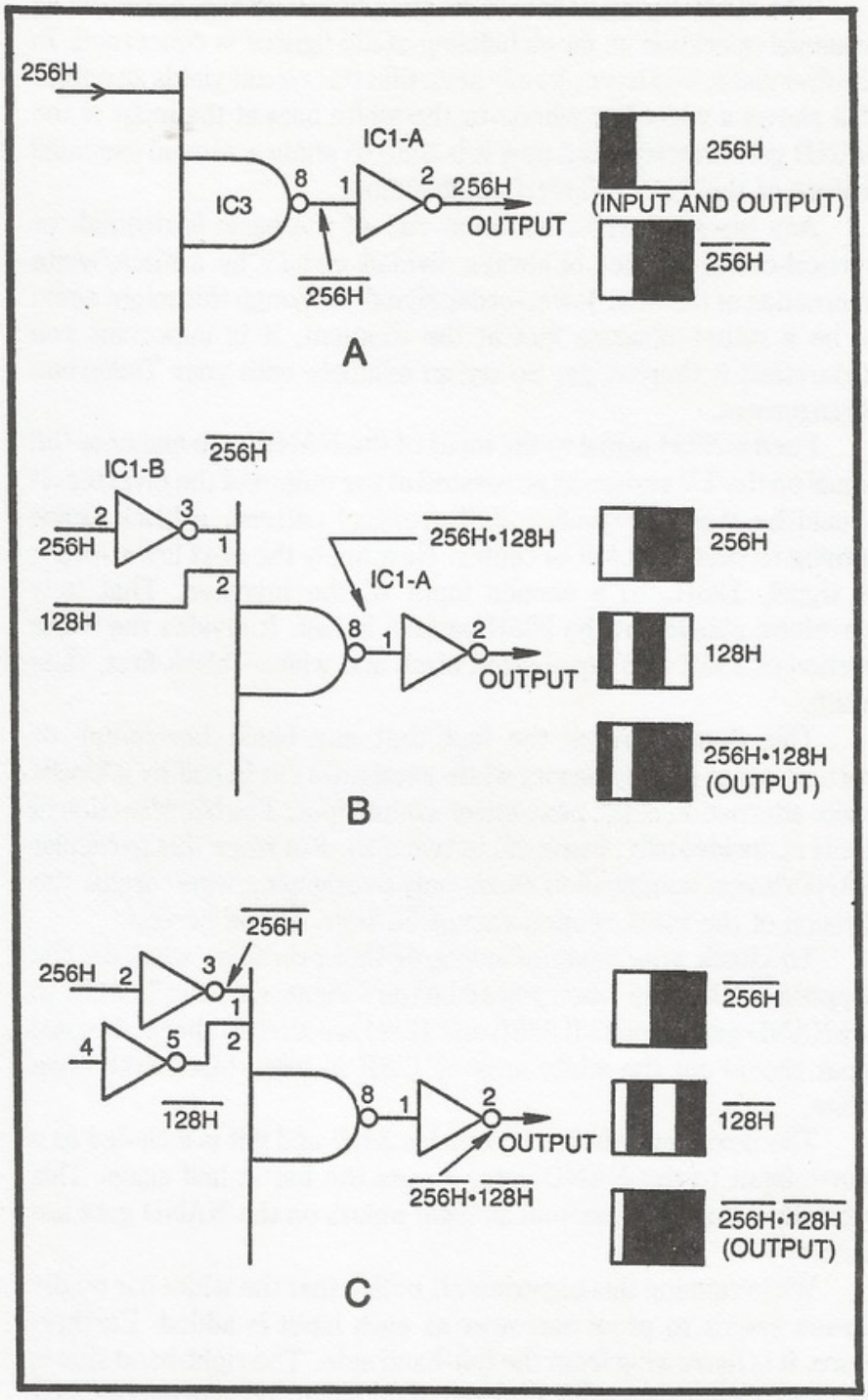


Fig. 3-4. Some vertical lines and bars from the Tinkerbox assembly. (a) A twice-inverted 256H signal yields a 256H pattern on the screen. (b) A NAND gate followed by an inverter combine two different horizontal-count signals such that the resultant is a white area where two white areas of the original signals overlap. (c) Another example of creating a moderately narrow, vertical white bar where two input signals have overlapping white areas.

The NAND gate, followed by an inverter circuit, performs an essential operation as far as building static figures is concerned. In the first place, you have already seen that this circuit yields an output that shows a white bar wherever the white bars at the input of the NAND gate overlap. And now it is time to study a second essential feature of this NAND/invert combination.

Any black or white bar from one of the basic horizontal- or vertical-count sources is always divided equally by a black/white alternation of the next-lower-order signal. Although this might seem to be a rather obscure fact at the moment, it is important you understand it thoroughly. So try an example with your Tinkerbox arrangement.

Feed a 256H signal to the input of the NAND gate and note the signal on the TV screen as generated at the output of the inverter. It should be the now-familiar 256H vertical pattern: a black space turning to white just left of center. Now apply the next lower-order H signal, 128H, to a second input of the inverter. That new waveform should cut the 256H pattern in half. It divides the white portion of 256H into equal-sized black and white—black first, then white.

This demonstrates the fact that any basic horizontal- or vertical-count display has its white section(s) cut in half by a black/white alternation of the next-lower-count input. The black section of 256H is, incidentally, being cut in two also. But since this particular NAND/invert combination yields only overlapping white areas, the division of the black section cannot be seen on the screen.

To check your understanding of this principle, what do you suppose will happen when you add a third input, specifically 64H, to the NAND gate circuit? If 256H and 128H are already there, the 64H input should cut the white area of 128H in half—black first, then white.

The next-lower-order H signal is 32H; and if it is included as a fourth input to the NAND gate, it cuts the bar in half again. This procedure can continue until all eight inputs on the NAND gate are used.

While running this experiment, notice that the white bar on the screen seems to grow narrower as each input is added. Furthermore, it is narrowing from the left-hand side. The right-hand side is remaining fixed. Bear this effect in mind while considering the next set of experiments.

Remove all inputs to the 8-input NAND gate except 256 H. The image on the screen should then be the standard black-to-white 256H signal. Apply the 128H signal again, but run it through an

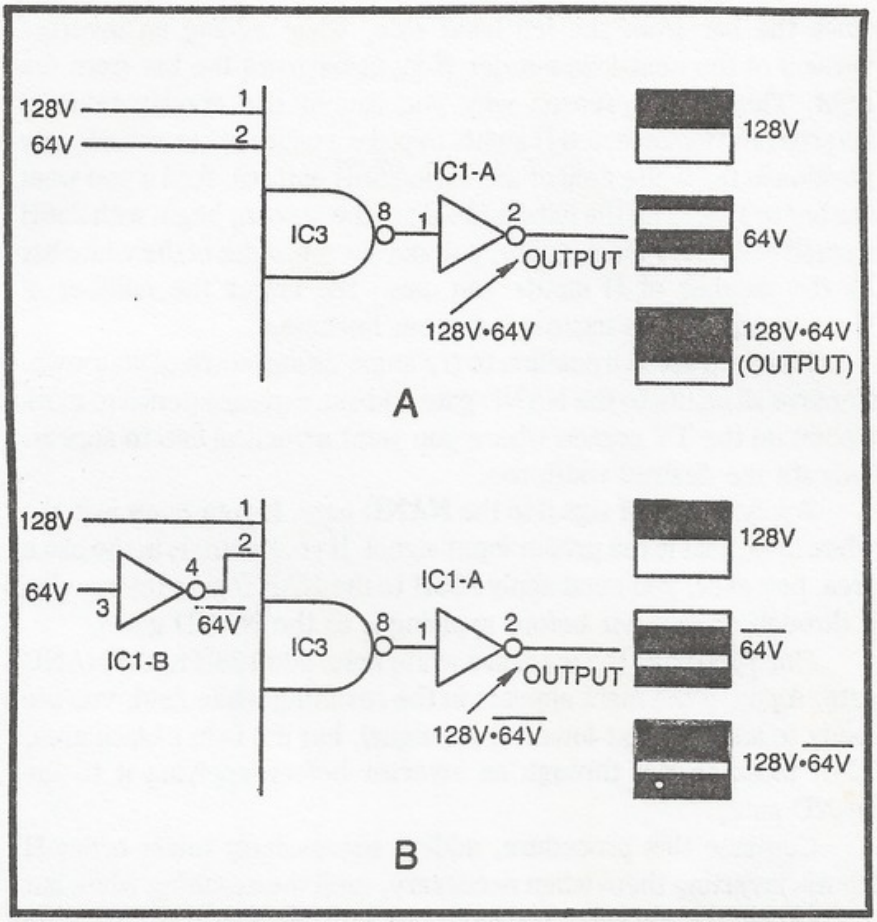


Fig. 3-5. Some horizontal lines and bars from the vertical-count sources. (a) A wide white bar at the bottom of the screen representing the area where white areas of 128V and 64V overlap. (b) Building a broad, horizontal bar just below the center of the screen.

inverter first. The inputs to the NAND gate are now 256H and 128H. How has this inversion of the 128H signal influenced the pattern on the screen?

The white portion of the 256H signal is cut in two by a white/black alternation this time. Recall that connecting a noninverted version of 128H to the NAND gate also cut the white portion of the 256H image in two, but with a black-to-white alternation.

Continue adding *inverted* versions of the H-count signals to the NAND gate until all eight inputs are used. If you are adding these inputs in decreasing order—from $\overline{128H}$ to $\overline{64H}$, to $\overline{32H}$, to $\overline{16H}$, etc.—you will see that the white bar on the screen narrows in from the right-hand side.

As far as the line-building effect on the TV screen is concerned, adding a noninverted version of the next-lower-order H input nar-

rows the bar from the left-hand side, while adding an inverted version of the next-lower-order H input narrows the bar from the right. There is no reason why you cannot use combinations of inverted and noninverted H inputs to place a white bar in virtually any position in the white field of the basic $\overline{256H}$ pattern. And if you want the bar to appear on the left-hand side of the screen, begin with 256H instead of 256H. Furthermore, you can set the width of the white bar by the number of H inputs you use: the larger the number of H-count inputs, the narrower the bar becomes.

Now you are in a position to try some design work of your own. Remove all inputs to the NAND gate and use a grease pencil to mark a point on the TV screen where you want a vertical line to appear. Indicate the desired width too.

Apply the 256H signal to the NAND gate. If your mark is in the white area, that is the proper input signal. If your mark is in the black area, however, you must apply 256H to the NAND gate (by running it through an inverter before applying it to the NAND gate).

Once you have the mark in a white field, add 128H to the NAND gate. Again, if the mark appears in the resulting white field, you are ready to add the next-lower-order signal, but if it is in a black area, 128H must be run through an inverter before applying it to the NAND gate.

Continue this procedure, adding successively lower-order H inputs, inverting them when necessary, until the resulting white bar has the position and width you indicated with the grease pencil on the screen. Allowing a small percentage of placement error, you will be able to place a single vertical line of any desired width anywhere on the screen.

This entire process can be summarized as a recipe for building a vertical line or bar, using the NAND/invert circuit on the Tinkerbox assembly.

Recipe for a Vertical Line or Bar

Begin with a white screen, assuming it is actually a full-screen vertical bar.

1. Is that big bar to be narrowed in from the left or right?
   If from the left, use 256H
   If from the right, use $\overline{256H}$
2. Is the resulting white bar to be further narrowed from the left or right?
   If from the left, use 128H
   If from the right, use $\overline{128H}$

3. Is the resulting white bar to be further narrowed from the left or right?
   If from the left, use 64H
   If from the right, use $\overline{64H}$

Continue including more H inputs, in decreasing order, until the desired line width and position is achieved. Remember that including an inverted H input narrows the bar from the right-hand side, while including a true (noninverted) H input narrows the bar from the left. Always begin with 256H and work one H input at a time toward 1H, but use as few inputs as possible to simplify the final circuit design.

After mastering the preceding technique for placing a vertical bar or line at any desired position on the screen, building horizontal bars will seem quite simple. Use the same combination of an 8-input NAND gate followed by an inverter, but apply vertical-count signals to the input of the NAND gate.

Begin the process with 128V. (256V isn't very useful because its first white-to-black alternation takes place only four lines before vertical blanking begins at the bottom of the screen.) If the desired horizontal line is to appear in the white portion of 128V, you are ready for the next step. But if the line is to be in the black region of 128V, invert that signal before applying it to the NAND gate.

Continue adding inverted or noninverted V-count inputs, in decreasing order, until the resulting white bar has the desired position and width. Note that adding a noninverted V-count input narrows the bar from the top, and adding an inverted version of the same signal narrows the bar from the bottom.

To check your understanding of the horizontal-bar-building procedure, make a grease-pencil mark on the screen where you would like a horizontal bar to appear. Then begin adding inverted or noninverted V-count inputs to the NAND gate, beginning with 128V and working downward through the lower-order V-counts signals.

The process can be summarized in a recipe for building a horizontal line or bar.

Recipe for a Horizontal Line or Bar

Begin with a white screen, assuming it is actually a full-screen horizontal bar.

1. Is that big bar to be narrowed from the bottom or top?
   If from the top, use 128V
   If from the bottom, use $\overline{128V}$
2. Is the resulting white bar to be further narrowed from the top or bottom?

If from the top, use 64V
If from the bottom, use $\overline{64V}$

3. Is the resulting white bar to be further narrowed from the top or bottom?
If from the top, use 32V
If from the bottom, use $\overline{64V}$

Continue using more V inputs, in decreasing sequence, until the desired horizontal line width and position is achieved. Bear in mind that including another inverted V input narrows the white bar from the bottom, while including another noninverted V input narrows the bar from the top.

While experimenting with this general procedure for building vertical or horizontal lines, you might have noticed what happens whenever you skip over one of the lower-order signals. Try it. You will find that skipping one input in the normal sequence of high-to-lower-order inputs causes a pair of parallel lines to appear on the screen. Take note of this fact because it might be helpful to you later on.

BUILDING WIDELY SEPARATED PARALLEL LINES AND BARS

Single white vertical or horizontal lines might be useful for certain game designs, but it is more often desirable to create a pair of widely separated parallel lines. A case in point concerns building the border lines for many playing-field type games. Such a border can be built from a combination of widely separated horizontal and vertical lines.

Building a four-sided border figure is getting ahead of the discussion, however. You must first learn how to put a pair of parallel lines of any desired width and spacing you choose on the screen.

This procedure involves three basic steps. First, use the NAND/invert scheme on the Tinkerbox to create one of the two lines. Note the required inputs, then use the same circuit to create the second line. Finally, reduce the two circuits to their simplest possible circuit form and combine them in a Tinkerbox ORing circuit. Use the diagrams in Fig. 3-6 as a reference for studying and experimenting with this procedure in greater detail.

Suppose you want to place two parallel vertical lines on the screen. The two lines can have any desired width and relative position.

Begin by building one of the two lines, using the NAND/invert circuit shown in Fig. 3-6a. Do this by using the Recipe for a Vertical Line or Bar described earlier in this chapter.

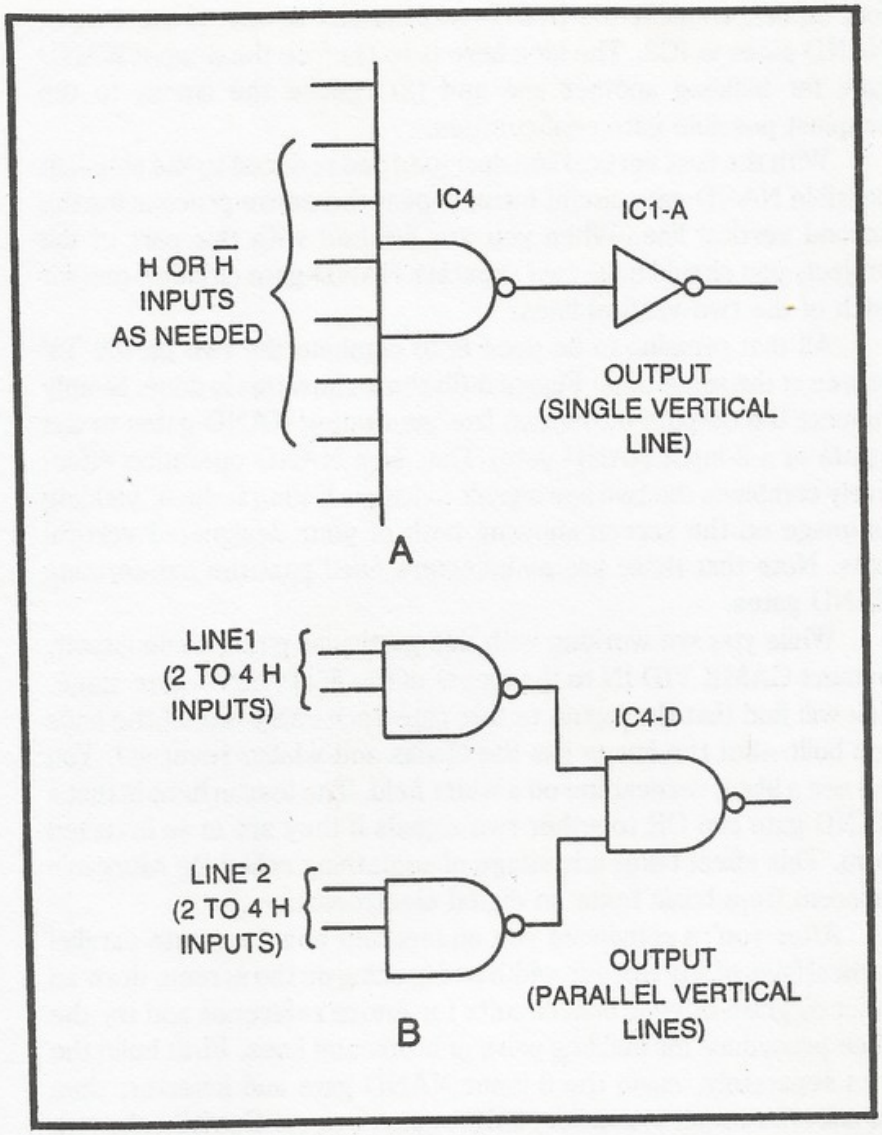


Fig. 3-6. Building white vertical lines and combining them into a single video signal. (a) Circuit for determining the horizontal-count specifications for each line. (b) Combining two simplified versions of the line-generating circuit.

Once you have formed this line, take careful note of the exact H-count inputs you used, and show whether each of them should be inverted or noninverted.

Unless you are trying to build an extremely narrow line, you won't really need more than four combinations of inverted and noninverted inputs to the NAND gate. If it turns out that the line requires only two inputs, transfer the connections to the 8-input NAND gate over to one of the 2-input NAND gates in IC4. (Refer to Fig. 3-2 for the appropriate pin numbers.) If the line calls for three or

four inputs, transfer the NAND connections to one of the 4-input NAND gates in IC2. The idea here is to (1) free the 8-input NAND gate for building another line and (2) reduce the circuit to the simplest possible gate configuration.

With the first vertical line designed and reduced to the simplest possible NAND gate circuit form, repeat the entire process for the second vertical line. When you are finished with this part of the project, you should have two separate NAND-gate circuits, one for each of the two vertical lines.

All that remains to be done is to combine the two on the TV screen at the same time. Figure 3-6b shows how this is done. Simply connect the outputs of the two line-generating NAND gates to the inputs of a 2-input NAND gate. That final NAND operation effectively combines the two line signals in a logic ORing fashion, yielding an image on the screen showing both of your designated vertical lines. Note that there are no inverters used past the line-forming NAND gates.

While you are working with this particular parallel-line circuit, connect GAME VID IN to the inputs of the final NAND-gate stage. You will find that the inputs to this gate each carry one of the lines you built—but the image has the blacks and whites reversed. You will see a black vertical line on a white field. The lesson here is that a NAND gate can OR together two signals if they are in an inverted form. This effect takes advantage of something called De Morgan's theorem from basic texts on digital electronics.

After you're convinced you understand how to create parallel vertical lines of any relative width and spacing on the screen, draw an exact diagram of your final circuits for future reference and try the same procedure for building pairs of horizontal lines. First build the lines separately, using the 8-input NAND gate and inverter, then transfer the circuit to smaller NAND-gate devices. Combine the two lines as shown in Fig. 3-6b. Refer to the Recipe for a Horizontal Line or Bar as necessary.

Building Intersecting Horizontal and Vertical Lines

The Line/Bar Tinkerbox assembly can be used for creating intersecting horizontal and vertical lines. Suppose, for example, the experimenter wants to build a tic-tac-toe pattern on the screen. This is essentially a pair of vertical and a pair of horizontal parallel lines combined into one picture. And if the lines are fixed at the extreme edges of the screen, it appears they are creating a border for many different kinds of TV games.

Creating intersecting parallel lines, both horizontal and vertical, is a simple matter of extending the Tinkerbox techniques already described in this chapter. First build the lines one at a time, using the 8-input NAND gate and inverter system. Keep track of the "formula" for each line, then reduce them to simpler NAND-gate inputs.

Next combine the vertical lines as shown in Fig. 3-6, and then combine the two horizontal lines in the same fashion. All that remains to be done at that point is to combine the two sets of parallel lines into one image. Figure 3-7 shows how this can be done.

Figure 3-7a shows the most straightforward technique for combining pairs of horizontal and vertical parallel lines. The idea is to build the horizontal and vertical parallel lines separately as described in the previous section of this chapter. Then combine the two sets of lines by first inverting them and then applying them to separate inputs of a 2-input NAND gate. The output of that final NAND-gate stage yields the composite image.

While this might be the most straightforward technique, it is not the most efficient. It is possible to do the same job using the circuit in Fig. 3-7b.

To understand how the circuit in Fig. 3-7b works, let's assume you have used the recipes for vertical and horizontal lines to get the line-generating specifications—the combinations of inverted and noninverted count inputs for each of the four lines. After reducing these input specifications to the point where you are using the simplest possible input NAND gates for each line, simply connect the outputs of the four NAND gates to a 4-input NAND gate. You will find that the output of that final NAND-gate circuit creates an image identical to the output of the more complicated looking circuit in Fig. 3-7a.

In fact it is possible to use the general circuit in Fig. 3-7b to combine any number of different lines on the screen. Simply build the horizontal and vertical lines separately, reduce the input NAND gate circuits to their simplest form, and then connect the outputs of each of the line-generating NAND gates to an output NAND gate. If the output NAND gate happens to be a 7430 8-input NAND gate, it is possible to combine as many as eight different combinations of horizontal and vertical lines into one image.

The circuits in Fig. 3-8 show the author's circuits for generating the tic-tac-toe pattern and a full-screen border for many different kinds of video games. The circuits are essentially identical, the only difference being the spacing of the pairs of horizontal and vertical parallel lines. Of course it is possible to create the same patterns, but

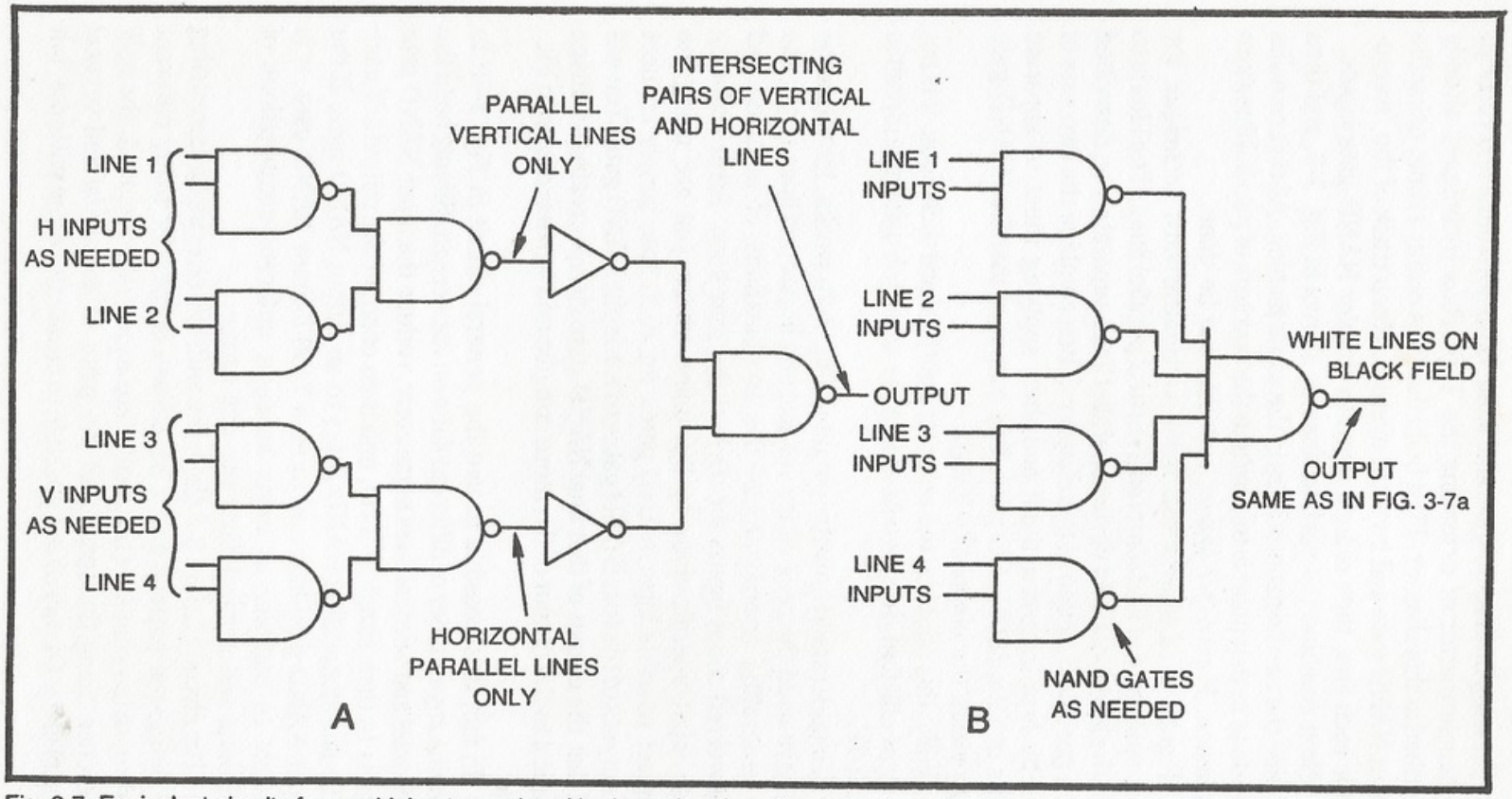


Fig. 3-7. Equivalent circuits for combining two pairs of horizontal and vertical bars.

with black lines on a white background, by inverting the composite signal before applying it to the GAME VID IN terminal of the Sourcebox unit.

Figure 3-7b is a key circuit in designing all kinds of video games and effects, so it should be clearly marked for future reference.

Building Narrow Lines More Efficiently

One of the features of the line-building procedure outlined thus far in this chapter is that the width of the line depends on the number of horizontal- or vertical-count inputs used. The narrower the line is supposed to be, the more inputs one must use.

Now this calls for using a lot of inputs to make very narrow lines. It would be nice if there were some way to modify the technique to reduce the number of count inputs required for making narrow lines. Fortunately, there is such a technique, and it quite often reduces the number of inputs to just two or three.

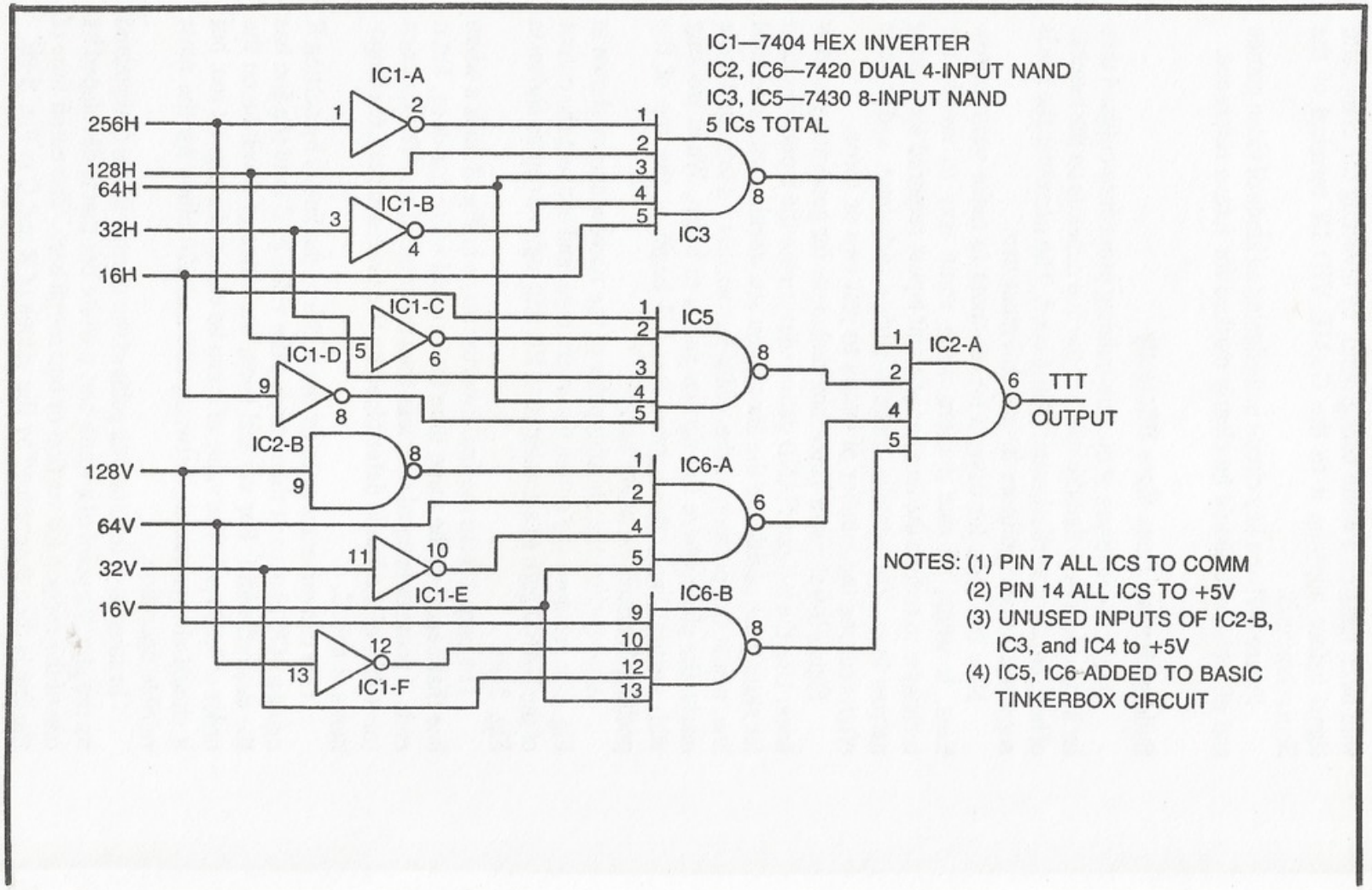
Figure 3-9 shows a Tinkerbox technique for generating narrow lines. Use the 8-input NAND gate to determine the inputs required for setting the position of the line. If you are working with a vertical line, adjust the position of the white bar so that its left-hand edge marks the place where the narrow line is to begin. When working with vertical lines, the narrow line will begin at the top of the position-determining white bar.

After setting the starting point of the narrow line as shown in Fig. 3-9a, remove the output inverter stage, and connect the output of the NAND gate to a combination RC and logic circuit as shown in Fig. 3-9b.

The output from the final inverter stage in Fig. 3-9b is a white line that begins at the same time the original white bar does, but it ends at some time equal to or less than the original bar. The duration (or width) of the line is determined by the time constant of components R and C.

One convenient way to fix the width of the line is by setting R equal to 470 ohms and then varying the value of C until the line has the desired width. For vertical lines, the value of C will be on the order of 0.002 μF. The value of R can be changed a little bit too, but it should always remain between 100 and 470 ohms for the most reliable operation.

In summary, the starting point of the narrow line is determined by the starting point of a white bar, a white bar that is developed by one of the recipes for vertical or horizontal bars. The cutoff time of the line is then determined by the values of R and C in Fig. 3-9b.


IC1—7404 HEX INVERTER
IC2, IC6—7420 DUAL 4-INPUT NAND
IC3, IC5—7430 8-INPUT NAND
5 ICs TOTAL
IC1-A
IC1-B
IC1-C
IC1-D
IC1-E
IC1-F
IC2-A
IC2-B
IC3
IC5
IC6-A
IC6-B
256H
128H
64H
32H
16H
128V
64V
32V
16V
TTT
OUTPUT
NOTES: (1) PIN 7 ALL ICS TO COMM
(2) PIN 14 ALL ICS TO +5V
(3) UNUSED INPUTS OF IC2-B, IC3, and IC4 to +5V
(4) IC5, IC6 ADDED TO BASIC TINKERBOX CIRCUIT

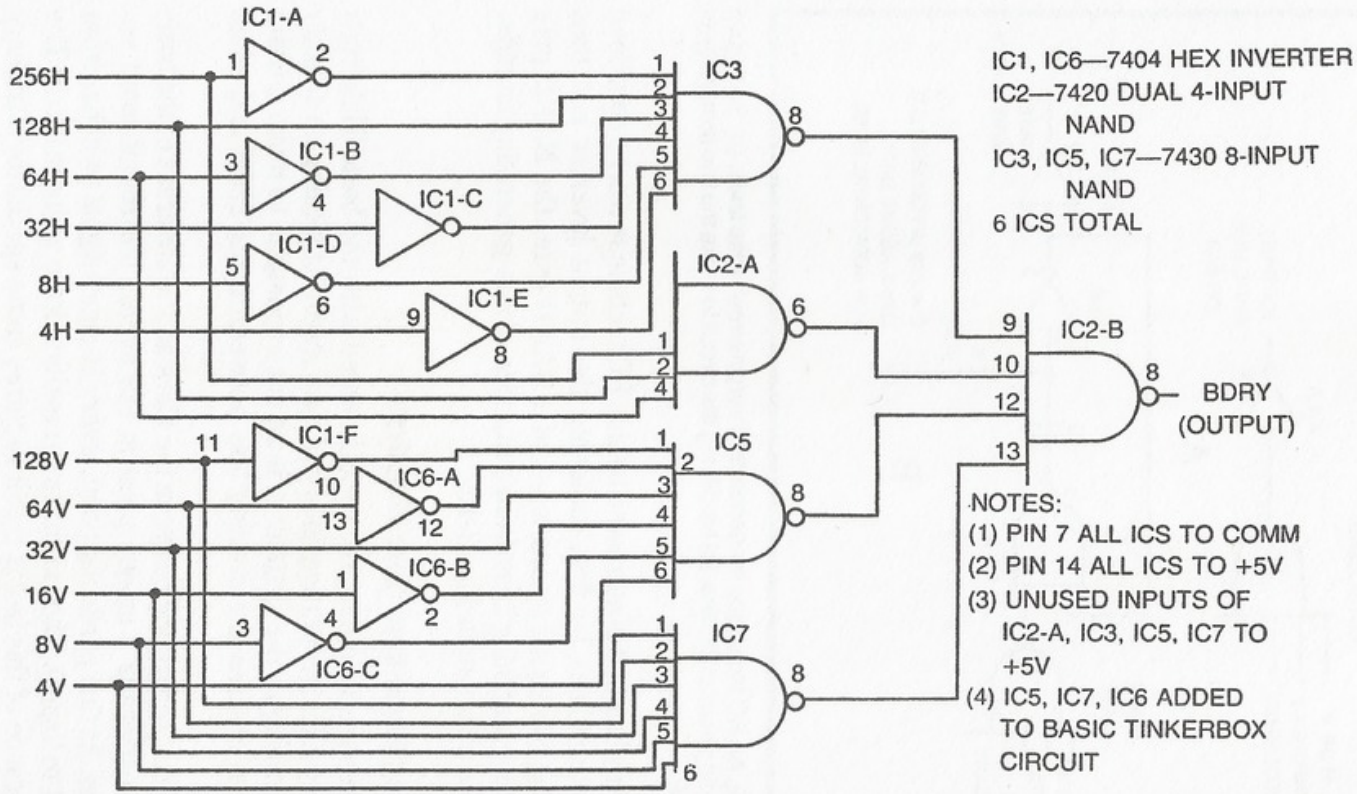


Fig. 3-8. Some applications of circuits combining pairs of parallel lines. (a) A tic-tac-toe image. (b) A border pattern for many TV games.

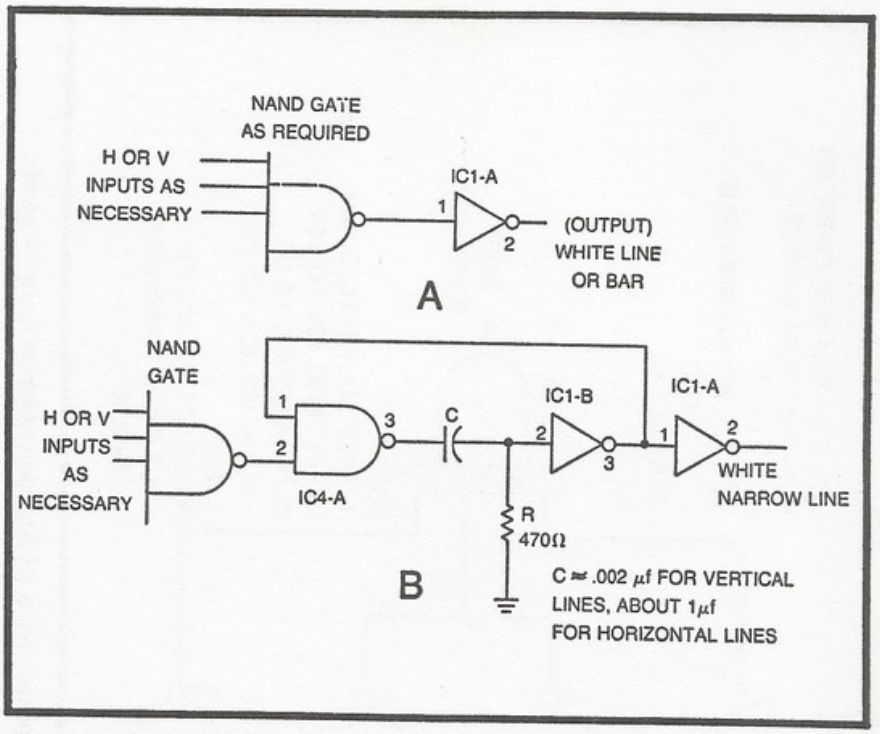


Fig. 3-9. An RC technique for generating very narrow white lines. (a) Setting up the starting point of the line. (b) Modifying the circuit to make the line as narrow as desired.

Any line that is narrowed by this RC technique can be combined with other lines by first removing the output inverter and then applying the signal to one input of a NAND gate, the NAND gate used for combining lines from a number of line-generating circuits. See the example in Fig. 3-10.

Building Broad Bars More Effectively

Anyone who has now experimented with the basic Tinkerbox technique for generating lines and bars might be picking up another shortcoming to the system: it is all but impossible to make a white bar that crosses the dividing line between black 256H and white 256H.

The dividing line between the black and white areas of the basic 256H pattern is a unique point in the overall horizontal-count sequence. At that particular point, every H-count signal switches from logic 1 to logic 0. They all make a transition from white to black. The significance of this fact is that it requires some special logic trickery to make a wide white bar extend continuously through this unique dividing line; and that particular bit of trickery hasn't been fully described yet.

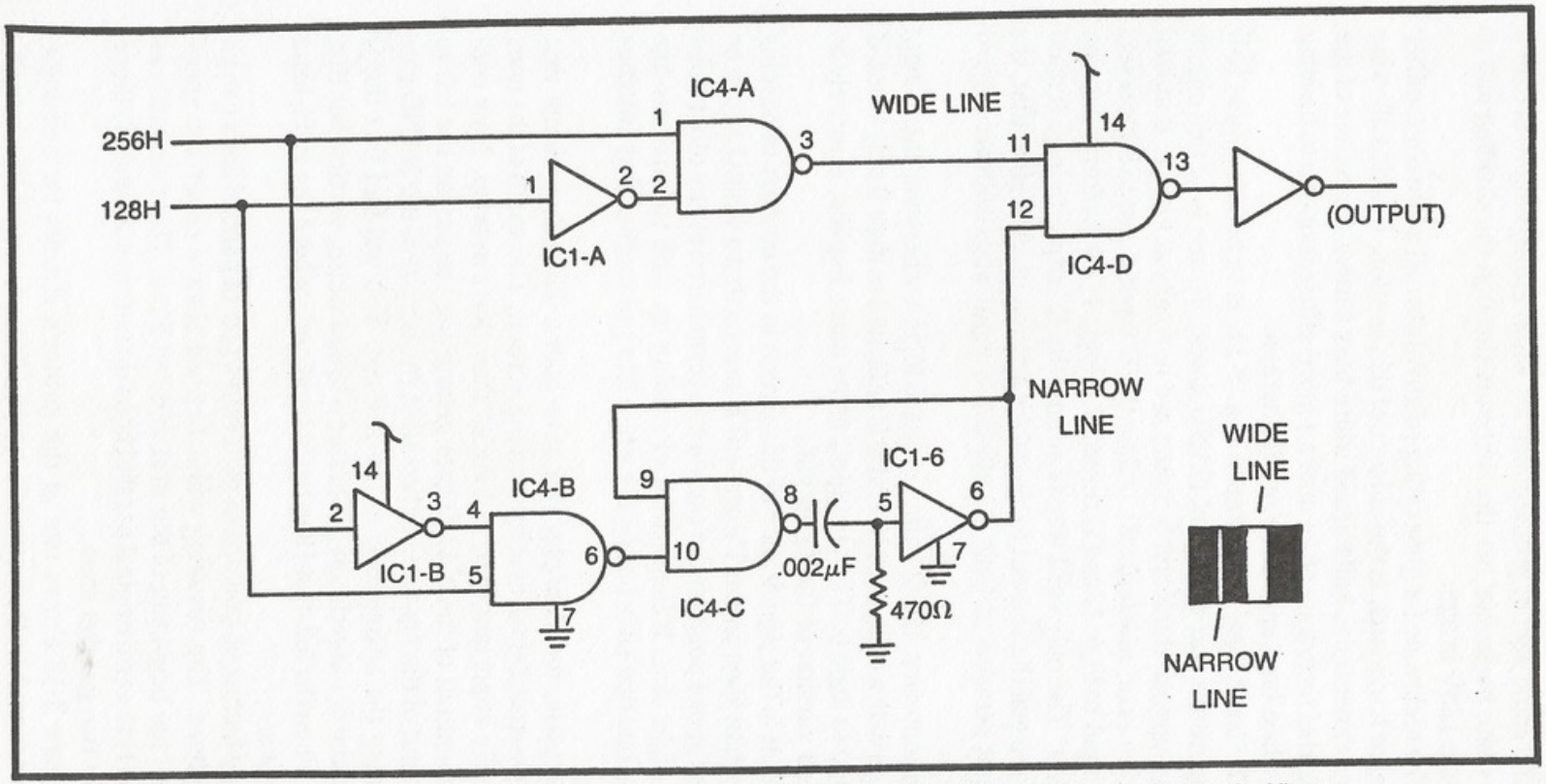


Fig. 3-10. Using the basic line-generating circuit and the RC narrow-line scheme together. The result is a narrow vertical line near the left-hand side of the screen and a moderately wide vertical bar just right of center.

The same sort of problem occurs when attempting to create a broad white horizontal bar that extends through the dividing line of the basic 128V image.

The technique for generating relatively broad white bars, either vertical or horizontal, is the subject of this section. Besides allowing the experimenter to build broad white bars across the center of the screen, this technique often offers a more efficient option to building broad white bars anywhere on the screen.

The basic idea, illustrated in Fig. 3-11, is to build a simple $\overline{R}$-$\overline{S}$ flip-flop from a pair of 2-input NAND gates. To see how this circuit works, suppose the $\overline{R}$ and $\overline{S}$ inputs are normally at logic 1. If indeed this is the case, momentarily pulling the $\overline{S}$ input down to logic 0 sets the Q input to logic 1, and Q remains at logic 1, even after $\overline{S}$ returns to logic 1. The only valid way to return the Q output to logic 0 once again is to pull the $\overline{R}$ input to logic 0 for a moment. And after that, the Q output remains at logic 0 until the $\overline{S}$ input sees another logic-0 pulse.

In summary, the Q output is set to logic 1 whenever the $\overline{S}$ input is pulsed with a logic-0 level, and Q is returned to logic 0 only when $\overline{R}$ is pulsed to logic 0. The $\overline{Q}$ output, as its name implies, is merely an inverted version of the Q output.

What is the significance of this circuit in the context of building broad white lines on the TV screen? It means that a white bar can be initiated at one point on the screen and terminated at any other point to the right of it. It is a matter of coming up with two narrow-line signals, one that switches on the white line and another that switches it off.

Suppose, for example, you have built a Tinkerbox circuit that draws two parallel vertical lines on the screen. The width of the lines isn't really important, just as long as they don't overlap. Now connect the output of the NAND gate generating the line on the left to the $\overline{S}$ input of the flip-flop, and connect the output of the NAND gate generating the second line to the $\overline{R}$ input. You will find that the Q input of the flip-flop circuit generates a broad white, vertical bar that begins when the left-hand line begins and ends where the right-hand line begins.

Any technique you use to generate a pair of parallel lines can be applied here. The resulting white bar will always cover the space between the beginning of the first and last lines. This rule holds as long as (1) the two original parallel lines do not overlap and (2) there are only two parallel lines.

Figure 3-12 shows one of the author's circuits for building a white vertical bar that extends across the center of the screen.

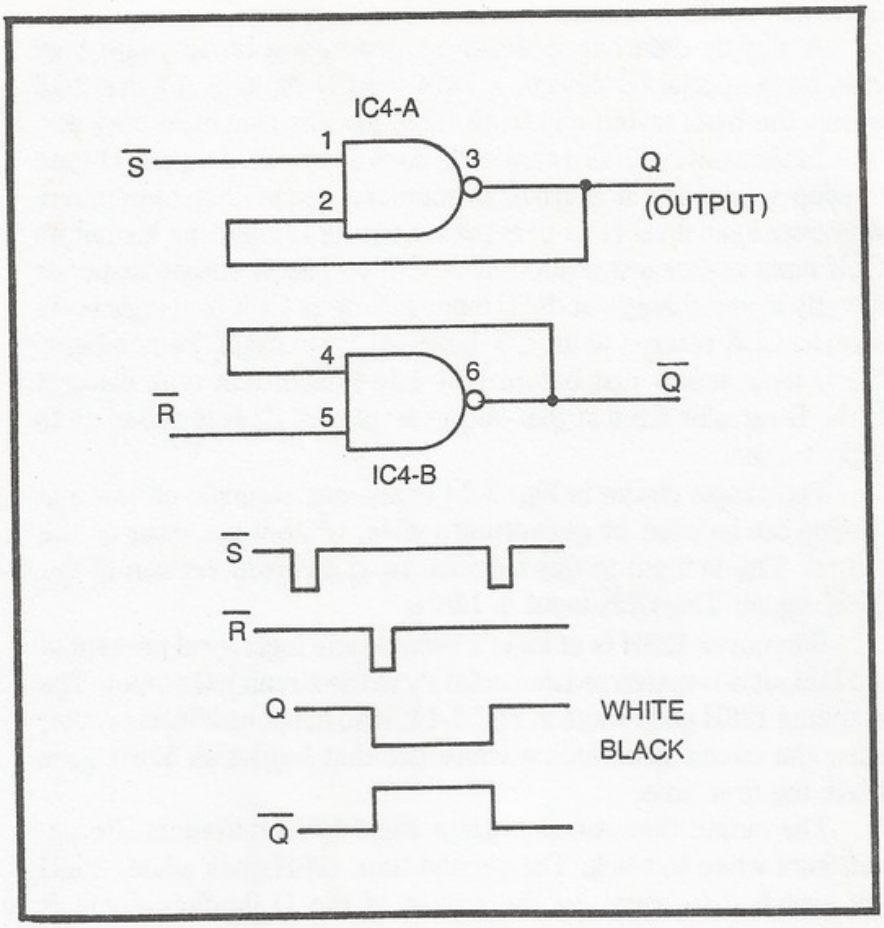


Fig. 3-11. A basic R-S flip-flop circuit and relevant waveforms.

IC2-A defines the starting point of the bar. Viewed on the TV screen, the output of this NAND gate appears as a relatively narrow black bar just left of the center of the screen. IC2-A thus defines the starting point of the white bar being generated in this example.

IC2-B in Fig. 3-12 generates the end-of-bar signal. Looking at the output of this NAND gate on the screen, it appears as a relatively narrow black bar just right of the center of the screen. Combining these two black-bar signals in the $\overline{R}$-$\overline{S}$ flip-flop ICs 4-A and 4-B, yields the final result.

The same $\overline{R}$-$\overline{S}$ technique can be applied to the task of generating broad vertical bars. The only real difference is that the inputs to the first set of NAND gates come from vertical-count sources rather than horizontal-count sources.

Try building some broad white bars, both horizontal and vertical ones, until you are confident you can handle the $\overline{R}$-$\overline{S}$ flip-flop procedure.

A slightly different approach to generating broad white bars calls for a special IC device, a 7474 dual D flip-flop. Figure 3-13 shows the basic layout and truth table for this nice little device.

In the context of drawing wide bars on the screen, the D-type flip-flop works like a short-term memory circuit that remembers whatever logic level (1 or 0) is present at its D input the instant its CLK input makes a transition from 1 to 0. The Q output responds directly to any changes at the D input as long as CLK is at logic 0. As soon as CLK returns to logic 0, however, the circuit "remembers" the D input it saw just before that 1-to-0 transition took place at CLK. D remains fixed at that output level until CLK is pulled up to logic 1 again.

The simple circuit in Fig. 3-14 is just one example of how a D flip-flop can be used for generating a wide, white vertical bar on the screen. The D input in this instance is an inverted version of the 256H signal. The CLK input is 128H.

Whenever 128H is at logic 1 (white) any logic level present at the D input is transferrred immediately to the circuit's Q output. The first time 128H goes white in Fig. 3-14, it so happens 256H is white; thus, the circuit generates a white bar that begins as 128H goes white the first time.

The output then remains white while 128H makes an alternation from white to black. The second time 128H goes white, 256H has switched to olack, so the output of the D flip-flop circuit is switched to logic 0 (black on the screen).

It is left to the experimenter to decide whether the $\overline{R}$-$\overline{S}$ or D-type flip-flop is best under specific conditions. Neither is better under all circumstances.

Building Broad Bars and Multiple Parallel Lines by Foldover

There is yet another technique for building bars and parallel lines on the screen. This technique still uses the basic Tinkerbox approach to getting things started, but it calls for an additional IC to complete the job.

Understand from the outset that this foldover technique is useful only under certain circumstances described later in this section. Usually it is simpler and more effective to use a straight Tinkerbox, RC-modified, or flip-flop-modified technique. The foldover procedure is presented here for two reasons: first, it completes the list of possible ways to generate bars and lines; and second, it introduces a digital principle that will become especially

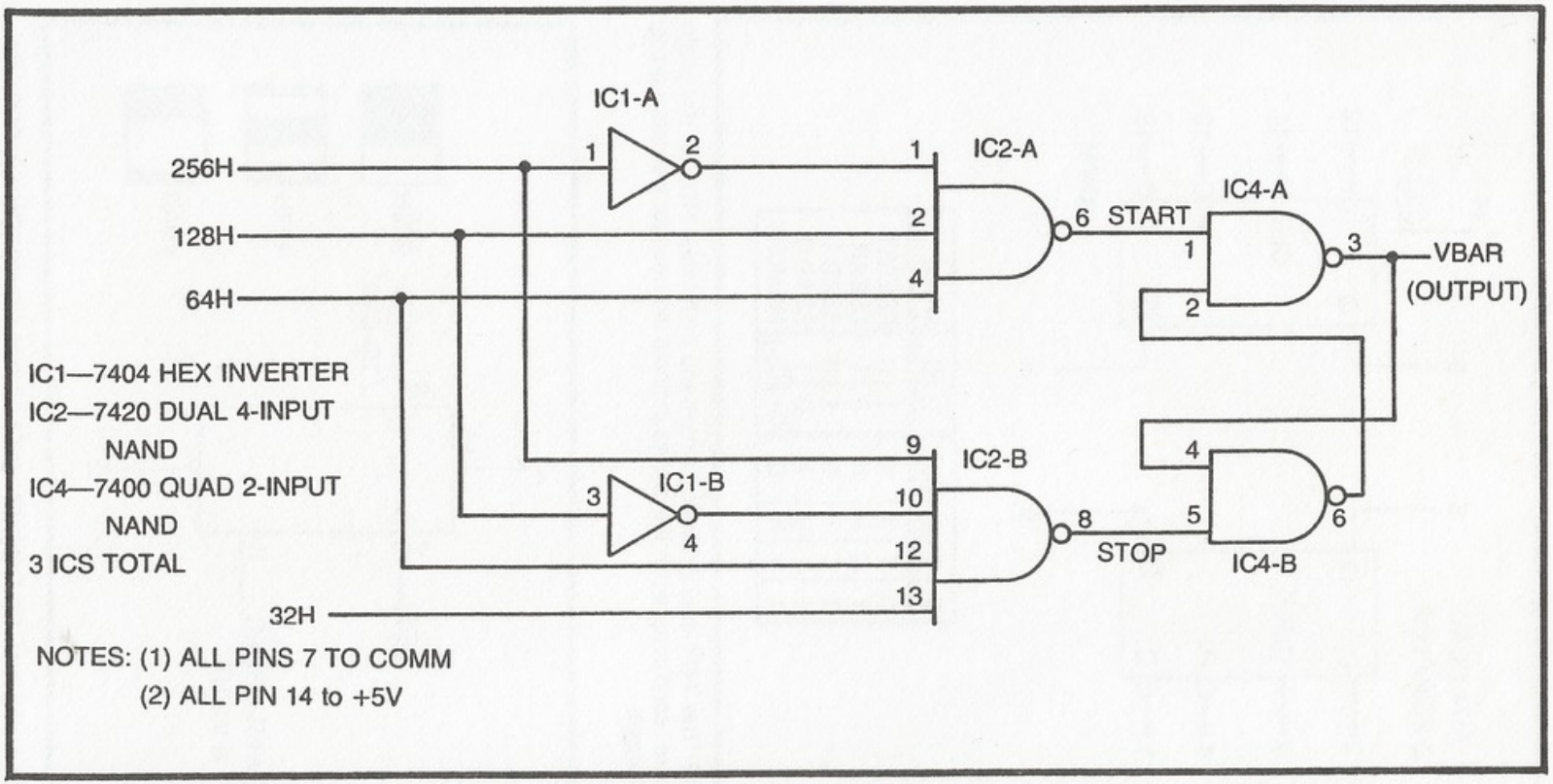
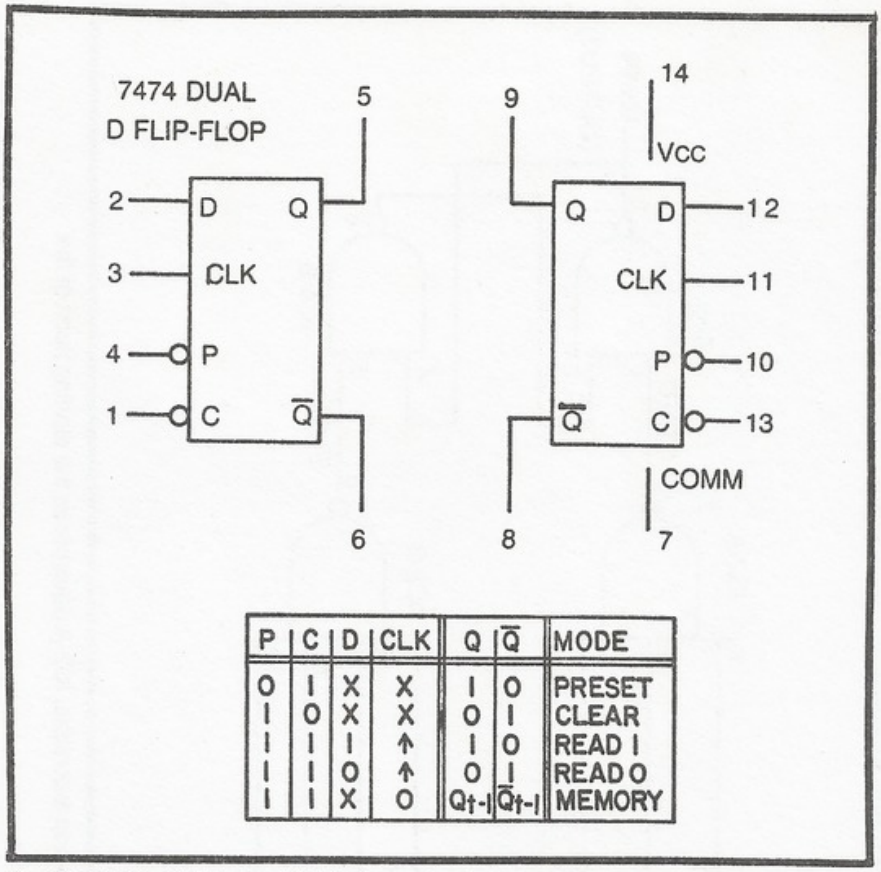


Fig. 3-12. Generating wide, center-crossing bars using the R-S flip-flop technique. IC2-A determines the starting point of the bar and IC2-B fixes its end.



P	C	D	CLK	Q	$\overline{Q}$	MODE
0	1	X	X	1	0	PRESET
1	0	X	X	0	1	CLEAR
1	1	1	↑	1	0	READ 1
1	1	0	↑	0	1	READ 0
1	1	X	0	Q_{t-1}	$\overline{Q}_{t-1}$	MEMORY

Fig. 3-13. The 7474 dual D flip-flop circuit and truth table. (The Xs designate "don't care" conditions, while the arrows indicate the need for a positive-going level change).

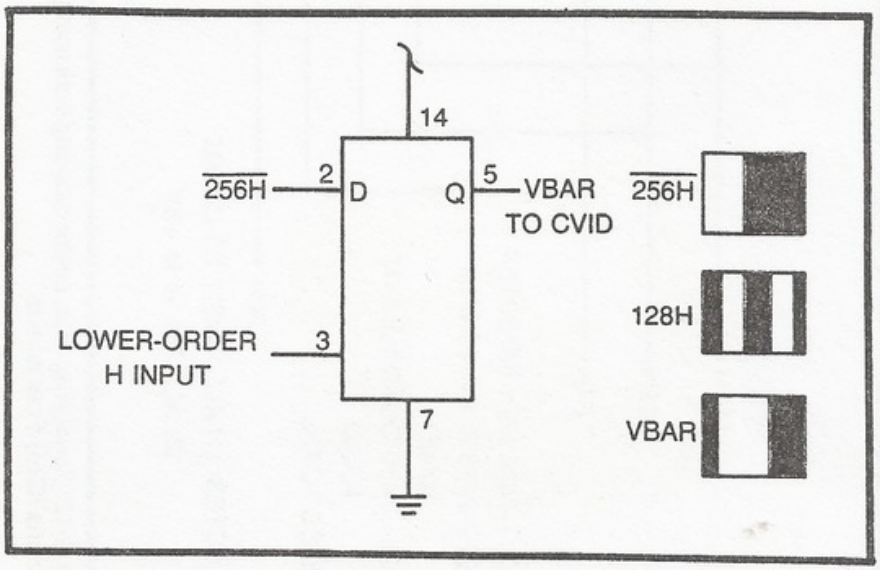


Fig. 3-14. Using the D flip-flop to generate a board, center-crossing vertical bar.

important when attempting to build more-complex figures such as racing cars, people, airplanes, rockets, tanks, and so on.

To see how the foldover procedure works, suppose you have built a vertical white line using $\overline{256H}$, 128H, $\overline{64H}$, and $\overline{32H}$. As shown in Fig. 3-15a, these inputs occupy all four inputs of a 4-input NAND gate on the Tinkerbox assembly. If the output of this gate is run through an inverter before it is applied to GAME VID IN, it generates a moderately narrow white line that is just a bit left of center. See Fig. 3-15b.

Now modify the inputs to the line-generating NAND gate by running them through a set of EXCLUSIVE-OR gates (a 7486 quad 2-input EXCLUSIVE-OR). As long as the control input to the 7486

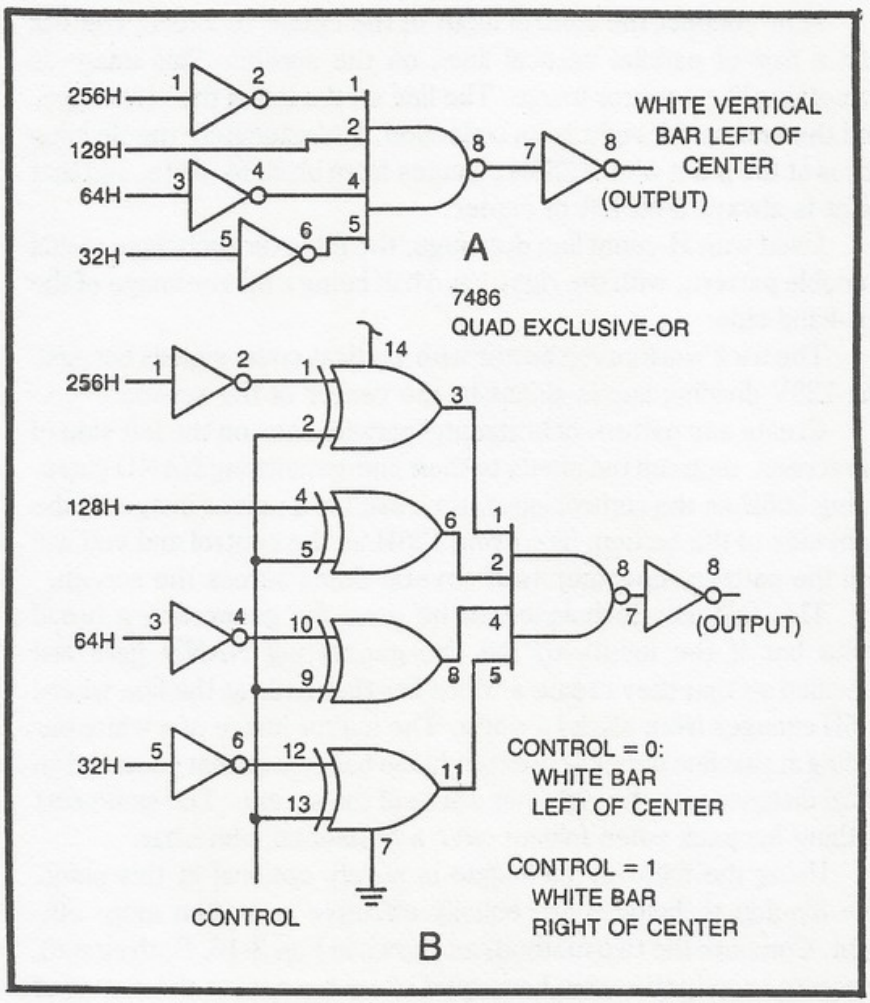


Fig. 3-15. The basic foldover bar-drawing technique. (a) Circuit for establishing the size and position of the left-hand half of the image. (b) Final circuit for doubling or folding over the original image.

IC is connected to COMM (at logic 0) the white line appears at the same place on the screen it had before the EXCLUSIVE-OR gates were installed. But connecting that control line to +5V (logic 1), the line shifts to the right of center.

What is happening here is a reversal of the horizontal-count sequence as the NAND gate sees it. Reversing or inverting the outputs from any digital counter circuit makes it appear to count backwards. So when the control input to the EXCLUSIVE-OR gates is a logic 0, the NAND gate sees the H-count inputs arriving in their normal up-coming sequence. Setting the control input to logic 1, however, creates the effect of a counter that is running backwards. In a sense, the NAND gate is fooled into reacting as though the horizontal count is running from right to left across the screen.

Now connect the control input of the circuit to 256H. You will find a pair of parallel vertical lines on the screen. This image is something like a mirror image. The line on the left is the real image, and the one on the right is its reflection. Unfortunately the dividing line is at the point where 256H changes from black to white, and that point is always a bit left of center.

Used with H-count line drawings, the foldover technique yields a double pattern, with the right-hand half being a mirror image of the left-hand side.

The trick works even better with vertical-count signals because the 128V dividing line is closer to the center of the screen.

Create any pattern of horizontal parallel lines on the left side of the screen, then run the inputs to their line-generating NAND gates. Using 256H as the control input, the result is a mirror image on the right side of the screen. Try using 128H as the control and you will find the pattern repeating itself several times across the screen.

This foldover technique can be used for generating a broad white bar if the inputs to the line-generating NAND gate are specified so that they create a white bar that ends at the line where 256H changes from black to white. The mirror image of a white bar ending at that line effectively extends the bar across that point and an equal distance into the right-hand side of the screen. The same sort of thing happens when folding over a horizontal white bar.

Using the foldover technique is purely optional at this point. The flip-flop techniques are equally effective and often more efficient. Compare the two methods as shown in Fig. 3-16. Both circuits generate exactly the same horizontal white bar across the center of the screen. The scheme in Fig. 3-16a uses a $\overline{R}$-$\overline{S}$ flip-flop, while the one in Fig. 3-16b uses the foldover technique.

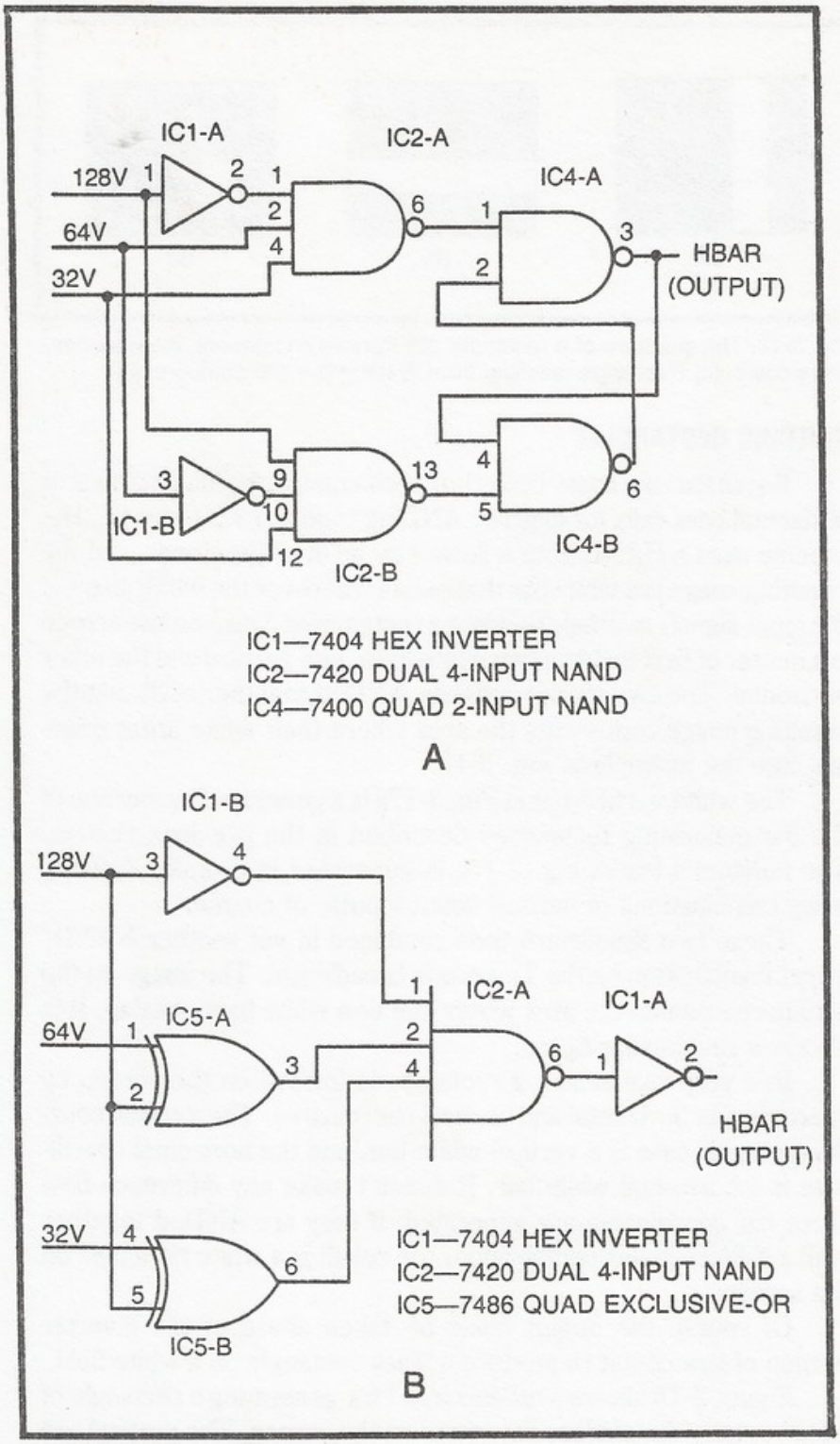


Fig. 3-16. Equivalent circuits for generating a wide, center-crossing white bar. (a) Flip-flop version. (b) Foldover version.

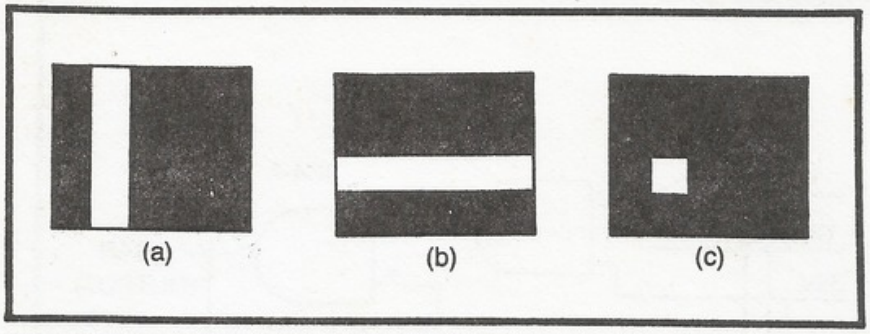


Fig. 3-17. The structure of a rectangle. (a) Vertical component. (b) Horizontal component. (c) Rectangle resulting from ANDing the two components.

BUILDING RECTANGLES

Recall that the basic Tinkerbox technique for building vertical or horizontal bars calls for logically ANDing together white areas. The scheme uses a NAND gate followed by an inverter circuit, and the resulting image is a white bar that exists wherever the white areas of the input signals overlap. Building a rectangular figure on the screen is a matter of first building two white bars, one vertical and the other horizontal. The two signals are then ANDed together such that the resulting image represents the area where their white areas overlap. See the example in Fig. 3-17.

The white vertical bar in Fig. 3-17a is a generated by any one of the bar-generating techniques described in the previous section. The horizontal bar in Fig. 3-17a is generated in a similar fashion, using combinations of vertical-count inputs, of course.

These two signals are then combined in yet another NAND/invert combination on the Tinkerbox breadboard. The image on the screen represents the area where the two white bars overlap. It is always a rectangular figure.

In a very real sense, a rectangle is formed on the screen by specifying its horizontal and vertical coordinates. The vertical coordinate in this case is a vertical white bar, and the horizontal coordinate is a horizontal white bar. It doesn't make any difference how these bar coordinates are generated; if they are ANDed together with a NAND/invert combination, the result is a white rectangle on the screen.

Of course the output could be taken ahead of the inverter portion of this circuit to produce a black rectangle on a white field.

Figure 3-18 shows a model circuit for generating a rectangle of just about any dimensions anywhere on the screen. The vertical and horizontal coordinates are generated by a straightforward NAND-gate procedure, using the Tinkerbox components. The outputs of

these line-generating NAND gates are inverted, then applied to a 2-input NAND gate. If the rectangle is to be white on a black field, the signal from the 2-input NAND gate is run through yet another inverter before applying it to GAME VID IN.

One good way to build a rectangle is to sketch its outline on the TV screen with a grease pencil. Then work with the Tinkerbox circuit to build a white vertical bar that runs down through the rectangle drawing, fitting the position and width as closely as possible. That operation specifies the rectangle's vertical component. Write down the H-input specifications and repeat the operation, generating a horizontal bar that fits the rectangle's specified position and height. Again, mark down the V-input specifications.

Reduce the two circuits to their simplest NAND-gate forms and feed their outputs to a 2-input NAND gate and inverter as shown in Fig. 3-18. If you've specified the vertical and horizontal inputs properly, there's your rectangle on the screen.

This rectangle-building procedure can be summarized as follows.

Recipe for Building a Rectangle or Square

1. Build a horizontal white line or bar as described in Recipe for a Horizontal Line or Bar. The position of that bar

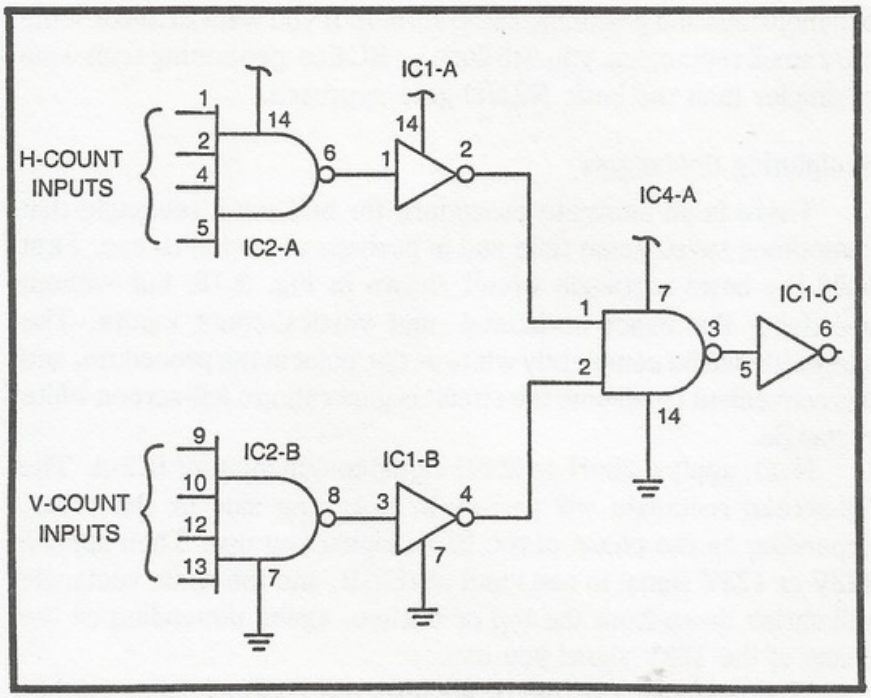


Fig. 3-18. A basic circuit for building rectangles.

determines the horizontal position of the rectangle or square, and its horizontal width determines its final width.

2. Build a vertical white line or bar as described in Recipe for a Vertical Line or Bar. The position of this bar or line determines the vertical position of the rectangle, and its horizontal height fixes the height of the final product.
3. If the 8-input NAND gate is used for building these two lines or bars, transfer them to simpler NAND gates. A line or bar requiring four inputs, for example, can be transferred to one section of the dual 4-input NAND gate.
4. Run the outputs of the two sets of NAND gates through INVERT gates and then to the inputs of a 2-input NAND gate.
5. Invert the output of that 2-input NAND gate and apply the resultant signal to CVID—and presto! There's the square or rectangle. This step, incidentally, is responsible for creating a white square on a black field. The situation can be easily reversed (same square, but black on a white field) by omitting the final inverting operation—take the CVID from the ouput of the 2-input NAND gate described in step 4.

Try building some rectangles of your own, specifying a variety of dimensions and positions on the screen. If you want to make some very small rectangles, you will find the RC line-generating technique is simpler than the basic NAND-gate approach.

Sculpturing Rectangles

There is an alternate procedure for building a rectangle that sometimes saves some time and is perhaps more fun to use. First build the basic rectangle circuit shown in Fig. 3-18, but without specifying the exact horizontal- and vertical-count inputs. The screen should be completely white at this point in the procedure, and it is convenient to assume the circuit is generating a full-screen white rectangle.

Next, apply a 256H or 256H signal to one input of IC2-A. The full-screen rectangle will narrow in from one side or the other, depending on the phase of the 256H signal you use. Then apply a 128V or 128V signal to one input of IC2-B, and the white rectangle will shrink down from the top or bottom, again, depending on the phase of the 128V signal you use.

Then apply an inverted or noninverted 128H signal to another input of IC2-A and watch the rectangle narrow even more. Apply a

64V or $\overline{64V}$ signal to a second input of IC2-B to make the rectangle pull down farther from the top or up from the bottom.

Continue adding more H and V inputs to their respective NAND gates until the resulting rectangle has the relative dimensions and position you desire. I've called this a *sculpturing* technique because it gave the experimenter a feeling of sculpturing or trimming a figure on the screen.

The general procedure for sculpturing a rectangle on the screen can be summarized in a basic "recipe."

Recipe for Sculpturing a Rectangle or Square

Begin with a white screen, assuming it is actually a full-screen, white rectangle.

1. Is that big white rectangle to be narrowed in from the left right?
   If from the left, use 256H
   If from the right, use $\overline{256H}$
2. Is the resulting vertical bar to be reduced down from the top or upward from the bottom?
   If from the top, use 128V
   If from the bottom, use $\overline{128V}$
3. Is the resulting white square to be further narrowed from the left or right?
   If from the left, use 128H
   If from the right, use $\overline{128H}$
4. Now is that white rectangle to be reduced downward from the top or upward from the bottom?
   If from the top, use 64V
   If from the bottom, use $\overline{64V}$
5. The white square can be sculptured further by alternately reducing its horizontal and vertical size and position.

Simplifying the Final Rectangle Circuit

All of the rectangle-building circuits described thus far use at least three inverter circuits, one from each of the line-generating NAND gates and one at the output of the NAND gate that combines the two lines. This circuit is quite appropriate when experimenting with various ways to build a desired rectangle, but there happens to be a simpler way to do the same job. This simpler operation calls for using an IC that is not included on the Tinkerbox parts list, but it ought to be specified in any final circuit that is to be part of a permanent game system.

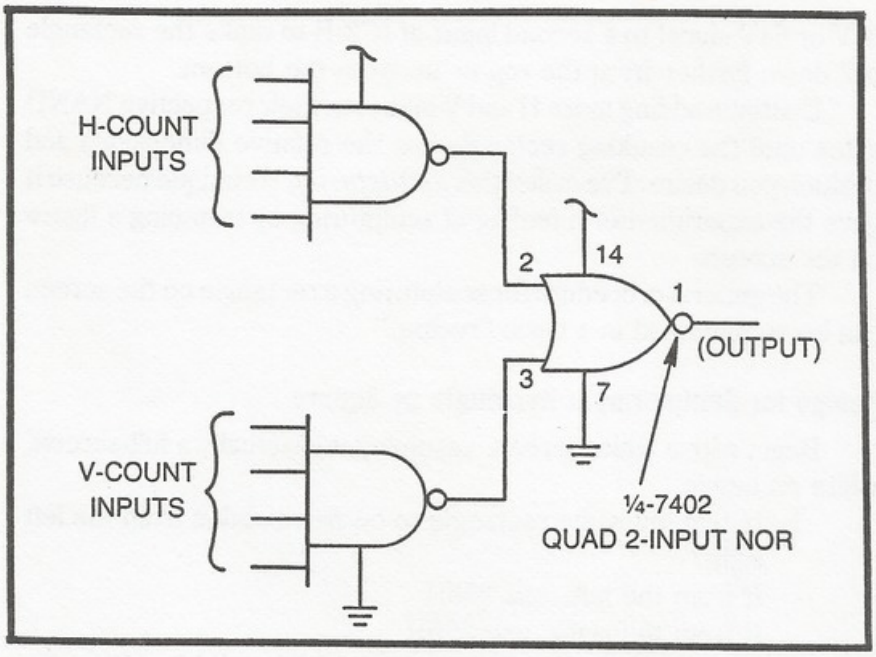


Fig. 3-19. A simplified rectangle-building or sculpturing circuit

Rather than inverting the output of each line-generating circuit, applying the signals to a NAND gate and then inverting the result, simply take the outputs directly from the line-generating NAND gates and apply them to the inputs of a 2-input NOR gate—one section of a 7402 quad 2-input NOR IC package. Compare the basic Tinkerbox rectangle-generating circuit in Fig. 3-18 with the simplified NOR-gate version in Fig. 3-19.

The circuit in Fig. 3-19 takes advantage of one of De Morgan's logic theorems that says two signals can be ANDed together by performing a NOR operation on inverted logic inputs. Well, the signals from the line-generating NAND gates are really inverted versions of the lines they are to generate. (Note that those outputs generate black lines on a white field.) So running them directly to a NOR gate yields an ANDed output; and that's exactly what the rectangle-building operation is based on.

NOR gates are used quite frequently throughout this book where it is necessary to AND together signals that are already in an inverted state. Using that one NOR gate eliminates the need for at least three inverter circuits.

COMBINING ANY NUMBER OF STATIC FIGURES ON THE SCREEN

Single lines, bars, and rectangles—the sort of figures described throughout this chapter—have little value in themselves. Two or

more of these basic static figures must be combined on the screen to make up a more useful and interesting game pattern.

Speaking in digital terms, the process of combining two or more static figures on the screen is a matter of ORing together the individual elements. In terms of Tinkerbox technology, this means running the output of each figure-generating circuit to a separate input of a NAND gate. If the output of that NAND gate is then inverted, the resulting signal is a composite image.

The circuit in Fig. 3-20 shows how four different static figures can be combined into a single, composite video waveform. The procedure takes advantage of De Morgan's theorem, where a NAND gate performs an OR operation if the inputs are in an inverted form.

Most of the line, bar, and rectangle generators in this chapter yield inverted signals (black and white) anyway, so the NAND-gate circuit is the most convenient and efficient one for this particular job. Note that the output of the NAND gate is an "upright" or white-on-black signal.

SOME INTERESTING PATTERNS FROM STATIC-FIGURE COMPONENTS

Figure 3-21 shows a variety of circuits that create some fascinating images on the screen. The basic idea is to combine a certain horizontal-count input with a vertical-count input of the same order, 64H with 64V, for instance.

The circuits built around NAND gates produce a regular pattern of squares on the screen. If the video signal is taken directly from the output of this NAND gate, the result is a set of black squares on a

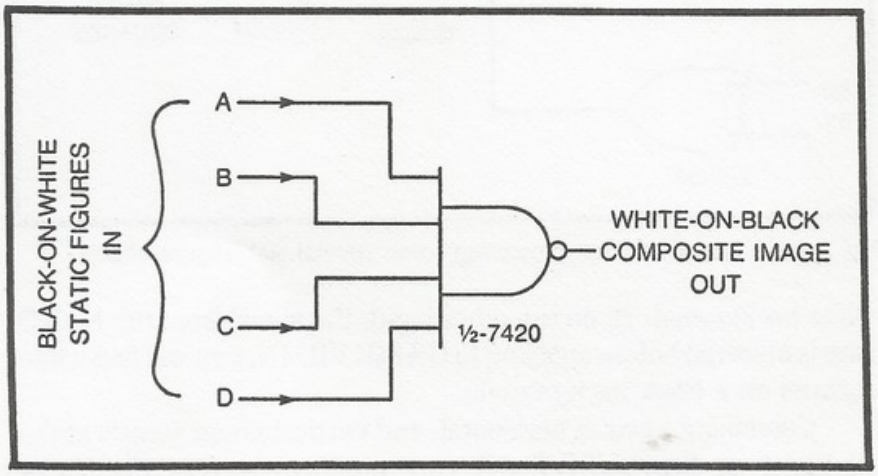


Fig. 3-20. Combining two or more figures into a single video signal

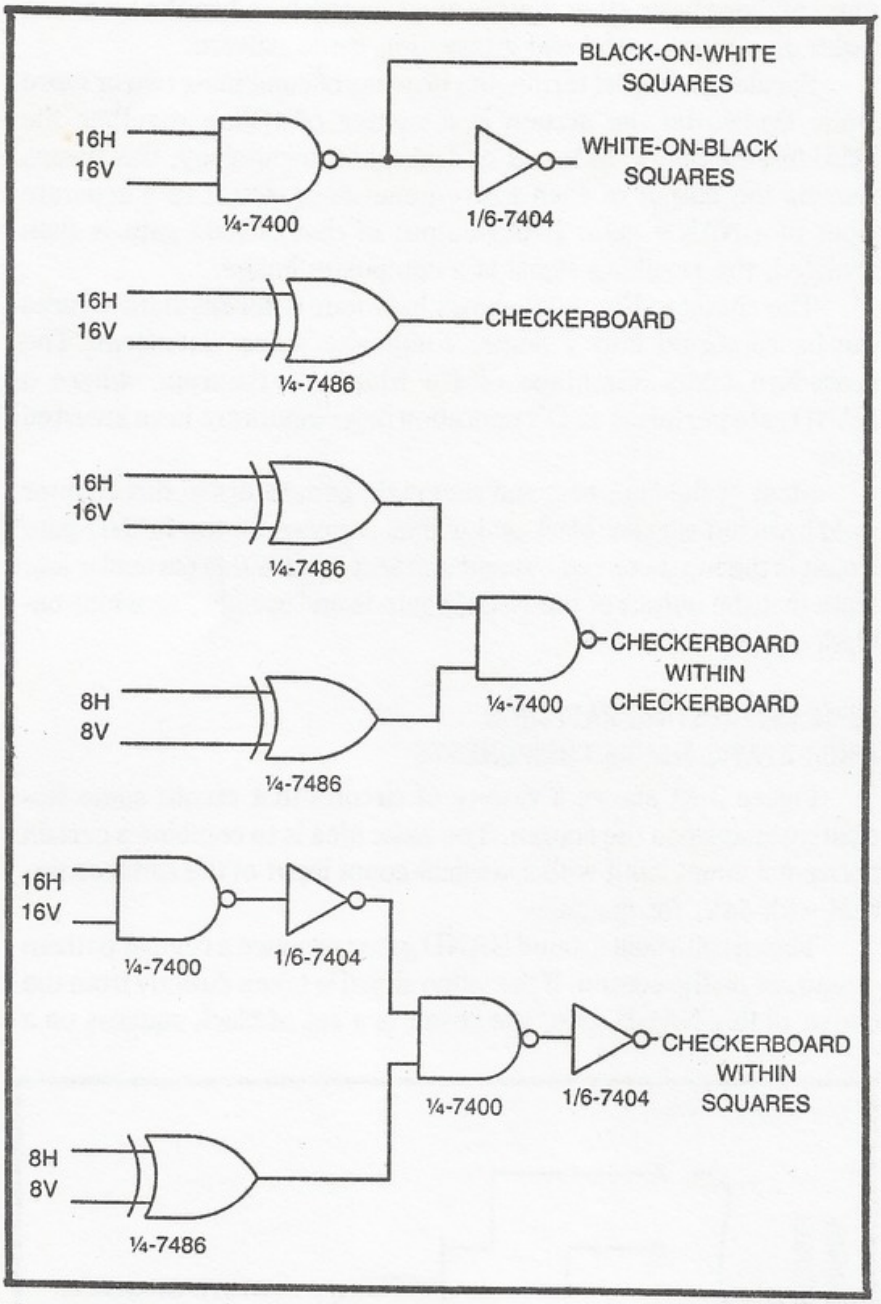


Fig. 3-21. A few circuits for generating some special static-figure effects.

white background. If, on the other hand, the signal from the NAND gate is inverted before applying to GAME VID IN, you will find white squares on a black background.

Combining a pair of horizontal- and vertical-count signals at the inputs of an EXCLUSIVE-OR gate produces a very distinctive checkerboard pattern.

Whether you use the NAND-gate squares or EXCLUSIVE-OR checkerboard, the pattern becomes finer as the input signals decrease in order. A 32H, 32V checkerboard pattern, for instance, is much more coarse than one generated from 8H, 8V inputs.

Checkerboards within checkerboards, checkerboards within squares, squares within checkerboards, and so on make some fascinating images on the screen. Many of them are potentially useful for special effects in TV games you will want to develop later on.