

ATARI PONG E CIRCUIT ANALYSIS & LAWN TENNIS: BUILDING A DIGITAL VIDEO GAME WITH 74 SERIES TTL IC's.

Dr. H. Holden.

UPDATED 2013- see end of article:

- 1) Questions & Answers with Mr Allan Alcorn
- 2) The Ghost in the Machine.

Background:

In the early 1970's a new product was born. It was a digital video game which simulated table tennis. Digital video games had existed prior to this however none were executed so elegantly or were such a commercial success. The ball in this Ping Pong game was merely a short segment of 4 lines of video signal seen as a small moving square on a TV monitor. The new arcade game was arcade Pong by Atari. It came in a large yellow and brown cabinet. The video screen image included a net, two bats and a score system for each player. Every time the player missed the ball then the opponent would gain a point. It was "electronic table tennis". Sometimes they were called Paddle games as the bat was referred to as a paddle in the USA.

As well as video, sound was also produced when the ball struck the bats or when losing a point or when bouncing off the top or bottom of the image. It has been said that Pong ushered in the video arcade game era and also the home video game era that followed in the mid to late 1970's. One could say that this game started it all.

The Pong game came to New Zealand in the early 1970's when I was a teenager. As luck would have it a friend (David Williamson) worked as a technician in the arcade game and juke box industry at a company called Combined Enterprises in Auckland. I had the opportunity to play

the game there and watch the game being assembled and the pcb's being serviced. David kindly explained how the logic gates worked and drew up the truth tables for some of them. I was determined that one day I would figure out exactly how the game worked, be able to repair them and build one myself.

Atari's Arcade Pong PCB contained 66 IC's. Gates and flip flops of every kind, a pair of 555 timer IC's and a few transistors. It was simply hard wired TTL logic and predates microprocessor and software controlled video games.

Very quickly in the years to come a version of the entire circuit was fabricated with large scale integration (LSI) onto a single IC. These IC's were used in home pong units which contained RF modulators so they could be linked to a home TV set.

Also pretty well right away after Atari released Pong other companies started making the arcade game as Atari didn't patent the design. Many were identical copies with some differences in the design and the screen image.

The photos below show Atari's original arcade game console (yellow) and two knock offs by Chicago Coin and Midway:

THE NEWEST 2 PLAYER
VIDEO SKILL GAME

PONG

from ATARI CORPORATION
SYZYGY ENGINEERED

The Team That Pioneered Video Technology

FEATURES

- STRIKING Attract Mode
- Ball Serves Automatically
- Realistic Sounds of Ball Bouncing, Striking Paddle
- Simple to Operate Controls
- ALL SOLID STATE TV and Components for Long, Rugged Life
- ONE YEAR COMPUTER WARRANTY
- Proven HIGH PROFITS in Location After Location
- Low Key Cabinet, Suitable for Sophisticated Locations
- 25¢ per play

THIS GAME IS AVAILABLE FROM YOUR LOCAL DISTRIBUTOR

SEEBURG MID-ATLANTIC CO.
1812-18 WORCESTER ST.
BALTIMORE, MD. 21230

Manufactured by

ATARI, INC.
2962 SCOTT BLVD.
SANTA CLARA, CA.
95050

Maximum Dimensions:

WIDTH - 26"
HEIGHT - 50"
DEPTH - 24"
SHIPPING WEIGHT:
150 Lbs.



It's a smash, slam—a real hit—Chicago Coin's New

TV PING PONG

Get set for 2-player TV PING PONG from Chicago Coin, the newest leader in electronic games. This exciting money maker actually simulates real ping pong on a full size television screen.

Players serve and return an electronic ping pong ball across the screen. Free wheeling front panel control knobs operated by two players allow simulated ping pong results. The best return shots, a built-in static noise, captures the sound of the racket hitting the ball. Scoring is according to official rules and is shown on digital readouts above the screen.

All the skills and speed of real ping pong play are needed for TV ping pong.

LOCATION TESTED FOR PROFIT
Consistently outstanding money maker, because of high player interest, at pre-introduction test locations.

MOVING GAME
First player to reach score of 15 ends game. Adjustable to final score of 11.

EASY MAINTENANCE
Front panel removable for fast service access.

SOLID-STATE ELECTRONICS
Large 19-inch television screen and modern electronic circuitry. No warm up needed. Shielded screen.

RIM
Completely self-contained in handsome wood grain finish console. Kick plate for added protection.

2 PLAYERS FOR 25¢

CHICAGO COIN MACHINE DIV.
CHICAGO DYNAMIC INDUSTRIES, INC.
3728 W. DIVERSEY BLVD., CHICAGO, ILLINOIS 60614

WINNER
from MIDWAY

The television skill game
that fits the mood of any scene.

- Large Size Solid State T.V.
- Additional Adjustable Timer Control
- Dimensions: 26½" wide, 24" deep, 64" high

MIDWAY MFG. CO. 3750 River Road—Schiller Park, Illinois 60176 phone: (312) 676-1333
DISTRIBUTED BY

PRINTED IN U.S.A.



Screen image on original Atari Arcade game which used an 11 inch diagonal Hitachi TV as the monitor: (To take this photographic image the ball motion was disabled).

Despite the lack of software, the ball in the game can have 42 distinct velocity vectors or directions and speeds (see analysis below).

Notice how the two clones of Atari's Pong above (Chicago Coin & Midway) have the typical 1970's fashions, artwork and colours. They also they used larger TV monitors than Atari. Atari used a small 11 inch diagonal "off the shelf" Japanese (Hitachi) TV set and disabled its TV receiving system and injected the video signal and audio into its video amplifier and audio amplifier so as to use it as a TV monitor.

After making Pong, Atari went on to design a number of video arcade games including TANK where two players fought it out on a battlefield with mines and in a maze while controlling two small tank images that could fire shells.

Later Atari produced the famous Asteroids game. It was quickly determined that the players liked to fight it out, either with the machine, or a competitor player. Here is an interesting yet disturbing scan taken from Atari's computer logic manual for their Tank game around 1974:

But most importantly, the designers of Tank have tapped into a heretofore unused reservoir of violent and competitive energy found in nearly all players. Historically, video games have employed non-violent competition between players (e.g. all paddle and driving games) or violent competition between a player and the machine (e.g. Computer Space). But no game has as effectively enabled the players to so violently and aggressively attack each other while yet remaining within the limitations of social acceptability.

So not only did Atari usher in the dawn of the commercial video gaming industry they also had identified an unpleasant human quality and

realised that it could not only be reinforced but also it would pay off financially via coins in mechanisms.

Later software based games became more detailed involved and realistic. Incredibly violent themes appeared such as those in video games such as Doom where the opponents are blasted into piles of blood and guts “all within the limitations of social acceptability”

This certainly makes the original Pong Paddle tennis game look quite innocent and charming by comparison.

ATARI PONG – THE GREAT CIRCUIT TEACHER:

Atari's original pong circuit used most of the digital techniques available to the designer (Allan Alcorn) in the early 1970's. The use of every sort of gate, NAND, NOR, XOR in various configurations, JK Master-Slave flip Flops, Edge triggered Flip Flops, Multiplexers, Ripple counters, Synchronous counters with modulo $N + 1$ counting, Decade counters, Binary to 7 segment encoders, a binary adder IC, Inverter gates, the linear 555 timer IC and the National Semiconductor LM309 5V voltage regulator IC.

Study of the circuit design is an excellent learning experience for those wishing to learn hard wired or “glue logic” TTL circuit techniques.

Importantly and cleverly the game was configured to behave in a realistic manner. For example when the ball struck the player's bat, the position it struck the bat determined the trajectory at which the ball would leave (or reflect) from the bat. This was done by dividing the bat into segments and simply reversing the existing horizontal component of the ball's motion and encoding data from the intersection of the bat segment and ball and using this data to control the vertical component of

the ball's motion. Also if there was a volley of hits (with no misses) then the horizontal component of the ball's motion would progressively increase. This simulates the rapid play of an increasingly stressful volley until one player would inevitably miss the ball and the horizontal speed is re-set.

Also a "net" as a broken white vertical line was shown to divide the screen into two halves and seven segment scores mixed into the video to record the points for each player. Each player could move his/her bat up and down vertically to attempt to engage the moving ball.

Very shortly after Atari released the arcade game the pcb was cloned by others. There were reports of blank pcb's being sold in the UK through hobby shops and the buyer would populate them with the TTL IC's. Atari published the schematic, but there was little information available at the time on how the circuit actually worked. This information would have been in the form of timing and pulse diagrams and logic tables and circuit descriptions.

Chicago Coin must have figured out the circuit functionality as they made an interesting modification in the vertical velocity encoder circuit to make their game play a little differently. This would have been difficult without a good working knowledge of its operation. Despite this their pcb was still nearly a 1:1 copy.

Over the years the game has also been emulated in software to play on computers. However I have never encountered a version in software that plays exactly the same as the original game and in most cases it is a poor facsimile of the real thing. This is probably because the exact features of the games bat (paddle) architecture, vertical velocity encoder and controller and horizontal velocity controller are not known to the software engineers making the "pong emulator" and these details are not

immediately evident from inspecting the schematic diagram. In addition the non interlaced scan would also need emulation and the 42 velocity vectors which are described here for the first time. With the information in this review article it would be possible to craft an exact emulator.

With Atari's later games such as TANK, they produced a very comprehensive "computer logic" manuals with timing diagrams, oscilloscopes and circuit descriptions.

Who designed Pong?

The original design of Atari's original Pong belongs to Mr. Allan Alcorn who was a brilliant young TTL logic designer working for Atari in the early 1970's. Therefore the credit for the design of any Pong circuits (unless otherwise noted) in this article belong to him and not to the other Arcade Pong makers who copied his work.

To get to grips with Atari's original design it is necessary to review the basic principles of logic gates and DeMorgan's theorem and also look at the ways in which XOR's can be realised with novel gate arrays.

Firstly when a line is drawn over a letter it means that the logic level is reversed, like passing it through an inverter. In this written text an inverted pulse A would be written "(not)A".

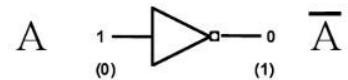
In general most TTL designers used negative true logic, meaning that the output of the logic gate would be low in its active or true condition. There are a number of advantages of this. TTL IC outputs for example are good at sinking current but poor at sourcing it. Although in any given circuit the reverse or inverted version of the output was often needed. In Atari's Pong circuit there are no AND gates for example, but plenty of NAND gates. NAND gates can also function as OR gates with

negative true inputs. This fact and other gate equivalencies are shown below in DeMorgan's Theorem:

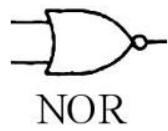
DeMorgan's Theorem:

$$A \cdot B = A \text{ and } B$$

$$A + B = A \text{ or } B$$

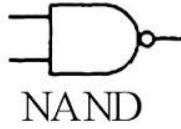


$$1) \quad \overline{A+B} = \overline{A} \cdot \overline{B}$$



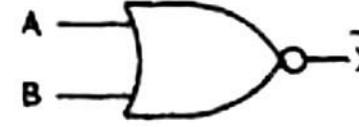
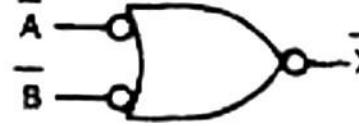
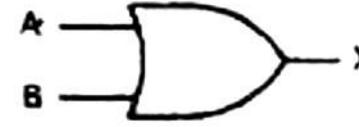
$$= \quad \text{Negated input AND}$$

$$2) \quad \overline{A \cdot B} = \overline{A} + \overline{B}$$



$$= \quad \text{Negated input OR}$$

In Atari's original Pong schematic often one type of gate in the same IC package is drawn in different ways according to the theorem and gate equivalency. It is drawn this way to help the reader understand what the gate is trying to do. For example if you have two logic levels and you want an output going high that occurs when either input is low, you would draw the gate as a negated input OR gate, even though its actually a NAND gate such as a SN7400 IC. If you drew it as a NAND it would imply that you wanted a low output when both the inputs were high. The table below summarizes gate equivalency and gate truth tables:

NAND GATE		TRUTH TABLE															
		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>H</td><td>H</td><td>L</td></tr> <tr><td>H</td><td>L</td><td>H</td></tr> <tr><td>L</td><td>H</td><td>H</td></tr> <tr><td>L</td><td>L</td><td>H</td></tr> </tbody> </table>	A	B	X	H	H	L	H	L	H	L	H	H	L	L	H
A	B	X															
H	H	L															
H	L	H															
L	H	H															
L	L	H															
NOR GATE																	
		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>H</td><td>H</td><td>L</td></tr> <tr><td>H</td><td>L</td><td>L</td></tr> <tr><td>L</td><td>H</td><td>L</td></tr> <tr><td>L</td><td>L</td><td>H</td></tr> </tbody> </table>	A	B	X	H	H	L	H	L	L	L	H	L	L	L	H
A	B	X															
H	H	L															
H	L	L															
L	H	L															
L	L	H															
AND GATE																	
		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>H</td><td>H</td><td>H</td></tr> <tr><td>H</td><td>L</td><td>L</td></tr> <tr><td>L</td><td>H</td><td>L</td></tr> <tr><td>L</td><td>L</td><td>L</td></tr> </tbody> </table>	A	B	X	H	H	H	H	L	L	L	H	L	L	L	L
A	B	X															
H	H	H															
H	L	L															
L	H	L															
L	L	L															
OR GATE																	
		<table border="1"> <thead> <tr> <th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>H</td><td>H</td><td>H</td></tr> <tr><td>H</td><td>L</td><td>H</td></tr> <tr><td>L</td><td>H</td><td>H</td></tr> <tr><td>L</td><td>L</td><td>L</td></tr> </tbody> </table>	A	B	X	H	H	H	H	L	H	L	H	H	L	L	L
A	B	X															
H	H	H															
H	L	H															
L	H	H															
L	L	L															

The other important type of gate not shown in the table above is the Exclusive OR, or XOR gate. This gate produces a high output when either one of its inputs is high, but not both at once. There is also no output when both inputs are low. This gate has multiple uses. It can be used in phase detector circuits in PLL's (phased lock loops) in radio equipment. XOR gates can also be used to reverse a logic level passing through it by switching the other input high. This function is used in Pong to reverse data values and result in the Ball in the game reversing its vertical direction at times. XOR gates can also be used as pulse or

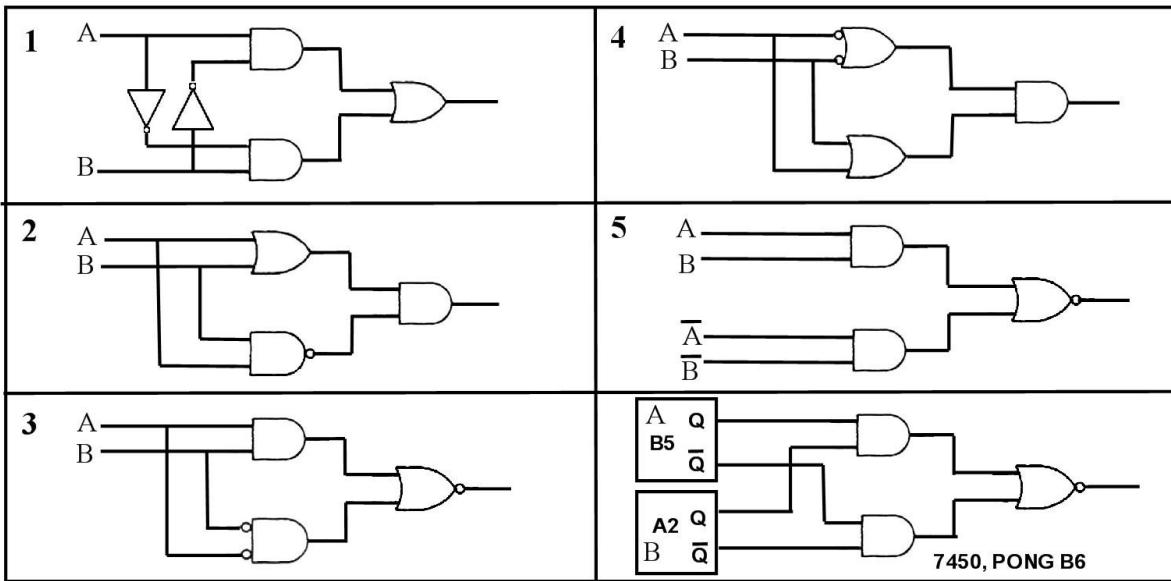
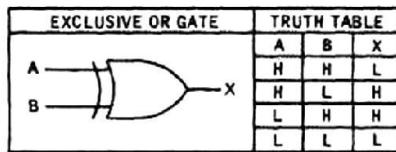
signal mixers, for example to mix Horizontal and Vertical synchronisation pulses.

The truth table for the XOR:

EXCLUSIVE OR GATE		TRUTH TABLE		
A	B	X		
H	H	L		
H	L	H		
L	H	H		
L	L	L		

The XOR gates used in Atari's PONG are the 7486. However there are ways of making XOR gates from other gate combinations and one of these is used in Pong. Due to the clever way this has been done it is necessary to take a look at the possible ways an XOR gate could be constructed from other standard gates:

XOR GATES & XOR IDENTITIES:



As can be seen from the table above there are a number of ways an XOR can be realised. Of particular interest here is version 3. Without the inversions at the lower AND gate inputs it can be drawn as version 5, a unique type of XOR gate that requires complimentary drive levels.

In Atari's Pong to realise this gate function they used the Q and Qbar outputs from two flip flops for the complimentary drive. This part of the circuit is in the vertical ball velocity encoder. Without knowing that this is an XOR identity, then that section of the circuit would be very difficult to understand. (The only curio here is that there was a spare XOR gate nearby that the designer didn't use and he opted for the XOR made from the 7450).

Other IC's used in PONG:

Apart from the gates there are a number of counter IC's and Flip Flops used in PONG. For the counter IC's it is best to look up their pdf data sheets for their truth tables. The types are 7493 (4 bit ripple counter) 7490 (Decade counter), 9316, or 74161 (synchronous counter with load & clear).

It is worth noting the two types of flip flops used, the 7474 (dual D *positive* edge triggered) and the 74107 which is a dual JK master slave type where the data is transferred from the master to the slave when the clock pulse falls low. These positive and negatively clocked properties are taken advantage of in the design.

Other types include the 74153 which is a multiplexer used as a data selector.

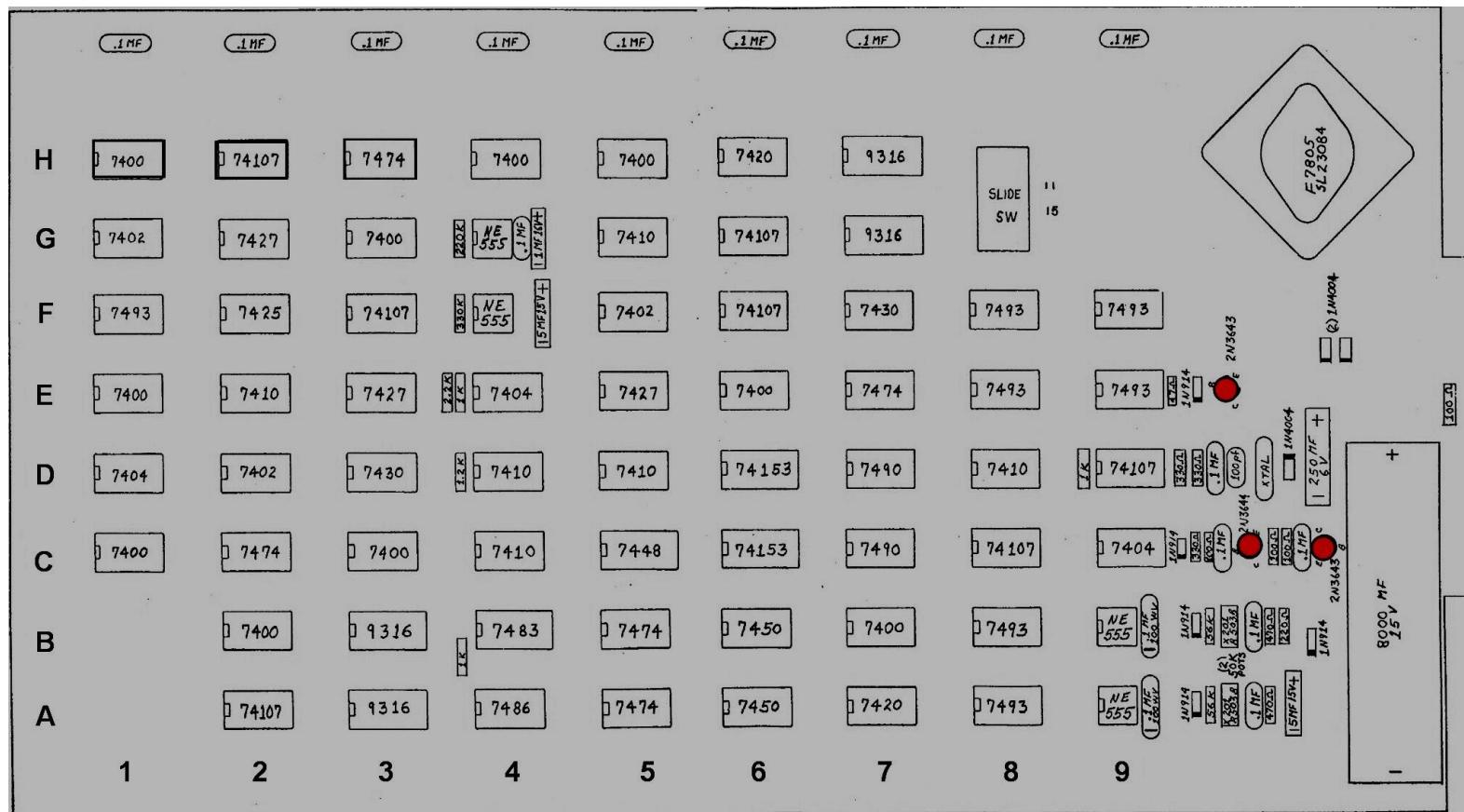
The 7448 converts BCD into seven segment data (for the scores).

A binary adder IC, the 7483 used in the vertical ball velocity encoder.

Also the four 555 timer IC's used in the bat generator circuits (two) and one for the serve timer and the other in the sound system in the score sound timer after a player misses the ball.

When studying this sort of design it is best to refer to Atari's original arcade PCB topography as each TTL logic IC is spaced on a specific location, so when studying parts on the schematic the IC gates can be referred to by a number and a letter. The important parts of the schematic have been reproduced here. A large form paper copy is the best one to have.

The diagram below shows the top view of the PCB. The three transistors are shown in red:



(There are a number of subtle and very clever features built into Pong, these will be discussed. In addition the original Pong game had a number of technical issues or problems. All of these, and the solutions for them, are addressed in this article).

PONG ARCHITECTURE & CIRCUIT ANALYSIS:

The first step in creating a video game as Atari did is to create Horizontal and Vertical synchronisation pulses of the correct frequency to lock the scanning raster on the CRT's face.

The American Television system specified a horizontal or Line frequency of 15,750 Hz and a field frequency of 60Hz. (In NZ, Australia, or the UK, the frequencies are 15,625 Hz and 50 Hz respectively). However most monochrome TV's or monitors have enough range in the H.Hold and V.Hold controls to lock to either system, some need modifying.

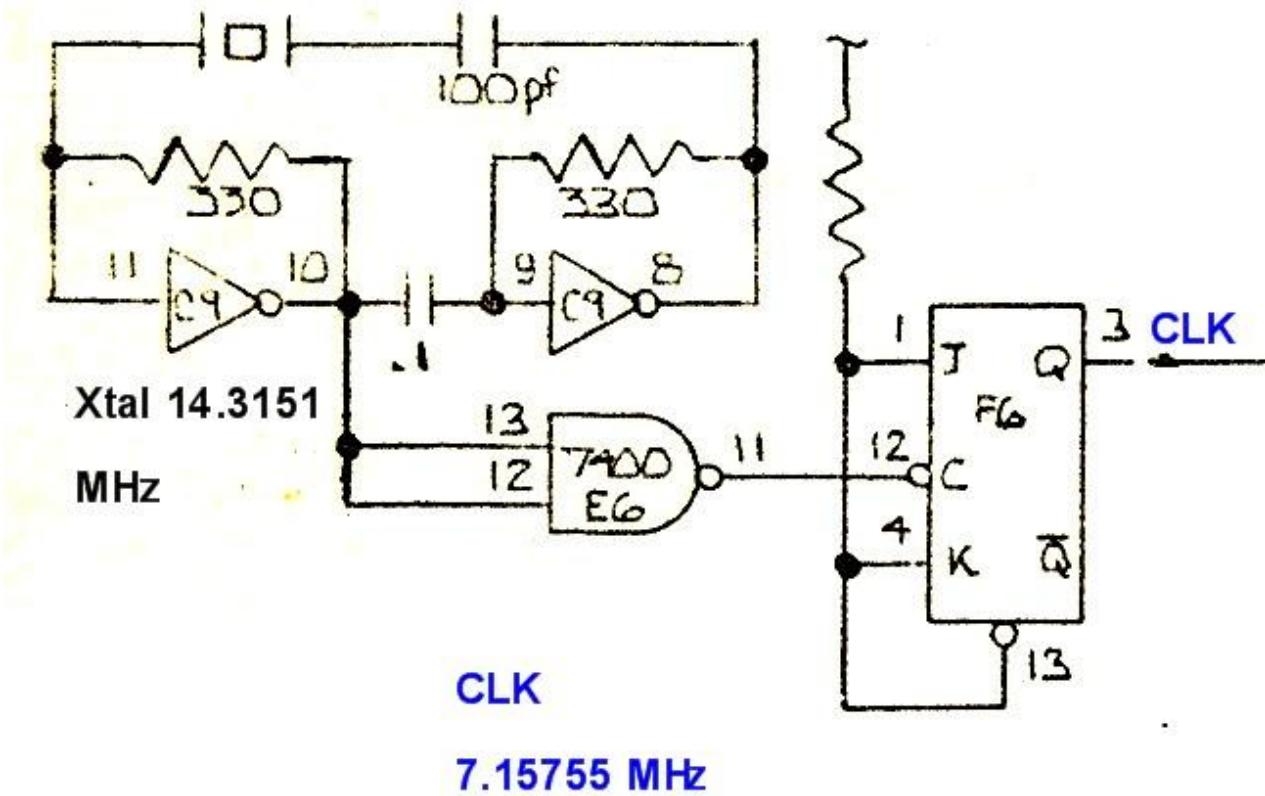
Sticking to the American video system for these examples, the scanning lines on the cathode ray tube (CRT's) face are seen by the eye as a frame of lines, 525 of them. They are scanned however 262.5 lines over one sweep of the vertical scan (60 Hz or 0.0166 s), then over the next sweep the other 262.5 lines are scanned to give the appearance of a 525 line scanning raster. Half the “frame” or one set of the 262.5 lines is called a *field*. So the “Field rate” is the 60 Hz frequency of the vertical scanning system and the frequency of the vertical synchronisation pulses. The line rate and the line synchronisation pulses are 15,750 Hz.

Due to the fact that the time it takes to scan one Field of lines is 1/60 seconds (including the time it takes to return the CRT beam from the bottom to the top of the picture) then only 262.5 lines can occur during this time. Therefore the second group of 262.5 lines end up fitting neatly between the first Field of lines scanned and this is called scan interlacing.

However in the synchronisation pulse generators of early video games like Pong, the sync pulse processors were not that sophisticated and the vertical frequency, or field rate, rather than ultimately being related to the horizontal frequency by a division rate of 262.5 was merely just the horizontal frequency divided by 262, which is the division rate of the vertical sync pulse generator in these early circuits. This means that each field of 262 lines is merely scanned on top of itself and not interlaced.

The non interlaced scan creates “gaps between the lines” and a coarse looking scanning raster, but this did not detract from the game (later Atari games such as Tank had scan interlacing and a more detailed video image).

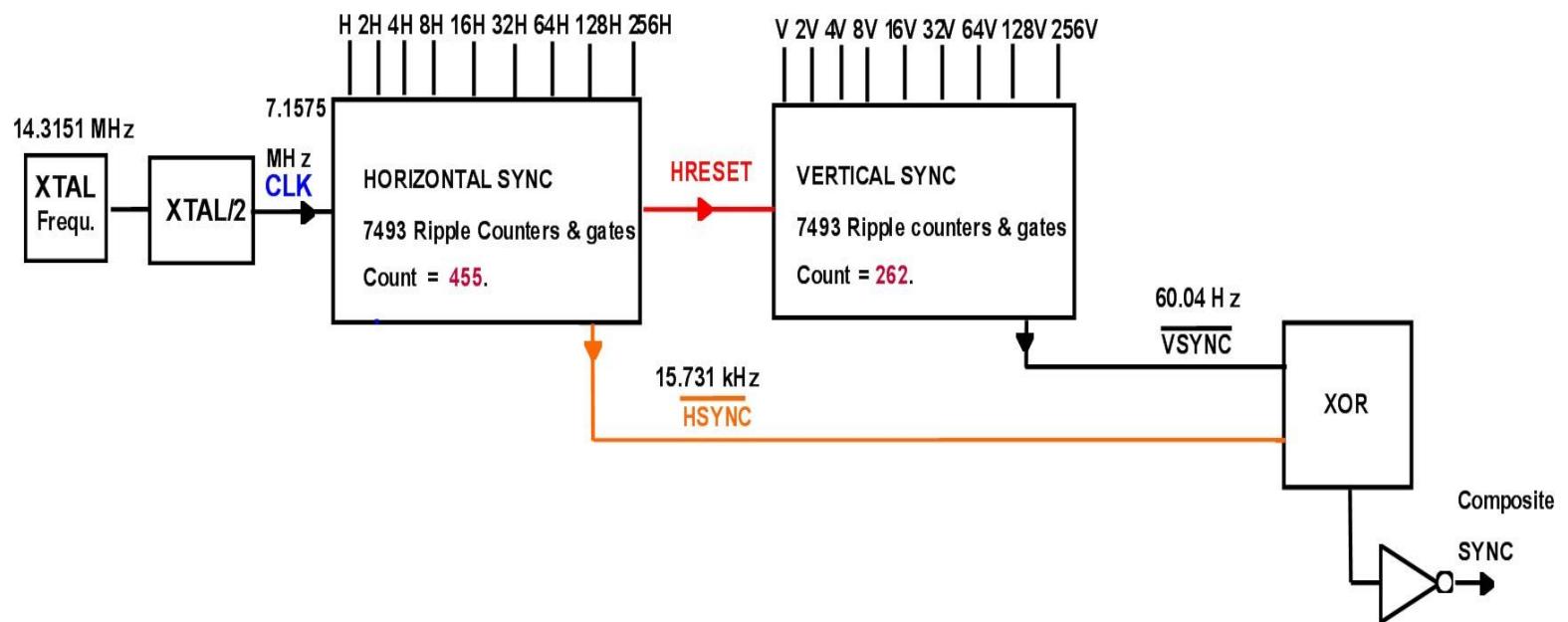
All digital video games have a master clock oscillator from which nearly all other logic signals are derived. Below is the Atari Pong master clock:



The master oscillator in Pong is a crystal controlled oscillator formed by two inverter gates C9 and it runs on testing at 14.3151 MHz due to loading even though it's a 14.318180 MHz labelled crystal. E6 acts as a buffer then the frequency is divided in half by the toggling JK flip flop F6.

The block diagram below shows the clock (CLK) and an array of 7493 ripple counters, flip flops and gates divide the clock down:

VIDEO SYNC GENERATION:



The 7493 IC contains 4 flip flops. Each flip flop output has half the output frequency of the preceding flip flop. For example if 7.1575 MHz is fed into the first flip flop the output from the first flip flop (on the H connection) will be $\frac{1}{2}$ that frequency, the second flip flop will be $\frac{1}{4}$ the frequency etc, as each flip flop divides the frequency by 2. So for example the output frequency on the "H" connection in the diagram

above is $7.1575 / 2 = 3.579$ MHz. The length of *one cycle* of H is the period or $1/3.579$ MHz and the length of the H pulse is half that as they are square waves. So for example **1H is close to 0.14uS** long and 4H for example is $4 \times 0.14 = 0.56$ uS long.

Appling the same logic to the vertical sync generator the *length 1V is close to 63.5uS* which is the horizontal period or one line. 4V therefore has a duration $4 \times 63.5 = 254$ uS or 4 horizontal scanning periods.

(Remember the figures for **1V and 1H** in bold above)

Also, for example, the 256H pulse would be 256×0.14 uS = 35.8uS or about half horizontal scan time and divide the screen in half. This is used to position the net for example. Also for example 128V pulse is 128×63.5 uS = 0.0081 sec or about half the vertical scan time (of 0.0166sec) and would divide the screen in half from top to bottom for example.

Each flip flop output can then be labelled H, 2H, 4H etc. The length of any pulse can be determined from the output label. For example a 2H pulse is twice as long as a 1H or a 4H pulse 4 times as long as a 1H pulse etc.

One 7493 containing 4 flip flops can only divide by 16 (4 bit output) and with another 4 flip flops can divide by 256. But a division rate of 455 is needed, so a single additional flip flop is needed to create a 256H connection so the division rate can be the required 455. This is a 74107 JK flip flop designated F6 on Atari's original Pong schematic.

The ultimate division value is controlled by gates arranged around these flip flops and another D type flip flop (E7) to control the number of pulses counted before the flip flops are reset to zero and the process repeats.

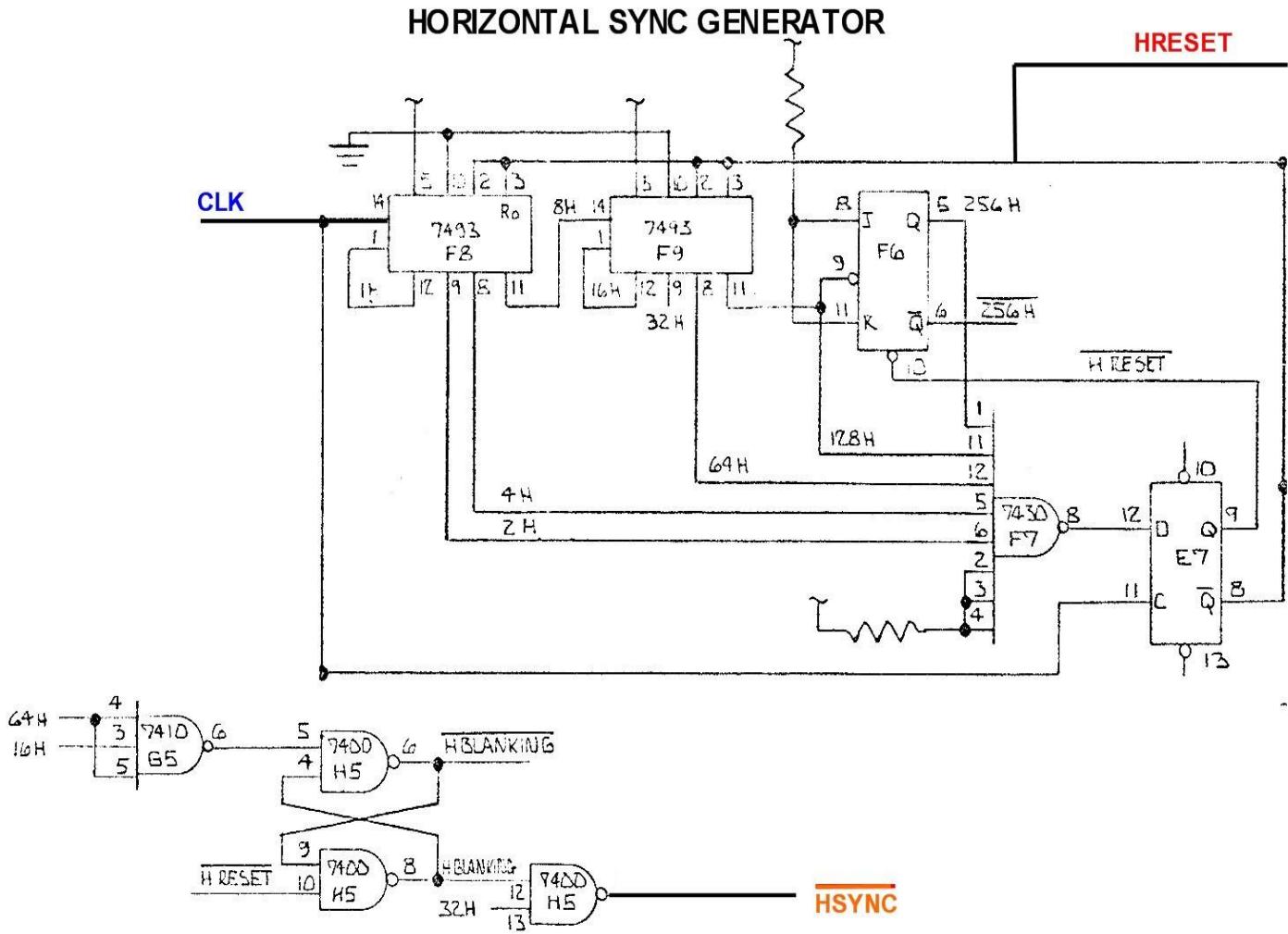
The horizontal reset pulse effectively accounts for one clock pulse when all the counters (in the horizontal sync section) are set to “zero”. So this system initially counts to $454 + \text{the reset pulse}$, which is a total of 455 counts.

Therefore the division value of the Horizontal counter system is 455.

The frequency of the output HRESET or HSYNC pulse, rounded to the nearest digit, is $7.1575 \text{ MHz} / 455 = 15,731 \text{ Hz}$, which is close to the Line scanning frequency of the TV monitor (15,750 Hz).

The outputs: H, 2H, 4H, 8H etc are very useful pulses and help to achieve the correct count and provide other important signals.

In more detail, the following circuit shows how this was achieved:



The flip flops in the 7493 have some of their Q outputs monitored by the 8 input NAND gate F7 (7430).

On the falling edge of the 454th clock pulse the output of the 7430(F7) pin 8 goes low. This is because on the first time around the 7493 counters count to their full capacity $1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255$ before the flip flop of F6 is clocked by the 256th pulse (the 7493 counters or flip fops overflow) to produce the 256 output on pin 5 of F6.

Then the 7493 counters have to count to $2 + 4 + 64 + 128 = 198$ to satisfy the input of F7 and cause F7's output, pin 8, to go low. The total count is then $198 + 256 = 454$. Then on the next *positive* clock pulse edge the low level, on the data input (D) of E7, is clocked to the Q output of E7 and the Qbar (or (not)Q) goes high. This generates the HRESET pulse and resets the flip flops and 7493's. The process then repeats, every 455 clock pulses, so it forms a “divide by 455” circuit.

The HRESET pulse falls low after the next master clock (CLK) pulse.

The 64H and 16H pulses and the HRESET pulse are used to create the (not)Horizontal Blanking or (not)HBLNK and HBLNK and 32H pulses are used to create the (not)Horizontal synchronising pulses or (not)HSYNC. Right away from the data above the horizontal sync pulse must be $32 \times 0.14\mu\text{s} = 4.48\mu\text{s}$ long.

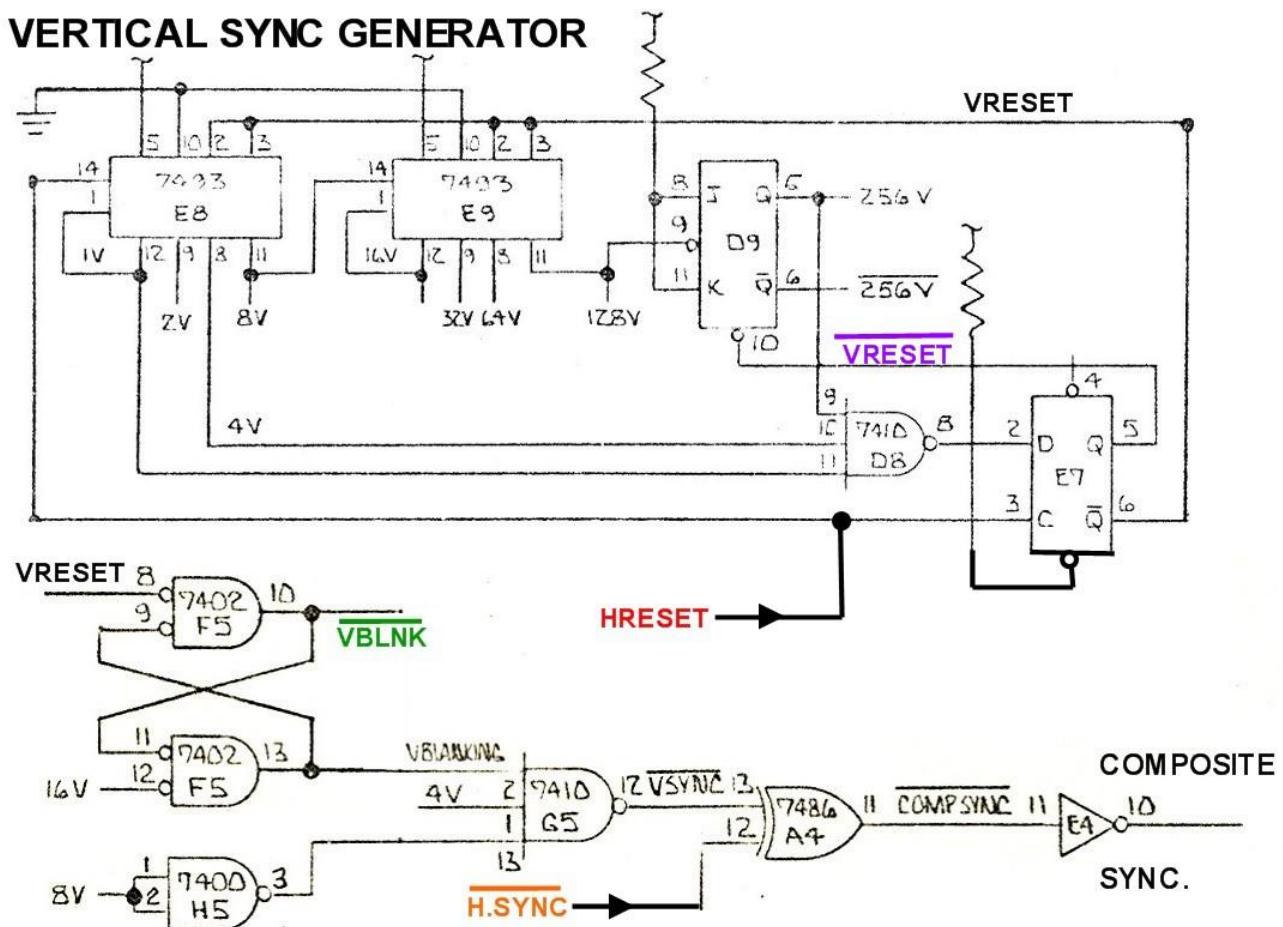
Overflow versus Reset counting and dividing:

The above example demonstrated some interesting points. If for example a 4 bit counter (such as a 7493) is clocked it will count to 15 and on the next pulse overflow and the 8, 4, 2, 1 outputs or D, C, B, A outputs will be zero. Therefore the frequency division of such a system is 16 because the zero also uses one clock pulse so the zero must be “counted” as well.

In the Horizontal sync circuit above the counter chain overflows once in the process to set the 74107 to create the 9th bit of the counter and uses a clock pulse after the 454th clock pulse to reset to zero.

Any counter or divider system needs to be very carefully studied to determine exactly what the count and division rates are, especially if there are other gates and separate flip flops added into it. I have seen some documentation showing that the Pong circuit above is a 454 divider which is incorrect.

The Vertical sync generator:

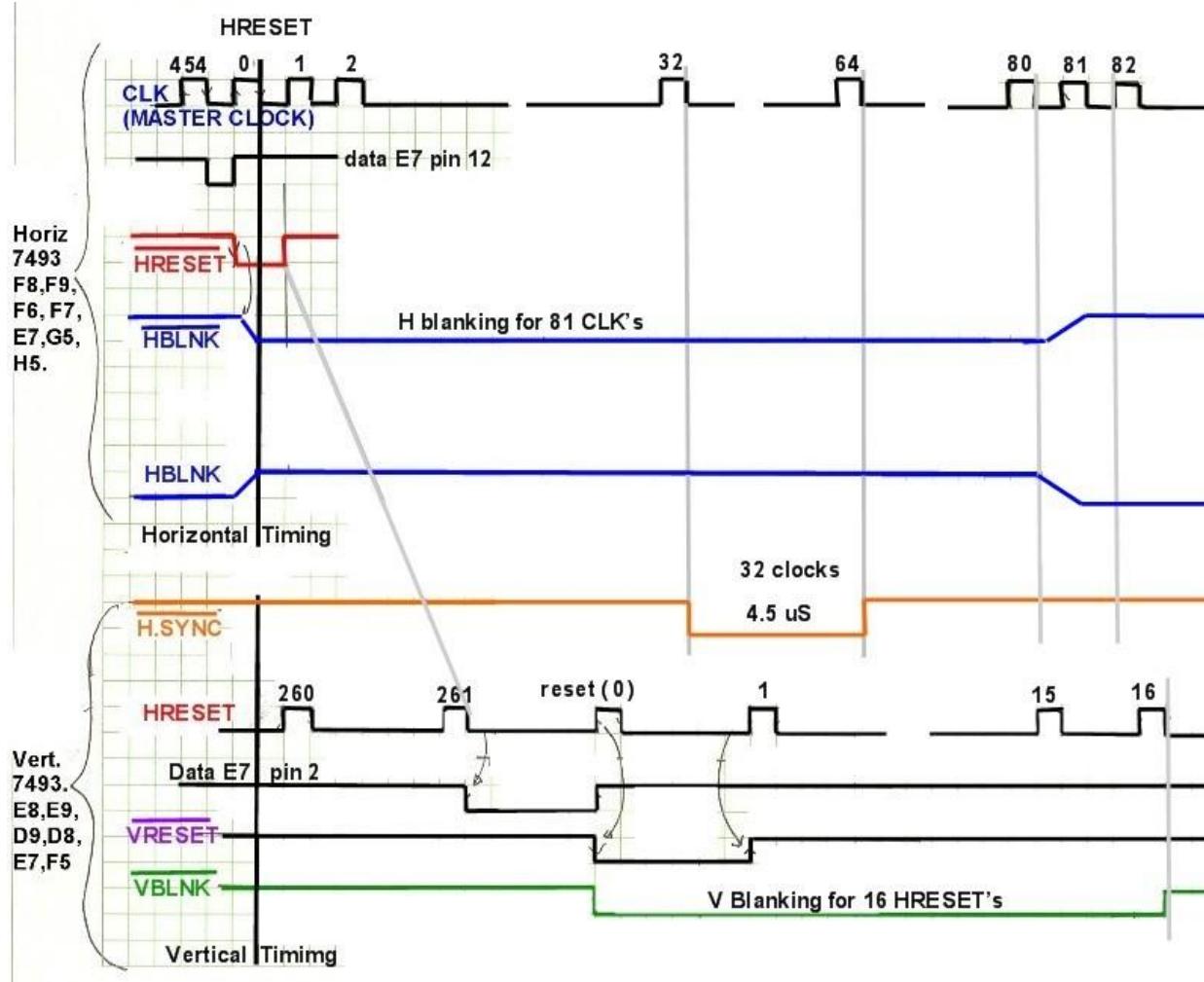


The vertical sync circuit is the same configuration as the horizontal sync system, except the drive pulse is the HRESET pulse, not the master clock or CLK.

During counting after 256 counts this sets pin 5 (the 256V connection) of D9 high, then after another $1 + 4 = 261$ counts to satisfy the inputs of the 7410 NAND gate D8, the output of D8 falls low. Then on the 262nd HRESET pulse the counters are reset.

The division of the horizontal rate is therefore 262. HRESET has a frequency of 15,731 Hz, so VRESET has a frequency of $15731/262 = 60.04$ Hz which is the Field frequency for the monitor or very close to it. The horizontal and vertical sync pulses are mixed together by the XOR gate A4 to become (not)composite sync.

Now it is useful to look at the basic timing sequence of the sync generators:



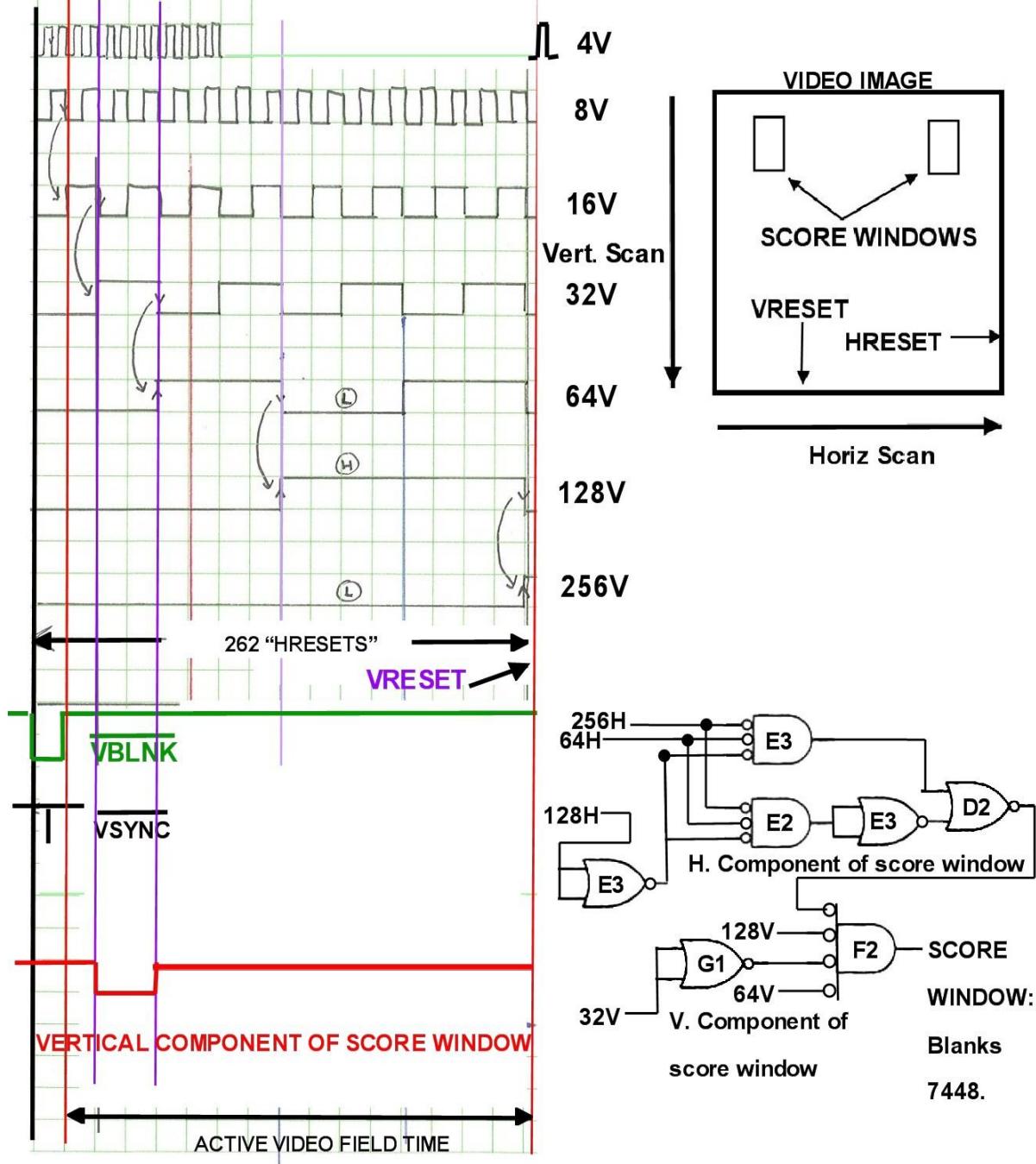
The timing of horizontal blanking (HBLNK) due to delays in the flip flops & gates is such that (not)HBLNK falls low during the zero master clock(CLK) pulse being high, or about 0.04uS after the rising edge of the CLK pulse and goes high again just after the rising edge of the 81st clock pulse. This is about 0.11uS after the falling edge of the 80th clock pulse which triggers this event. (These important delays can be calculated from the data on the flip flop and gates data sheets, or measured on a scope)

Therefore, the (not)HBLNK pulse is low for precisely 81 *positive going* master clock pulse edges. This is very important as the (not)HBLNK is used to inhibit the positive edge triggered 9316 horizontal ball position counters (see below) over exactly 81 clock pulses at the T enable input. From the data sheet of the 9316 : “ High to Low level transitions at the enable P or T inputs should occur only when the clock input is High”

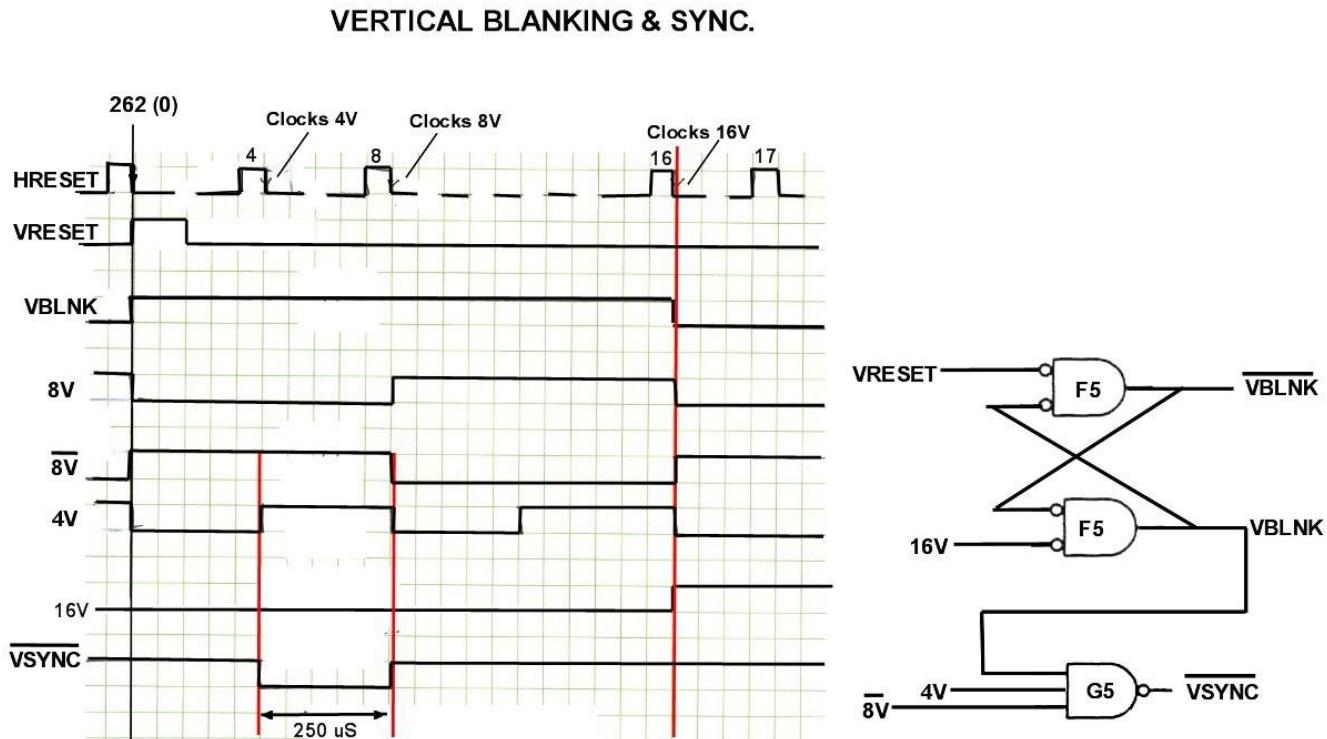
One interesting point is that counters such as the 7493 are “ripple counters” in that the pulses ripple down the chain of flip flops. For the pulses further down the divider chain there is a delay with respect to the timing of the master clock transitions due to the cumulative delays of each flip flop in the chain. This does not happen with synchronously clocked counters. These delays caused some problems in the Atari pong circuit in some cases. For example the signal which generates the net (see below) uses the 256H signal and the CLK to generate the net pulse. In some later production run Pong pcb’s the total delays in the SN7493AN counter IC’s were on the very minimum edge of the specs and this compounded through the divider chain. The altered timing or clock skew ended up producing a thin and weak looking net pulse in the video. This can be cured by altering the circuit or adding some delay (see section on modifications to Pong).

Horizontal and vertical pulses from the sync generators can be used to either create or select various parts of the video image. As an example, shown on the diagram below, where they are used to select a window of time (and position) in the video scanning raster where the scores can be displayed. The gate configurations in the diagram show how selecting various H and V pulses can select various areas on the screen.

VERTICAL SYNC GEN. 1V, 4V & 256V HIGH SCORE WINDOW:



More of this sort of gating is used to select the actual score segments which are displayed in the score window areas. The equivalent timing diagram for the horizontal sync and the horizontal component of the score windows is shown later on. The following diagram is an expanded area around vertical sync to show how the vertical sync pulse is generated:



As (not) VSYNC is 4V wide, right away it is easy to see that it is $4 \times 63.5\mu\text{s} = 254\mu\text{s}$ wide. (This is the advantage of remembering the 1V and 1H values cited above).

VIDEO SYNC & BLANKING VIDEO STANDARDS vs PONG:

In the standard American analog video system the vertical sync pulse is about 3 horizontal periods long or about 190 μs wide. In that standard

interlaced video system there are equalising pulses before and after the vertical sync pulse which is serrated. The non interlaced Pong system does not have these. Instead it has just a block shaped pulse for the vertical sync.

Also, in the standard TV system, typically the standard video blanking interval is 0.05 vertical intervals, or 833uSec, or nearly 1000uS long.

In Pong the HRESET frequency is 15,731Hz, or a 63.5uS period and this is reduced to 1/ 32 of that frequency at the 16V output to 491.6 Hz, which has a period of 2.03mS and half that period is 1017uS. This could have simply been calculated from $16V \times 63.5uS = 1016uS$, because as noted above $1V = 63.5uS$. Therefore the length of VBLNK in Pong at a value of 1016uS is “close” to the 833uS standard.

The vertical sync pulse is ¼ of VBLNK, encoded by the 4V pulse, making it about 250uS long or close to the 190uS standard. The gate array above shows how the (not)VSYNC is generated from VRESET, 4V & 16V and (not)8V.

Blanking intervals are the vertical and horizontal intervals between active scanning lines on the monitor. During these times the sync pulses are inserted. The blanking areas are off the visible screen usually. The vertical blanking can be seen by adjusting the vertical hold on the monitor and making it roll by. In pong this allows the vertical sync to be seen as a black bar.

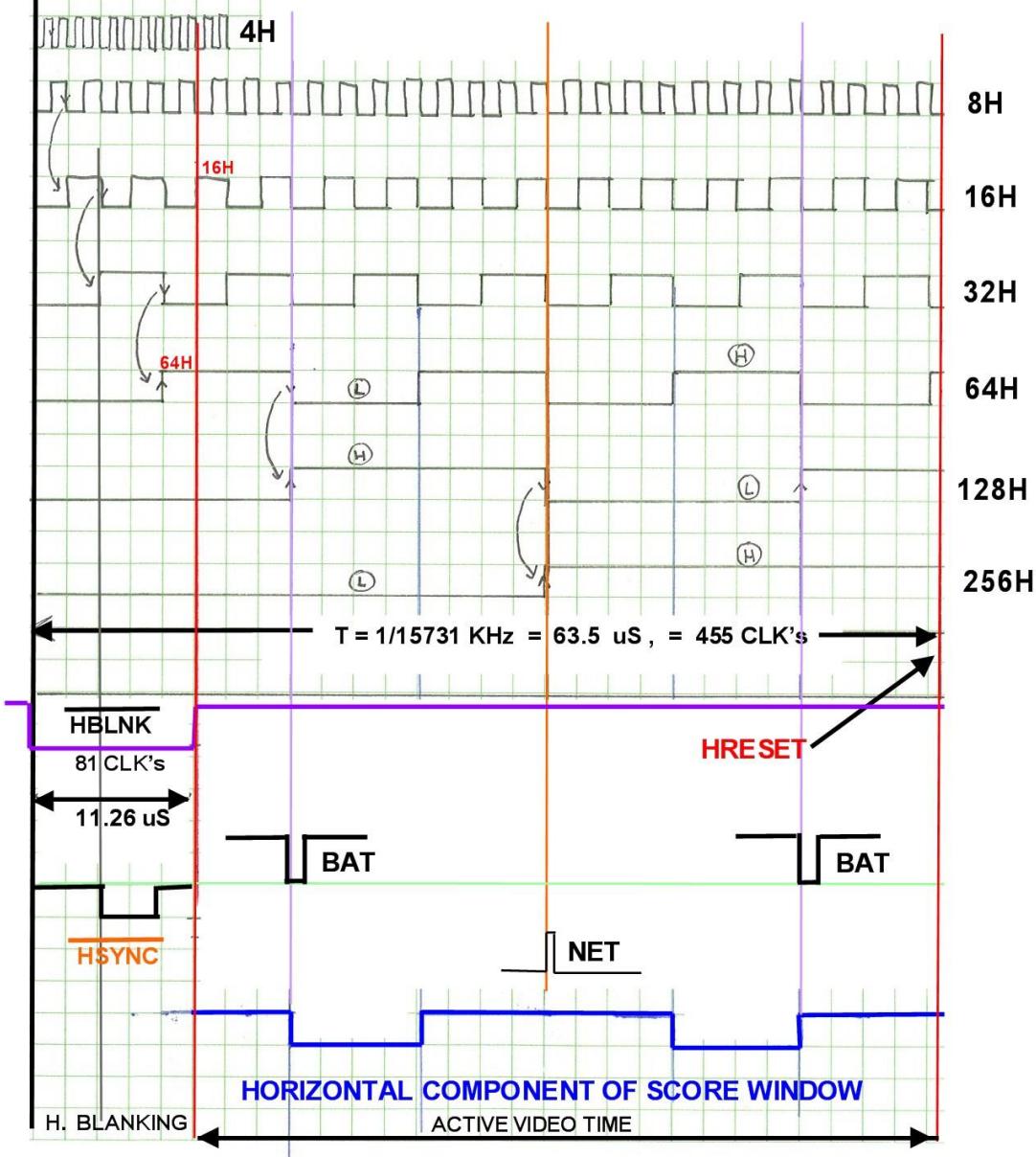
The standard horizontal blanking interval of the American analog system was 0.16 horizontal periods or about 10.2uS. The sync pulse leading edge was inserted 1.27uS after the beginning of blanking, a time delay called “the front porch”. The sync pulse width was 5.08uS long, leaving a “back porch” of about 3.81uS to the end of blanking time.

The width of horizontal blanking in Pong is coordinated by 64H and 16H (= 80H see h sync circuit gates above), $80 \times 0.14\mu\text{S} = 11.2\mu\text{S}$ for Pong horizontal blanking is close to the standard 10.2 μS .

The width of the sync pulse, coordinated by 32H, so is $32 \times 0.14\mu\text{S} = 4.5\mu\text{S}$, close to the 5.08 μS standard. However, in the original Atari Arcade Pong the (not)HSYNC pulse was gated in the wrong position within blanking and should have been set further to the left than shown in the diagram below. (see modifications section to fix this)

The Horizontal sync timing is shown below.

7493 Horizontal Timing, active video, scores and Bat gate.



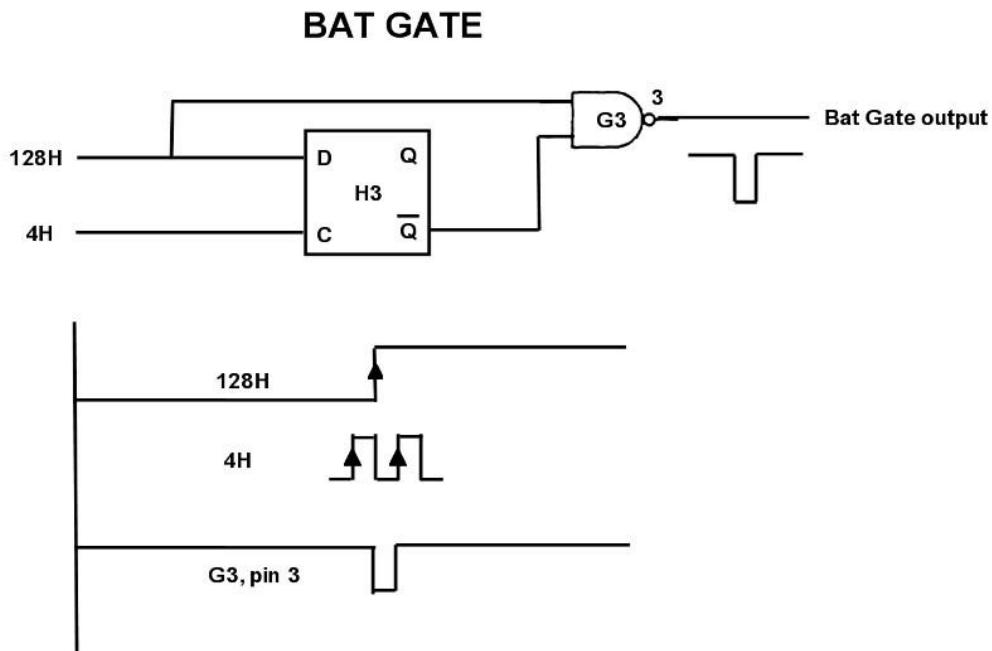
The chart above shows the position of the score window (in blue), the bats and the net relative to the horizontal system pulses.

HRESET occurs when 256H, 128H, 64H, 2H and 4H are high, which can be confirmed by looking at the 7494 horizontal sync circuit. There is not enough room on the above diagram to show the 2H pulses easily.

Also looking at the diagram above, the blanking or (not)HBLNK pulse is initiated with the HRESET pulse and is terminated by 64H and 32H (= 80H) pulses and you can see these are the pulses sent to gates H5 and G5 in the 7493 horizontal sync circuit.

Gating Pulses:

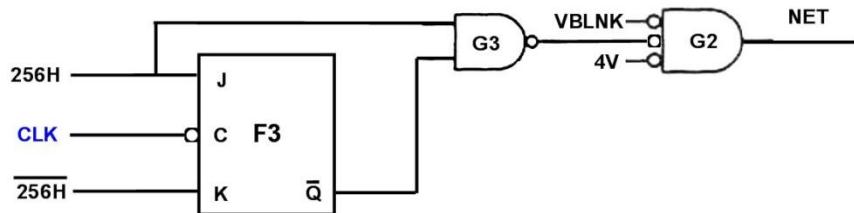
Looking at some gating in more detail, firstly the bat (paddle) gate, which is common form of pulse selector or pulse synchroniser:



The rising edge of the 4H pulse relentlessly clocks the low value of the 128H pulse to the Q and a high is on (not)Q. As soon as 128H goes high, the output of G3 pin 3 falls low. On the next positive going edge the 4H clocks a High to Q and a low to (not)Q so the output of G3 goes high and the pulse is terminated. Looking at the timing diagram above this process happens twice across one horizontal scan so both players' bats are gated.

A similar circuit, but using a JK flip flop, is used to create the net pulse:

NET GATE:



This circuit produces a narrow pulse, equal to one clock (CLK) pulse length which is only about 70 nanoseconds long. Due to clock skew in generating the 256H pulses that position it on the screen center, some later Atari PCB's had a very narrow net pulse that was visible but faint. How this happens and the solutions for this will be discussed near the end of this article in the modifications section.

It is worth noting here that the designer gated the net video signal pulse out of the vertical blanking interval using VBLNK and gate G2, however he omitted to gate the ball video pulse out of vertical blanking. Signals in the blanking interval of the video signal, other than sync pulses, can disturb the monitor's sync.

GENERATING & CONTROLLING THE BALL POSITION:

This is achieved by creating a duplicate set of horizontal and vertical counters to those in the sync generator circuit, but with a number of important differences. The counters are only allowed to count during active scan time when the ball is to be displayed. Also the total count can be controlled using the load inputs to the counters, which pre-loads the flip flops within them.

This is done with 9316(74161 synchronous counters). If the overall *effective count* matches 455 for the horizontal 9316 ball position counter and 262 for the vertical 9316 ball position counter then the counter outputs would be such that the pulse which is displayed as the ball is not moving. The term *effective count* was used as for periods the count is stopped or inhibited.

If the vertical 9316 counter is loaded with a different value, so it effectively divides by 261, the output frequency of the 9316 counter will be a little higher and the ball would be drifting upwards (arriving early) for example because the video raster is scanned from top to bottom in time. So you could think of the ball moving due to a frequency difference with respect to sync, and think of the ball controller as a ball *motion* circuit. However since it is a digital system and the ball position “counter” results in a different count over a field of time, placing the ball in a fixed different *position*, the system could be called a ball position controller as the designer chose to name it.

The exact counts within the 9316 counter ball position system have different actual numbers because of the inhibition of the 9316 counters during blanking time.

In both cases the horizontal and vertical 9316 counters control the ball's position and movement by altering the effective pulse division amount around the centre values of 455 and 262 of the sync system respectively.

In the case of the vertical 9316 ball position counter this only requires a change in the binary number (4 bit load data) loaded into the 9316 counter. The difference in overall output frequency (of changing ball position with time) is such that the ball motion is at a useful and useable speed for game play. The vertical system uses 4 bit load data as will be explained and a range of three counts above or 3 counts below the centre 262 value are enough to control the full range of vertical ball speeds up and down.

However in the horizontal counter system with a change to a division by one count only above or one below the centre value the relative ball motion is extremely rapid.

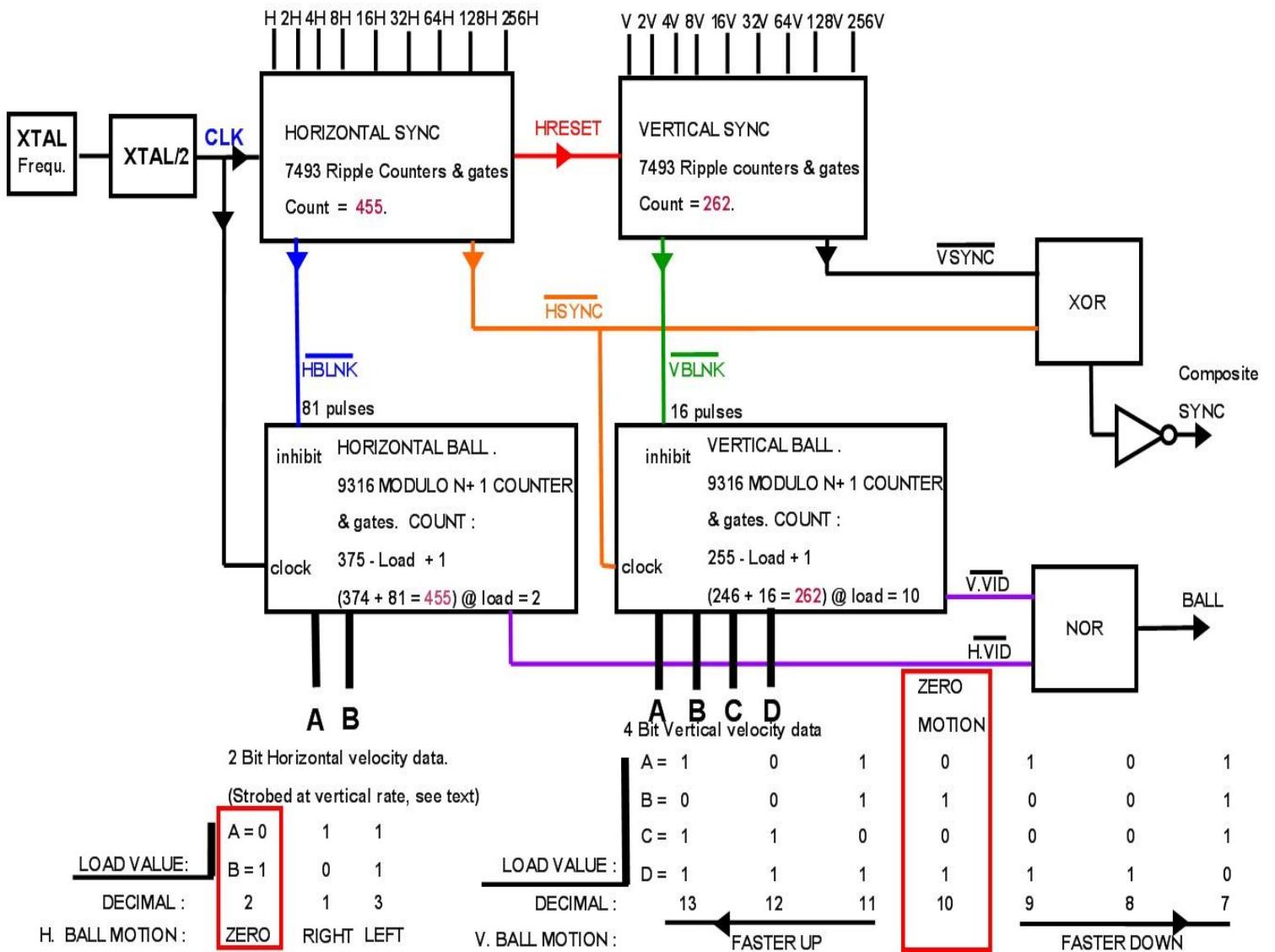
To slow the motion it requires that the horizontal ball motion be gated for short periods at the vertical rate.

The increments of gated time allowed for the ball to move are two horizontal intervals plus one or two added horizontal intervals to control the range of horizontal ball speeds. Also the horizontal motion is only allowed for these time periods once per vertical interval.

The three possible horizontal ball "speeds" are controlled by the number of hits (ball contacts with the bat) from the HIT Detector so the horizontal ball speed increases in a volley. The output from the HIT detector goes high when either player's bat is coincident with the ball signal (or ball video pulse) .This will be explained below.

Firstly a block diagram of how the ball position counters, which are loaded with the speed data and are linked to the sync generators:

VIDEO SYNC GENERATION & MODULO N +1 COUNTING & BALL GENERATION:



When the horizontal load value A=0 and B=1 (decimal value = 2) the ball would have zero horizontal motion. In practice although it may be zero for times the horizontal load value never stays on that number as the ball would be stopped in horizontal motion.

On the other hand the vertical load value 0101(A = 0, B = 1, C = 0, D = 1, = decimal 10) does occur for long period with a volley where the ball

is bouncing between the centre area of the two player's bats and the bats are perfectly aligned on the screen.

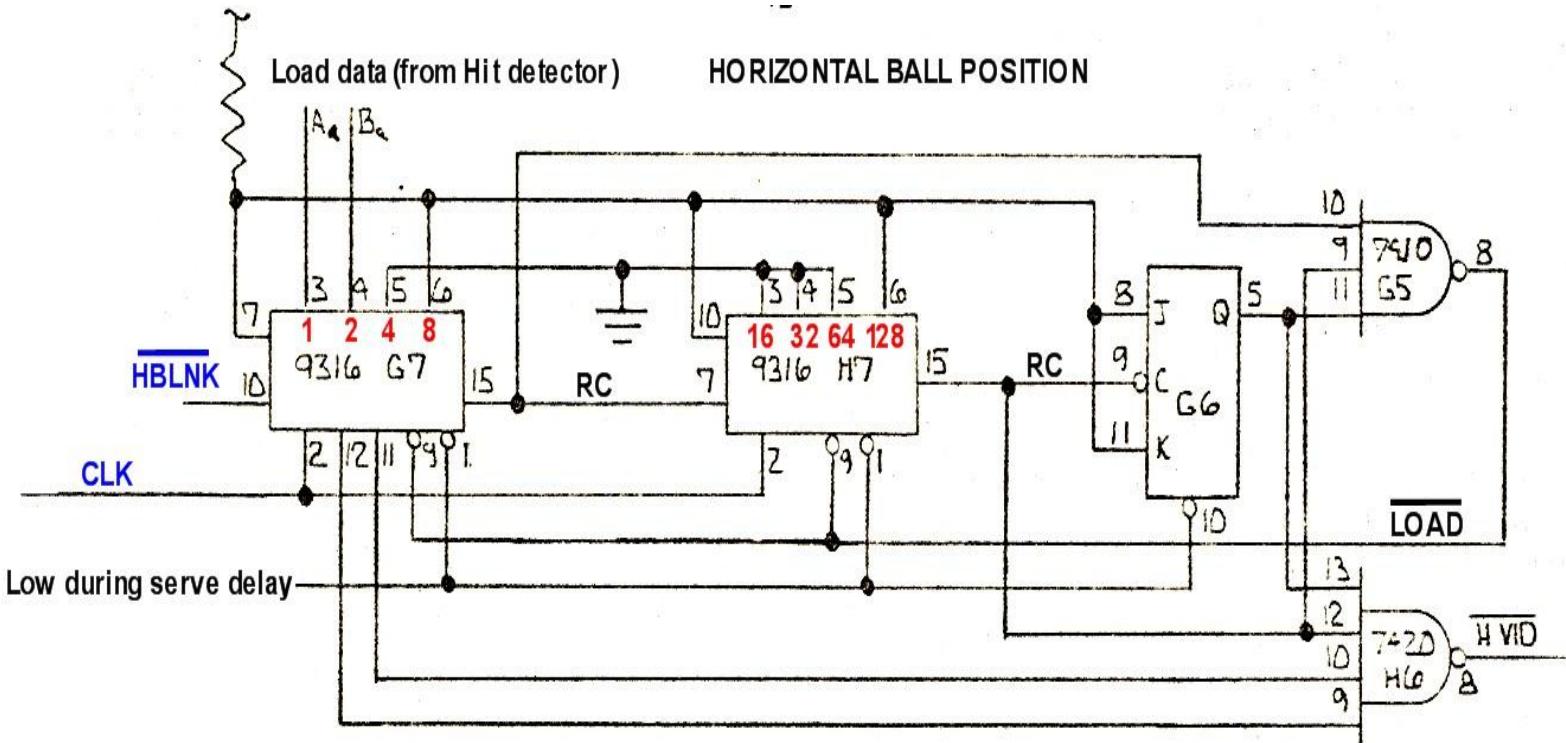
The vertical load values are generated by the position on which the ball hits the bat. The load values are also reversed (for example load value equivalent to decimal 12 flips over to that for decimal 8) when the ball hits the horizontal edge (top or bottom) of the raster (vertical blanking) so the ball reverses direction. This will be explained in the relevant sections below.

Ball position counter inhibition:

The horizontal 9316 is inhibited for 81 clock pulses and counting at the centre division rate of 364 with a load value of decimal 2 so that the overall count is 455 to match the horizontal 7493 sync counter.

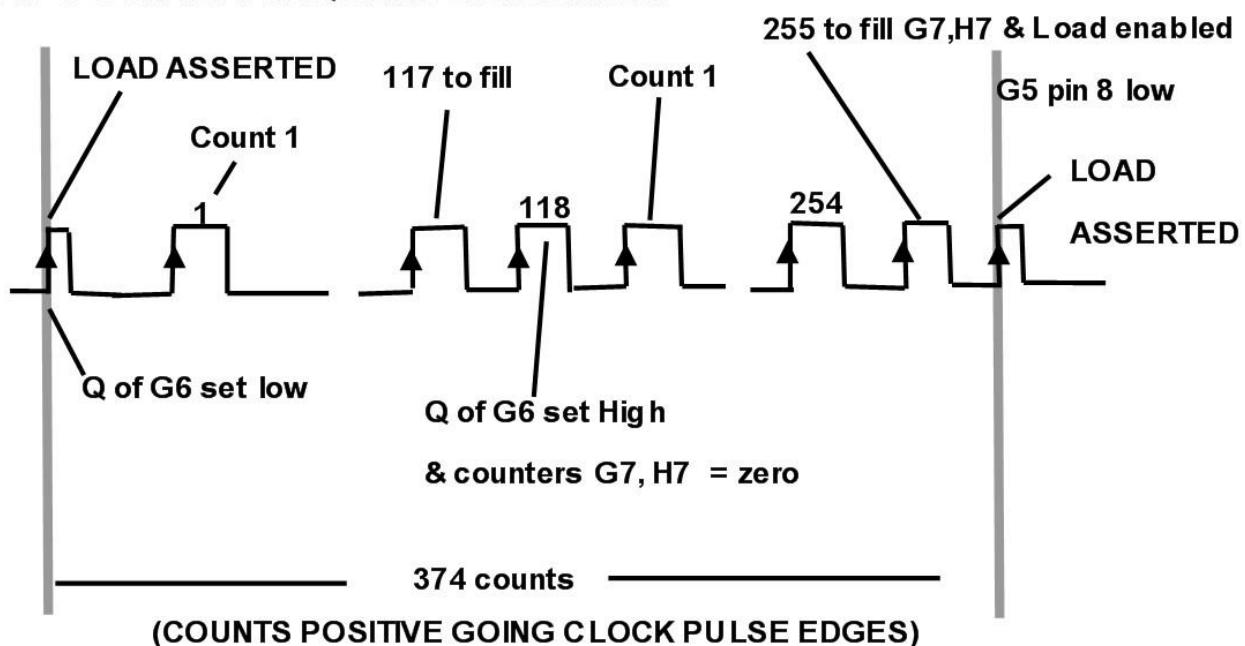
The vertical 9316 is inhibited for 16 (not)HSYNC pulses so that the overall count with a decimal load value of 10 is 262 to match the vertical 7493 sync counter.

The following circuits are the 9316 modulo $N + 1$ counters. They are called modulo $N+1$ because the total count (and therefore division rate) can be controlled by the load value N and the 1 relates to the process of loading the counters and asserting the load which uses +1 clock pulse:



It is best to show on a “sequence diagram” (not a timing diagram) how the *counts* add up as it is more involved than the simple 7493 counter system:

H 9316 COUNTER SEQUENCE DIAGRAM:



For the (not)load to assert the load pins 9 of the 9316 counters connected to G5 pin 8 need to be low. Then the load is asserted, synchronously, on the *next* clock pulse. These counters count the positive edge of the clock (CLK) pulses.

When the load is asserted (assuming A=0, B=1), G7 is loaded with 2 and another 8 (as its pin 6 is set high and B is high) and H7 is loaded with 128 as pin 6 is tied high. So to fill G7 and H7 the number of pulses required is $255 - (2 + 8 + 128) = 117$ clock pulses. This 117th pulse takes both the RC (ripple count) terminals, pins 15 of each 9316, high.

On the 118th pulse the RC output of H7 (&G7) falls low, clocking a high to G6 pin 5 and at the same time the counters G7 and H7 are zeroed as they have simultaneously overflowed.

The 9316 counters then count to 255 to fill on the usual way. At this point both the RC outputs are high again and because G6 pin 5 is still high *the load is enabled* in that the (not)Load on pin 8 of G5 is low. On the *next* clock pulse the load is asserted. Also the RC of each counter falls to zero. This toggles the flip flop G6 to have a low on its Q output (as it was previously high) and the 9316 counters are loaded as before and the process repeats.

(Note that when a JK flip flop has the J and K high it acts as a negative edge operated toggling flip flop)

The total clock pulse or count is therefore:

1 (for the load assertion) + 118 + 255 = 374 (when the two bit control data to the 9316 is A= 0 and B=1).

Due to the fact that the 9316 counter system is inhibited for 81 clock pulses by (not)HBLNK then the effective count interval, in terms of total counts with respect to the sync generator is $81 + 374 = 455$. This

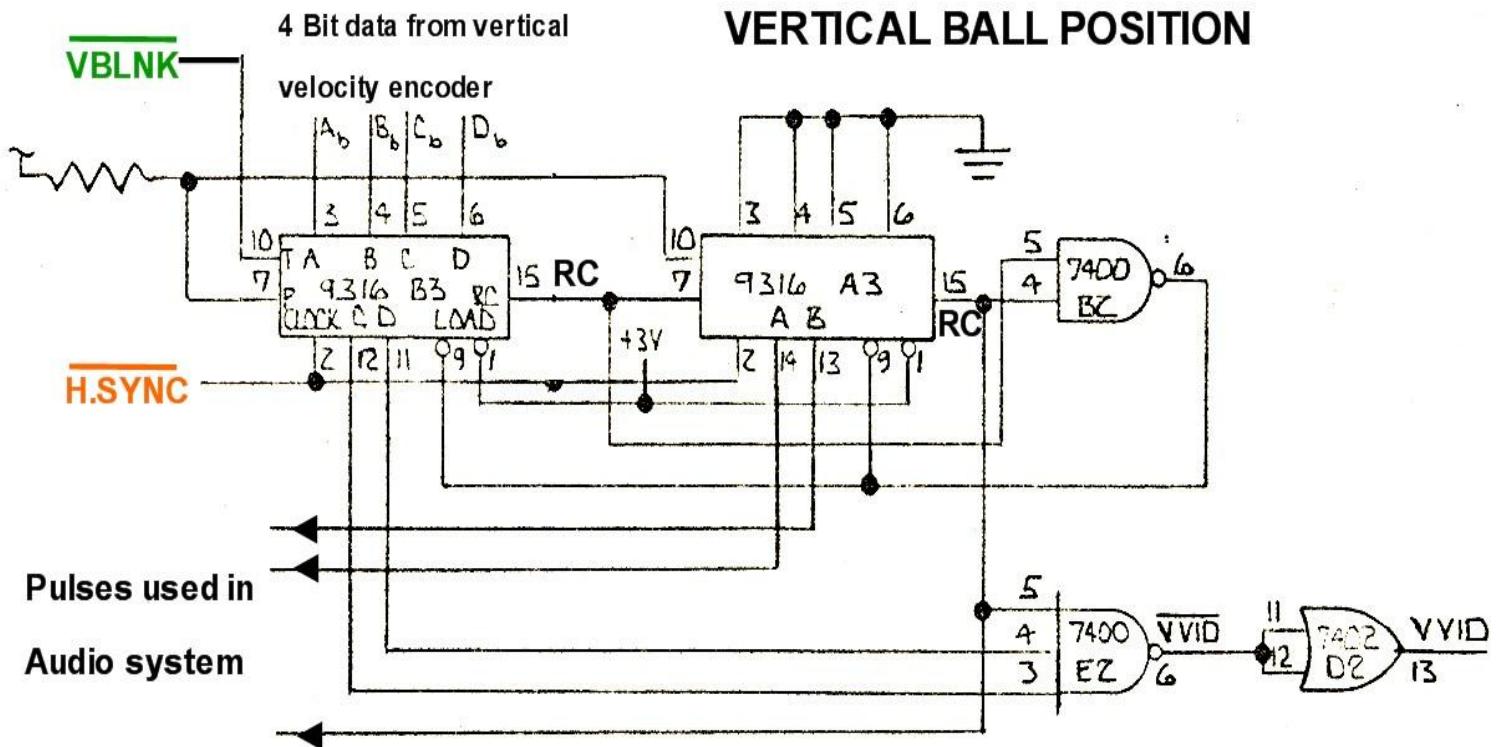
matches the division rate of the horizontal sync generator. So in this condition the horizontal ball video signal (not)HVID would have no motion with respect to the sync or video raster.

If the load value is A=1 and B=0 then the overall division value would be 456 and the (not)HVID frequency would be slower than the H sync pulses and the ball would be moving to the right on the screen (arriving late).

If A=1 & B=1, then the effective division rate is 453 and (not) HVID would have a higher frequency than the horizontal sync pulses and the ball would be moving left.

This system of generating a moving object on a video screen has sometimes been called the “slipping counter” method.

Similar processes take place in the vertical ball position circuit shown below, except that the flip flop is not needed because the total count is lower:



The total count for this vertical system is 255- LOAD VALUE + 1 which is 246 when the binary load value on the D, C B and A inputs is 1010 (has a decimal value of 10). How this comes about can be explained as follows:

The sequence of operation can be described from the count after the load. If B3 is loaded with decimal 10 (A=L, B=H, C=L, D=H) for example, the counter B3 counts another 5 pulses to fill (to a total of 15). On the next (6th) count B3 is set to zero(overflows) and the first flip flop's within A3 is set high “representing” a count of 16, but it is actually a count of 6 because B3 was loaded with 10 to start with. Then after $(255 - 16) = 239$ further counts, both counters B3 and A3 are full and both RC outputs are high enabling the load via the Nand gate B2 pin 6.

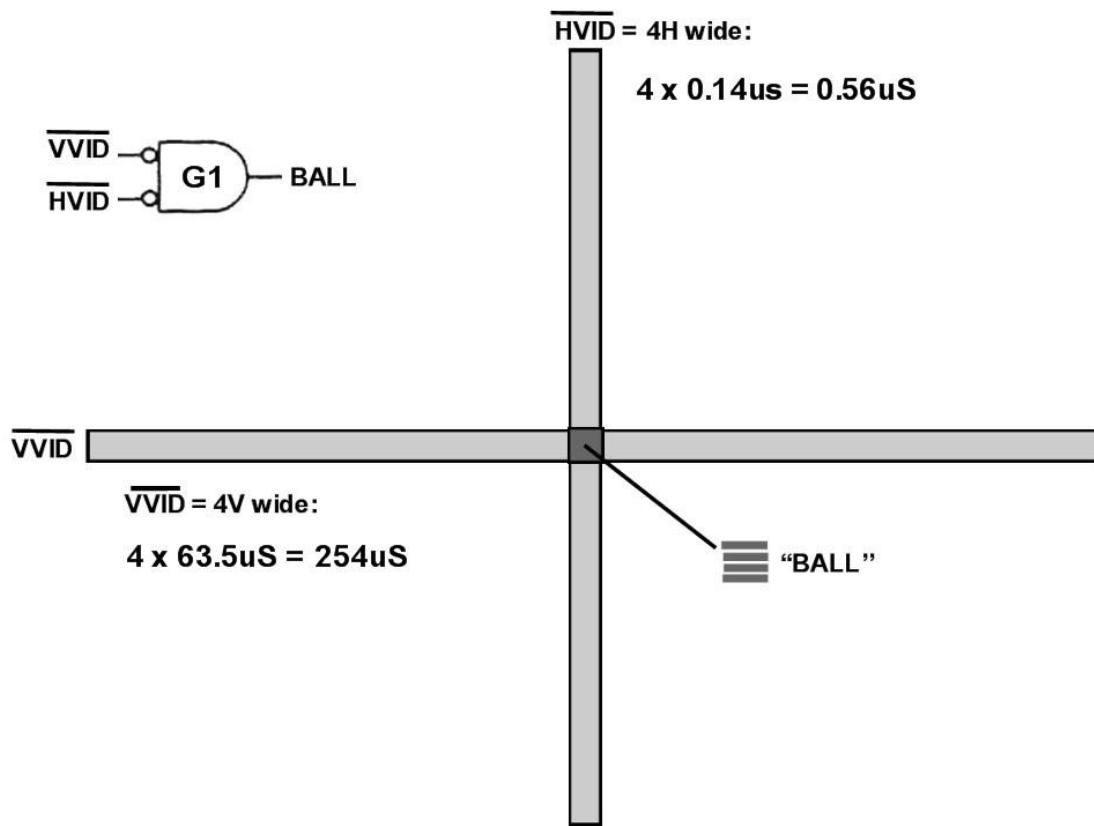
The load asserts on the next clock = (not) HSYNC. Therefore the total count is 239(capacity less 16) + 5(B3 fills) + 1(B3 zeros) + 1 (assert load) = 246.

Due to the fact that the (not)VBLNK inhibits the counter for 16 clocks (or (not)SYNC pulses in this case) the *effective count* is $246+16 = 262$ to match the Vertical sync rate.

Altering the load value around the value of decimal 10 controls the ball’s vertical velocity because the count is either a little higher or lower than the sync rate. The range of load values used is from 7 to 13 in Atari’s Pong and 6 to 14 in Chicago Coin’s cloned Pong. This control data is produced by the vertical velocity encoder driven by the bat & ball coincidence data which is one of the most interesting parts of the circuit.

VVID itself has a duration of four HSYNC’S (counts) as it depends on the C output of B3) or 4 lines of video or $4 \times 63.5 \mu\text{s} = 254 \mu\text{s}$ (which is the same width as a 4V pulse in the sync circuit). The video signal that

generates the ball from the screen image is created from the intersection of the (not)HVID and (not)VVID pulses. The ball is merely a short segment of 4 lines of video. The ball pulse itself is used in the HIT detector where the ball is coincident with the bats or paddles. However is the (not)VVID and (not)HVID pulses that are used to detect when the “ball” is coincident with blanking at the edges of the video scanning raster:



One interesting point to be emphasised here is that although both the vertical and horizontal ball position counters are inhibited during

counting over the vertical and horizontal blanking periods for 16 and 81 pulses respectively, the position of the (not)VVID and (not)HVID pulses, which comprise the ball, can appear in the respective vertical and horizontal blanking areas in the screen's video because at the moment the ball position counters are inhibited the ball signals HVID and VVID are HIGH. Therefore it can be detected when the ball enters these areas during those blanking times. This "elongates" the ball vertically right across the entire blanking interval. This property and some other factors relate to an intrinsic design issue in Atari's Pong which, under the right circumstances, causes the ball to become trapped around the vertical blanking period disabling the game. Information on this issue is in the modifications section.

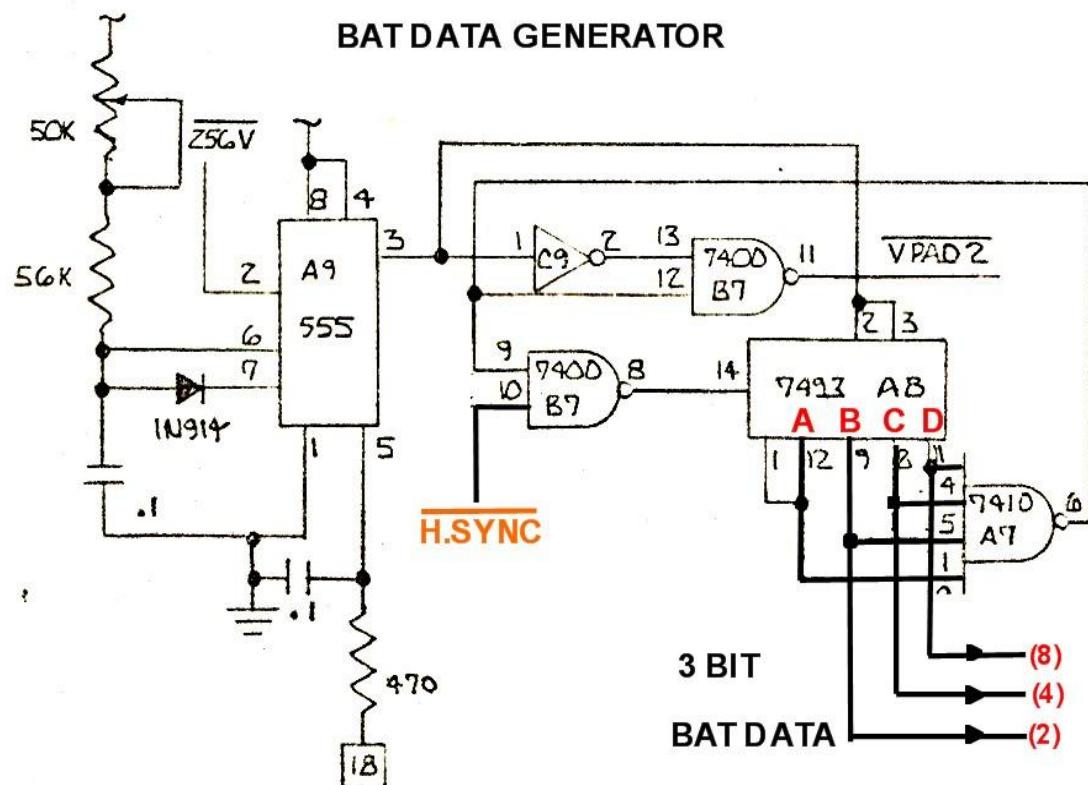
VERTICAL VELOCITY ENCODER:

To understand this system it is necessary to understand the bat generator. Each player's bat is composed 15 counts of the horizontal interval, each line having a count or a binary value which passes through an encoding circuit to control the ball's vertical velocity after the ball and bat are coincident. This data is clocked through the encoder circuitry to the load inputs of the 9316 in the vertical ball position circuit when the bat image and the ball image are coincident. This causes the ball to change its vertical velocity after it leaves the bat. Also the vertical velocity data is reversed (or inverted) when the ball hits the screen top or bottom so that the ball appears to bounce off these edges.

The bats are gated on to the video image in a specific horizontal position. (the bats can only move vertically).

Generating the BAT and the bat data from the players' potentiometers:

This is done with the circuit shown below. 3 bits of data are generated from a 7493 counter for each bat:



The two bat data circuits in the game are identical. (not)256V falls low at the beginning of vertical blanking. This triggers a 555 timer. This creates some delay (depending on the position of the player potentiometer connected across the 5 volt supply and to the 470 ohm resistor connected to pin 5 of the 555 timer IC A9).

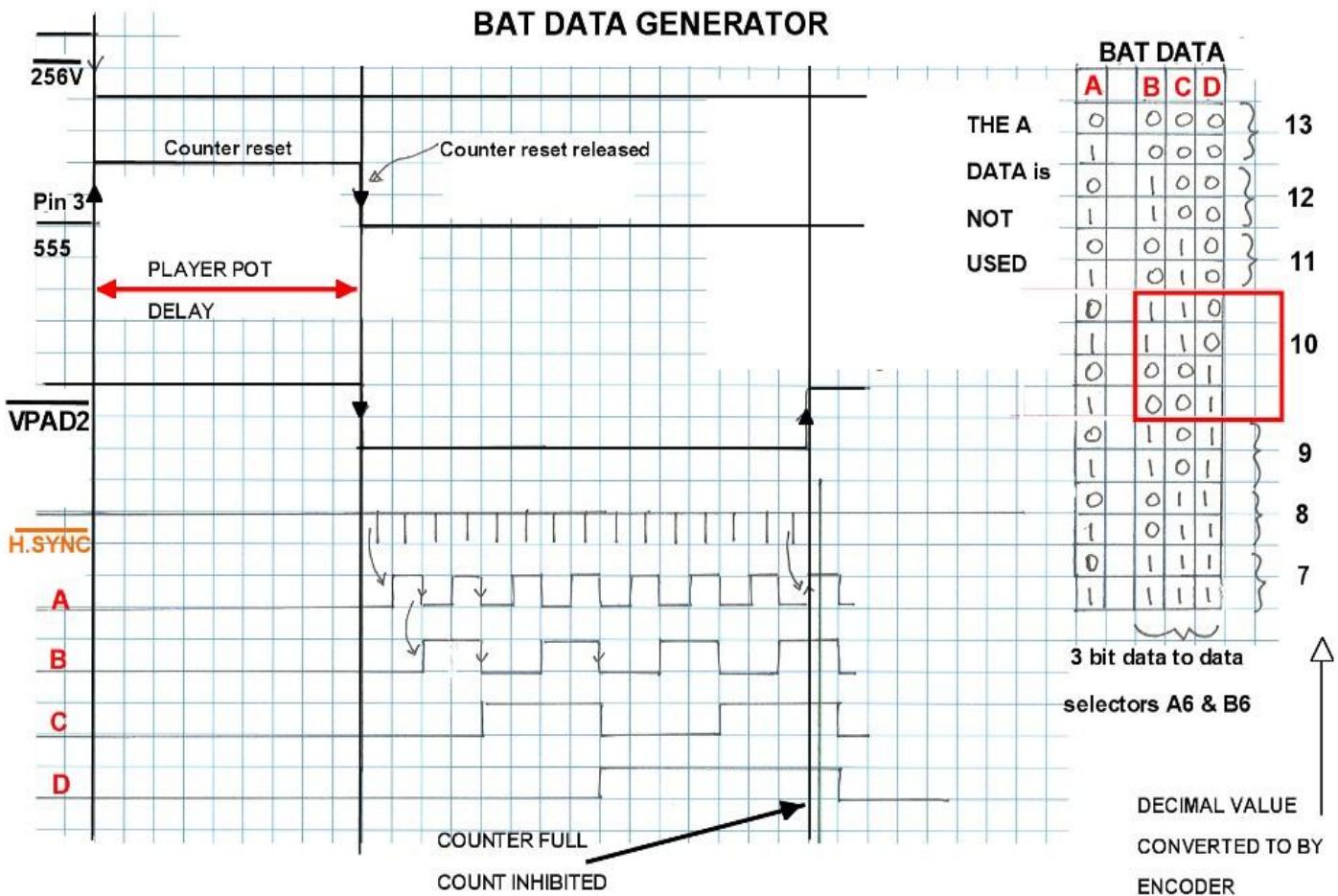
When the 555 timing period ends the reset (pins 2 and 3 of the 7493) fall low and the 7493 starts to count horizontal sync pulses. Also at the start of this count (not)VPAD2 goes low. The counter counts up to 15 and at

that time the output of A7 pin 6 goes low. This causes (not)VPAD2 to go high and also stops the count leaving the counter full.

Effectively 15 horizontal syncs (or lines) are counted, starting at some point on the scan after a player controlled delay from the start of the vertical blanking interval.

When the 555 is retriggered from the (not)256V the next time around and pin 3 of the 555 goes high again, this resets the 7493's outputs to zero and the process repeats.

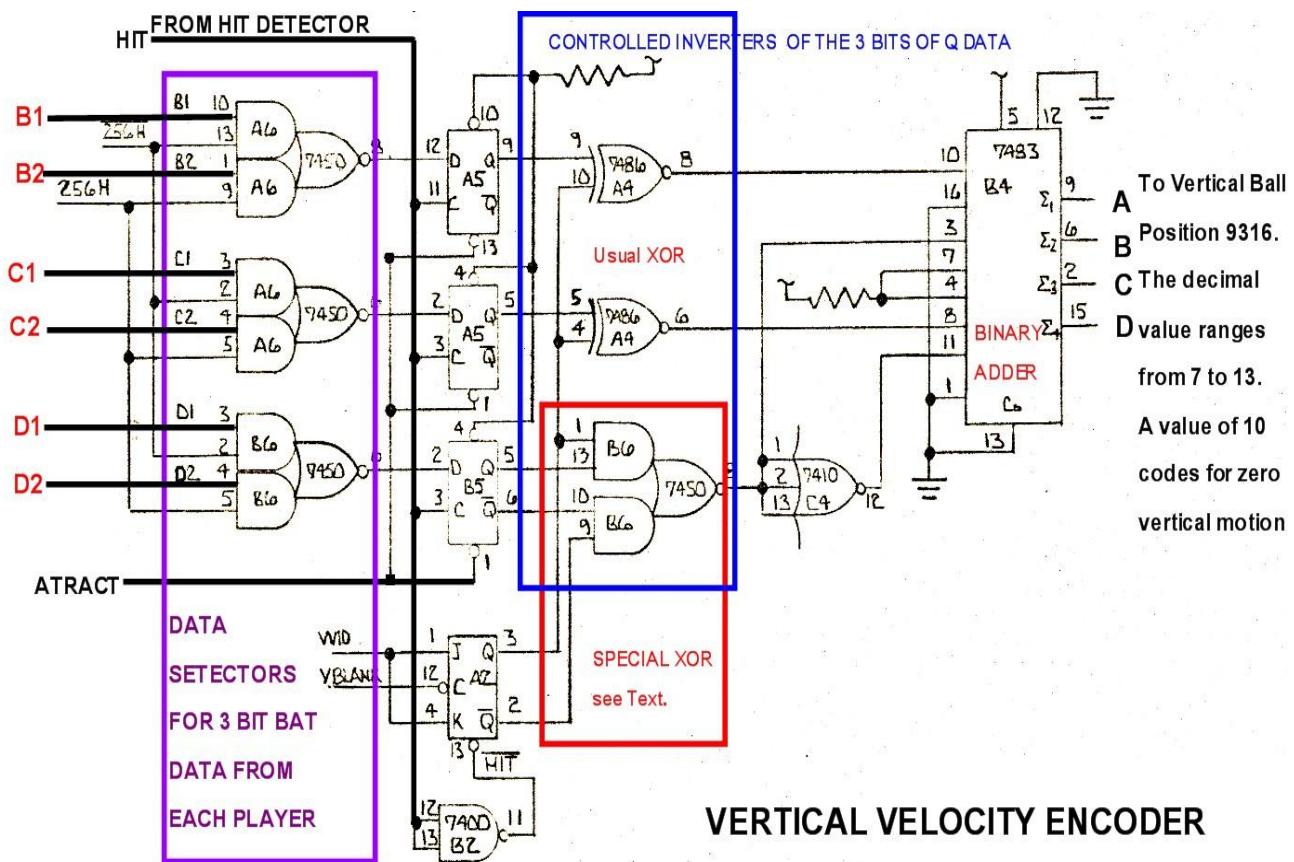
The least significant bit, from the counter's A output is not used. The binary values of the B,C & D outputs are ultimately encoded to a range of new values in the vertical velocity encoder.



All of the values shown in the table above in the red box are encoded to a value of 10, which is the value which when loaded into the vertical ball position 9316 circuit results in zero vertical ball motion. Therefore when the ball strikes the bat centre, which is the middle row of bat lines, the value of 10 is clocked through to the vertical ball position 9316 counter.

If the ball is coincident with the upper part of the bat (or earlier in the counting process) then either the values of 11, 12 or 13 is ultimately passed to the vertical 9316 which results in a progressively more rapid upward vertical component of the ball motion. If the ball is coincident with the lower part of the bat, the bat data count is higher. These values are encoded to 9, 8 and 7. When the ball is coincident with the lower bat, then 9, 8 or 7 is passed through to the vertical ball position 9316 and the ball motion is progressively faster downward.

The original Atari Pong vertical velocity encoder is shown below:



The 3 bit bat data, B, C & D from each players 3 bit bat data generator, labelled **B1, B2, C1, C2, D1 & D2** is initially passed to data selectors; a 7450 A6 and B6.

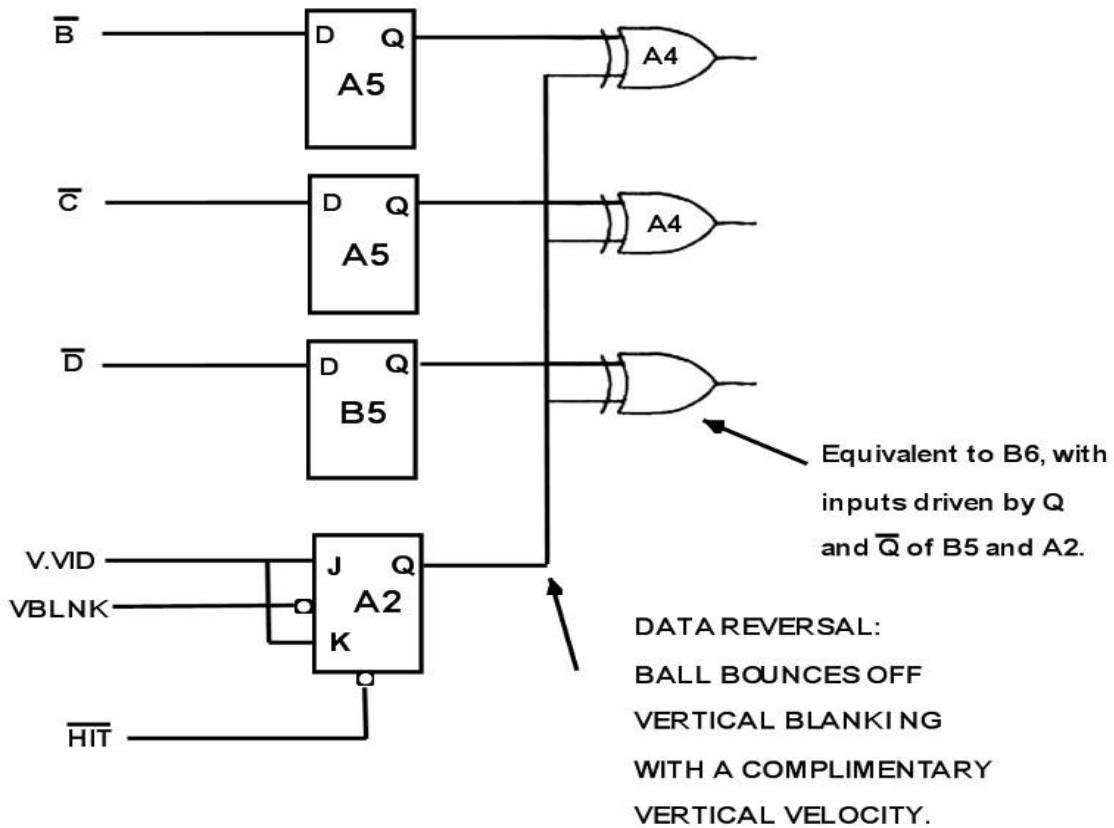
The (not)256H selects the left side of the screen image and 256H the right so this is how each players bat is selected from two identical player bat data circuits.

The 3 bit bat data, is inverted by the outputs of the 7450 data selectors and is connected to the D (data) inputs of the three 7474 flip flops, A5 & B5 to latch the data.

When either of the player's bats and the ball are coincident the HIT pulse (from the HIT detector) goes high. This clocks the 3 bit bat data (whatever it is at that moment) to the Q outputs of the three 7474 flip flops where the data is latched (stored).

The 3 bit bat data, which is now specific to either players bat passes via the XOR gates to binary adder IC 7483. The reason it is passed through the XOR gates is that the XOR, as mentioned, can be used to invert or not invert a logic level depending on the logic state supplied to the other XOR input.

Despite the complexity of the lower gate arrangement of B6 with the 7450 IC, it is merely an XOR. To re-draw the circuit using an XOR is more obvious what is happening here:



So simply the data values derived from the bat data and latched by the 7474 flip flops can be inverted, or not, depending on the Q output of the JK flip flop A2.

When the vertical component of the ball is coincident with vertical blanking (VBLNK) A2 is clocked when VBLNK falls low (at the end of blanking) and the vertical ball velocity reverses. For example 13 becomes 7, 12 becomes 8 or 11 becomes a 9 decimal value at the output of the binary adder IC. This makes sure that the ball “bounces off the vertical edge of the video raster” with *nearly* exactly the same vertical velocity value with which it approached it, making for a realistic bounce.

The (not)HIT pulse also clears the vertical motion reversing flip flop A2 to ensure the correct unaltered latched data is passed through the XOR's when the bat & ball "collide".

The ATTRACT input is used when the game is not in use. Its results in the ball being visible and bouncing around the screen and the player bats and scores are missing. It was done this way to attract people to the machine.

Looking at the 3 bit bat data passed to inverter gate C4 and to the input and then output if the binary adder is best shown in a table:

Vertical Velocity Bat Data Encoding, PONG E:

BATA DATA FROM 7493's	BAT DATA AFTER DATA SELECTOR 7450's	Decimal Value	7483 ADDER FUNCTION	ADDER FUNCTION WITHOUT C4	ADDER OUTPUT DECIMAL	ADDER OUTPUT DECIMAL	ADER OUTPUT BINARY to 9316
B C D	\bar{B} \bar{C} \bar{D}						VERT POSITION
0 0 0	1 1 1	7	6	7	13	14	1 0 1 1 0 1 1 1
1 0 0	0 1 1	6	6	7	12	13	0 0 1 1 1 0 1 1
0 1 0	1 0 1	5	6	7	11	12	1 1 0 1 0 0 1 1
1 1 0	0 1 1	4	6	7	10	11	0 1 0 1 1 1 0 1
0 0 1	1 1 0	3	6 + 1	6	10	9	0 1 0 1 1 0 0 1
1 0 1	0 1 0	2	6 + 1	6	9	8	1 0 0 1 0 0 0 1
0 1 1	1 0 0	1	6 + 1	6	8	7	0 0 0 1 1 1 1 0
1 1 1	0 0 0	0	6 + 1	6	7	6	1 1 1 0 0 1 1 0

+1

When \bar{D} = LOW, an extra 1 is added,
by inverter C4 pin 12

IF C4 IS OMITTED, AS DONE IN
CHICAGO COIN's PONG,
THEN A ZERO VERTICAL MOTION
STATE IS NOT POSSIBLE AS OUTPUT
VALUE OF "10" IS ELIMINATED

The binary adder is configured to add a value of 6 or 7 to the range of BAT data.

In Chicago Coin's Pong the gate C4 was omitted and this meant that the vertical ball motion was faster and the ball could never travel horizontally (with a zero vertical motion). This made the game a little more difficult and faster to play, simply by not using C4 (it was effectively replaced by a link).

Looking at the data, it is easy to see that if the bat data after the bat selectors (which is latched at the 7474 flip flop Q outputs) is inverted by the XOR gates that the table of output values will reverse, eg 13 would become 7, 12 become 9 etc and 10 not change. This reversal enables the ball to bounce off the top or bottom of the screen. This is achieved with the JK flip flop A2. A2 is clocked by VVID going low and toggles when the J & K inputs are high at the moment vertical blanking pulse (VBLANK) goes low.

There were three problems in this bat & ball control system:

1) Inadequate BAT or Paddle Range:

The range of bat motion provided to the player, even when ideally set on its preset pot, was limited so that there was a gap above and below where the ball could pass by, out of range of the player's bat. This could frustrate some players and encourage them to force the player controls. This can be improved with a simple modification (see modifications section). However reducing this gap increases the probability (unless rectified) of the problem cited in 2 below with the ball trapped in the vertical blanking area. This is the likely reason why the designer set the

bat range limited to where it was. Less gap between the bat and vertical blanking rate causes a faster bounce rate, if this reaches the vertical rate, the ball becomes locked around blanking, even when the bat moves away due to the A2 flip flop configuration and other factors to be discussed below.

One other theory, which I don't subscribe to, was that the limited bat range was "deliberate" meant that two super experienced players could not play the game indefinitely. This is less likely given that the designer would have experienced the lockup problem during R&D and that the game was made for "drunken players in a bar scenario" and likely the players would be missing the ball quite a lot especially as the horizontal ball speed increased in a volley and their responses probably would be disordered by alcohol.

2) Ball Trapped Around Vertical Blanking:

If the bat (paddle) range is either modified for a full range, or out of adjustment on the PCB preset, the bat is able to "push" the ball into vertical blanking. In some cases it can become "locked" there, even after the bat moves away, disabling the game play and requiring the game be power re-booted. This scenario is caused by the fact that the VVID pulse becomes elongated over vertical blanking & straddles the vertical blanking pulse. The widened vertical component of the ball oscillates around vertical blanking (rather than the raster area) and is unable to "escape". This "Lockup" is most likely the primary reason why Atari's Pong E had a limited bat range despite other stories behind that. Details on this design anomaly and the cure for it are explained in the modifications section.

3) Ball pulse (video signal) inside Vertical Blanking and Sync interval:

This can upset the vertical lock on the monitor a little again discussion & cure is in the modifications section.

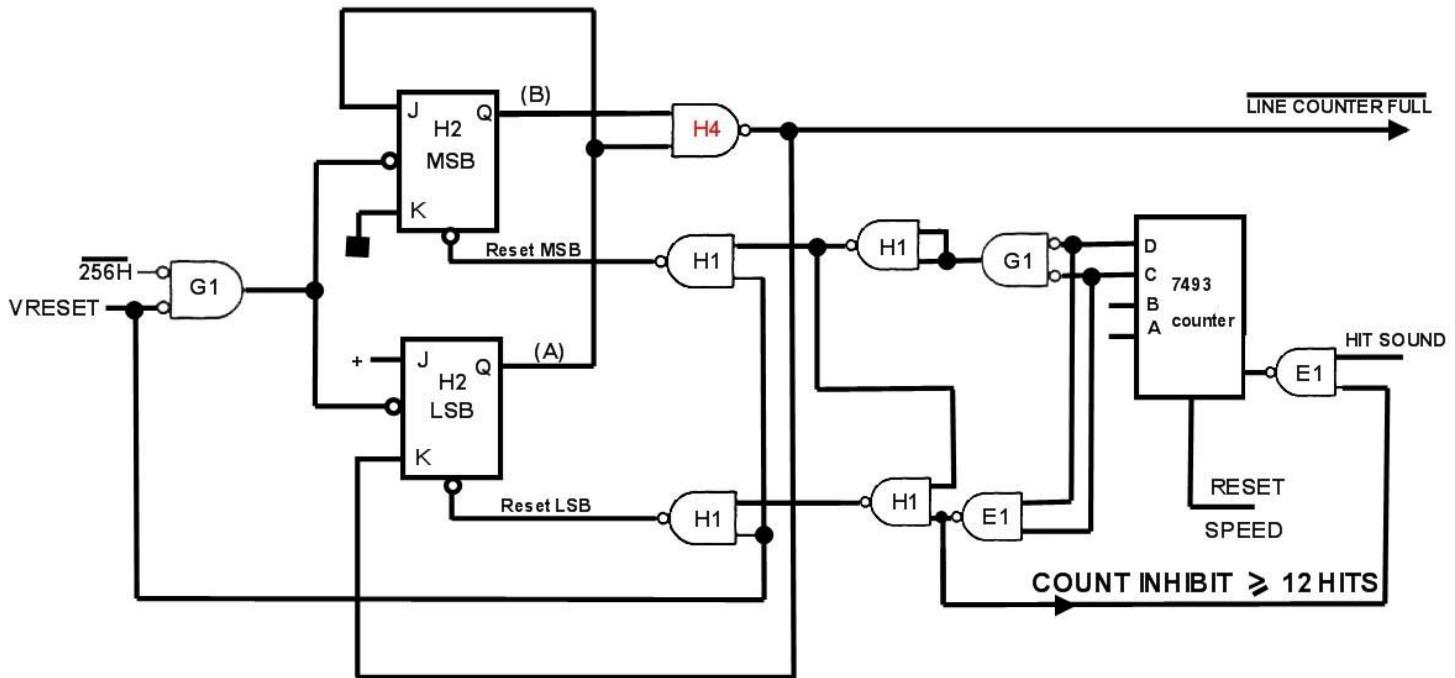
HORIZONTAL BALL VELOCITY ENCODER:

This is a 2 bit controller unlike the 4 bit controller for the vertical ball control. As mentioned the 2 bit horizontal data must be controlled at a lower frequency rate than the horizontal rate and the data values sent to the horizontal 9316 ball position & motion circuit that encode R and L ball motion are only present for brief periods of time at a vertical rate.

Binary load value 2, or A=0, B=1 codes for zero motion, A=1, B=0 (decimal 1) codes for motion to the right and A=1, B=1(decimal 3) codes for left motion. It is the time that the data spends on value 1 or value 3 that determines the right and left speed of the ball. The longer the 1 or 3 data is present then the faster the horizontal motion. This time period is gated at vertical rate, or once every 1/60 second and the time periods allowed are two horizontal periods (127us) plus one or two added 63.5uS periods depending on the total number of Hits recorded by the HIT counter. When there is a miss the ball passes by the players bat and hits HBLNK (horizontal blanking) and the Hit counter is resent.

A two bit counter made from two JK flip flops (H2) is set up to count vertical reset pulses and each bit of the counter, the LSB and the MSB can be independently reset. This 2 bit counter spends most of the time jammed full and can only count because of the resets:

HORIZONTAL SPEED CONTROL: HIT CONTROLLED LINE COUNTER



The schematic of the HIT controlled line counter is shown above. The counter counts horizontal timing from the (not) 256H pulse and is reset by VRESET.

This is a counter where the LSB and MSB jams full (a high on the Q outputs) and the number of possible horizontal counts depends on whether the LSB, or MSB, or both are reset(cleared) by the VRESET pulse. (the G1 & H1 combination form an OR gate).

2 bit counter value after VRESET :

LSB (A)	0	1	1	0
MSB (B)	1	0	0	0
	HITS	HITS	HITS	
	< 4	4 to 11	12+	

Looking at the count values in the table above after the reset, with hit count < 4, then one count fills the LSB (MSB already full).

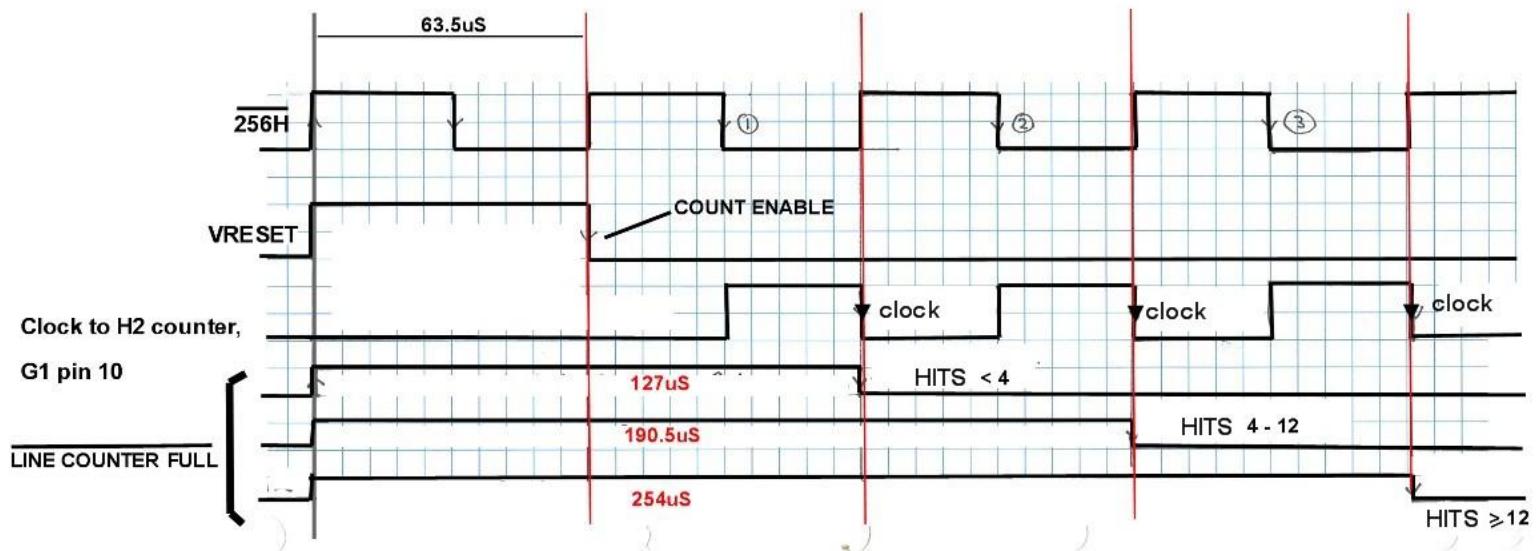
A hit count 4 to 11 it takes two counts to fill the MSB & LSB and it takes 3 counts to fill the LSB and MSB when the hit count is 12 or more. However it doesn't get to more than 12 in the 7493 counter because the count is inhibited at score 12. Put another way:

Once the < 4 hits (C& D low on the 7493) the LSB is reset(cleared to zero) by VRESET, so it takes 1 horizontal count to fill the counter pair. Hits 4 to 11(C or D) high the MSB is reset so it takes 2 counts to fill the counter pair. Hits 12 or greater (C& D High) the LSB and the MSB are reset, so it takes 3 counts to fill the counter pair.

It is best to look at a timing diagram below to see how the (not)LINE COUNTER FULL pulse is generated: The (not)LINE COUNTER FULL pulse controls the time that the ball is allowed to move horizontally.

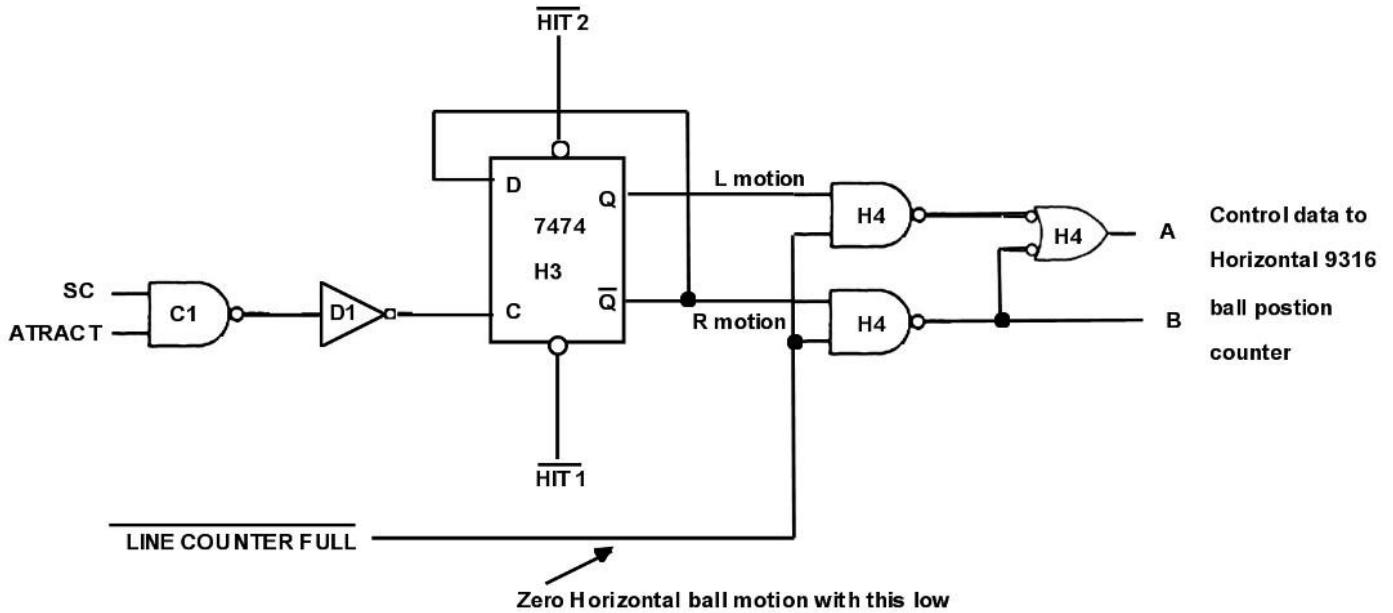
The graph below shows how the length of the (not) LINE COUNTER FULL pulse varies with the HIT count and how it is synchronised to the VRESET pulse. The allowed times are 127uS, 190.5uS and 254uS which are two, three and 4 horizontal intervals:

HORIZONTAL BALL SPEED CONTROLLER:



The (not)LINE COUNTER FULL which enables the horizontal ball movement is passed to a gate system to facilitate reversal of the 2 bit data that the loads to 9316 horizontal ball position counter. This is done so the ball's horizontal motion is reversed when the ball strikes the player's bat:

LEFT / RIGHT MOTION FLIP FLOP.



The (not)HIT 1 and 2 pulses are the players independent signals when their bat contacts the ball, whereas HIT and (not)HIT are the two combined.

The A& B data to the horizontal ball position 9316 reverses with a hit which makes the ball bounce off the bat with the opposite horizontal direction with the hit.

The SC pulse (from the Score Sound 555 timer) is produced when a ball misses the bat and hits horizontal blanking (HVID and HBLNK are coincident to trigger this event). The SC pulse is also used in the sound circuit.

When the machine is not in use, the ATRACT is high and the ball bounces around the screen. The SC pulse ensures the ball reverses direction on hitting horizontal blanking by clocking H3.

The (not)HIT 1 and (not)HIT 2 levels are zero as the bats are disabled in ATRACT mode and the bats therefore are not shown on the screen in the ATRACT mode.

During a game, the R and L motion outputs are also used to direct (gate) the (not)MISS pulse to the correct players score counter, so each player gets a score count that belongs to them.

Possible ranges of ball motion in Pong:

The right and left horizontal speeds as noted are not exactly the same, but both very close in opposite directions. Likewise the possible ball motion up and down with load values of 13 and 7 respectively are not exactly identical but very close.

We could summarise the ball's motion as a vector, having both a direction and a speed.

There are 6 different horizontal velocity vectors, three to the right and 3 to the left. There are three vertical speeds upwards and 3 complimentary speeds downwards and one state of zero vertical motion.

Calculating the velocity vectors of Pong:

This can be done by calculating the frequency difference of the ball(VVID component of it) compared to the sync pulses. For the vertical system it is quite straightforward. The table below shows the

calculations for the possible vertical speeds normalised to a 3 meter high raster(4 x 3 m screen):

LOAD VALUE	Vertical Ball Frequency Hz	Sync Frequency Hz.	Δf Hz = screen heights/second	X 3 Meters
13	$15731/(243 + 16) = 60.737$	60.042	- 0.695	2.08
12	$15731/(244 + 16) = 60.504$	60.042	- 0.462	1.39
11	$15731/(245 + 16) = 60.268$	60.042	- 0.226	0.678
10	$15731/(246 + 16) = 60.042$	60.042	0	0
9	$15731/(247 + 16) = 59.814$	60.042	0.228	0.684
8	$15731/(248 + 16) = 59.587$	60.042	0.455	1.36
7	$15731/(249 + 16) = 59.362$	60.042	0.680	2.04

Speed
M / sec

If the vertical sync pulse (or rate) is used as a frame of reference and for example if a some pulse frequency was 1 Hz difference from the sync frequency, then this pulse would “pass by” the sync pulse on the video display once per second, or cover a distance of one screen height per second. Therefore, multiplying the frequency difference by an imaginary screen height, say 3 meters, gives the vertical speed of the ball if it were displayed on a 1 meter height screen. Load values 11 to 13 the ball is moving up, and 9 to 7 moving down.

Notice that the complimentary up vs down speeds are almost identical so they can be considered to be the same for practical purposes. The Δf values with a negative number indicate the ball is travelling upwards.

Counting digitally to compare: The other way to work the above table out is to calculate it on the basis of counts rather than frequencies. For example at load value 13 there is a difference of three counts in the ball position counter for each vertical interval with respect to the vertical

sync counts. Lets say the load value is 13 and the leading edge of the ball is sitting at a count of 100 on one field and on the next field the leading edge of the ball is on a count of 97 (has moved upwards).The counts between the two ball positions, one field later, is $262 - 3$.

Likewise if the ball was moving down the counts between positions, one field later, would be $262 + 3$. Multiplying these counts by 63.5uS gives the *time* between the two ball positions as there is 63.5uS between each count. The fraction of screen height moved is the number of counts moved, 3 in this case, divided by the total screen height (equivalent to 262 counts, blanking included) times the hight of the monitor screen, 3m chosen for the example giving the *distance* travelled.

Therefore using a pulse counting model, while moving upwards with a load value of 13 the normalised speed is:

$$3/262(\text{height}) \times 1/((262-3) \times 63.5\mu\text{s}) \text{ (seconds)} \times 3 \text{ meters} = 2.08 \text{ m/s}$$

Going downwards with load value 7:

$$3/262(\text{height}) \times 1/((262+3) \times 63.5\mu\text{s}) \text{ (seconds)} \times 3 \text{ meters} = 2.04 \text{ m/s}$$

This agrees with the calculations based on frequency differences.

(If you wanted to know the ball vertical velocity values for any particular sized screen, simply measure the height in meters and multiply the screen heights/sec value in the table by that number).

To calculate the possible Horizontal ball speeds:

As noted in the horizontal ball speed controller the ball has 3 possible basic horizontal speeds. Looking first at the Δf values for the ball (HVID component of it). With a load value of one, the counts for the horizontal

9316 position counter are 375, and with a load of 3 the counts are 373, and in each case the effective counts, in terms of frequency division are 81 more in each case due to blanking inhibition.

Therefore the horizontal ball frequencies are:

$$7.15755 \text{ MHz} / (375 + 81) = 15696.4 \text{ Hz} \text{ (ball going right)} \Delta f = 34.4 \text{ Hz}$$

$$7.15755 \text{ MHz} / (373 + 81) = 15765.5 \text{ Hz} \text{ (ball going left)} \Delta f = -34.7 \text{ Hz}$$

$$7.15755 \text{ MHz} / (374 + 81) = 15730.8 \text{ Hz} \text{ (sync frequency)}$$

& zero ball motion)

So the Δf , though not exactly identical for R and L motion is close to 34.5 Hz. This would mean the ball with this value would be crossing the full screen horizontally 34.5 times per second if it were not slowed down by allowing it to move only for limited time periods every field: The time allowed for motion is dependent on the hit count:

HITS	HORIZONTAL time allowed for movement	VERTICAL time allowed for movement	Fraction of time per field $\times 10^{-6}$	$X \Delta f$ ($\Delta f = 34.5 \text{ Hz}$ screen widths/sec)	Normalised to 4m wide	H BALL SPEED m / Sec
<4	127 μs	1 / 60.04 s	7625.1	0.26	x 4	1
4 to 11	190.5 μs	1 / 60.04 s	11437.6	0.39	x 4	1.6
12+	254 μs	1 / 60.04 s	15250.2	0.53	x 4	2.1

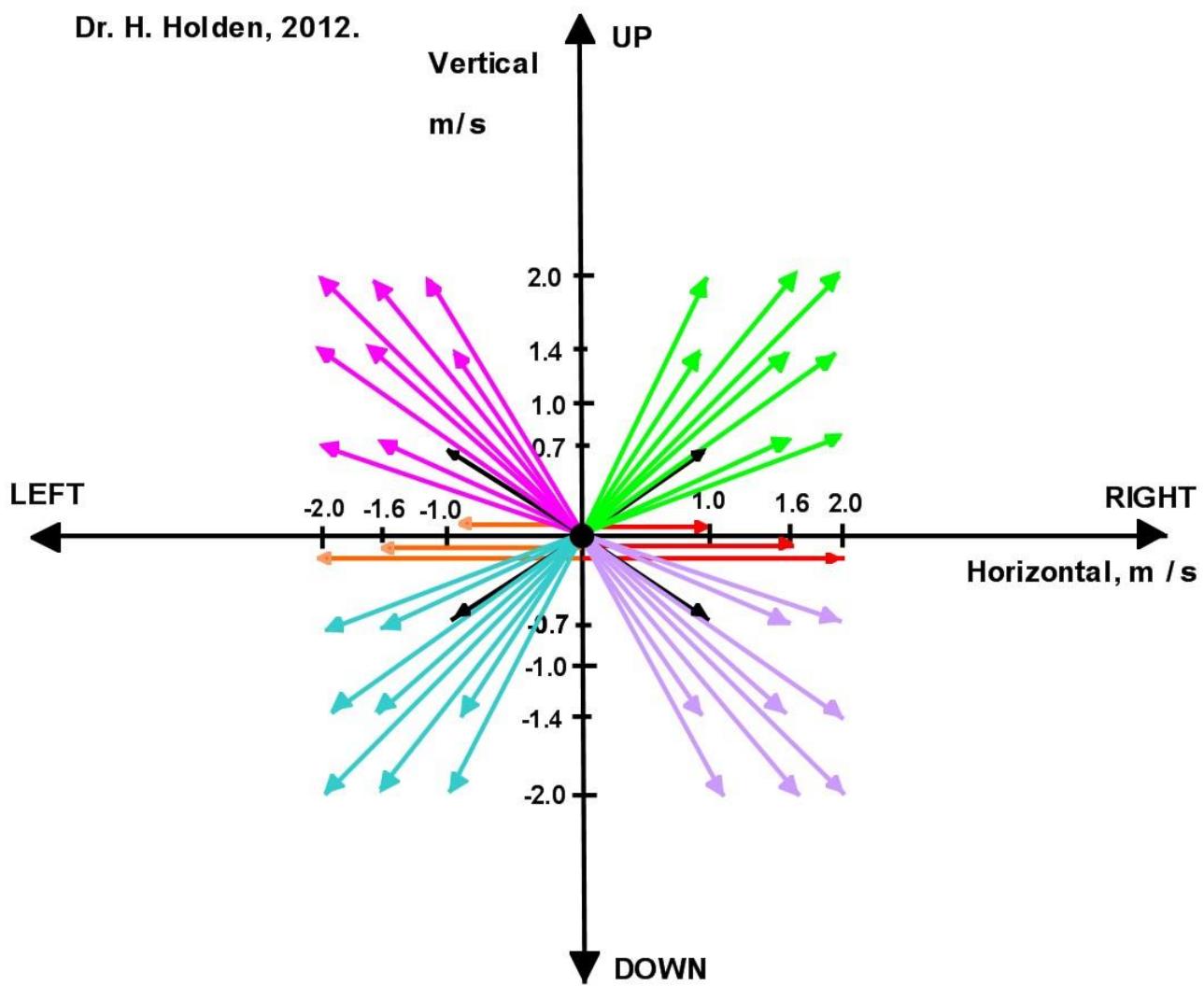
As can be seen the possible horizontal ball speeds are 1, 1.6 and 2.1 meters per second normalised to a 4 meter wide screen. This range of

horizontal speed is not dissimilar to those in the vertical system, where the numbers can be rounded to 0.7, 1.4 and 2 meters per second up or down. So the ball is also moving twice as fast horizontally in a good volley (HITS 12+) as when the game started.

To summarise this data on a vector diagram:

THE 42 VELOCITY VECTORS OF ATARI PONG:

Dr. H. Holden, 2012.



Pairs of vectors of different magnitudes lie nearly right on top of each other so the shorter one was drawn in black. The three right and left motion vectors (six total) which can occur when the vertical velocity is zero are drawn shifted away from the x axis so as to see them. As can be seen there are 9 vectors in 4 quadrants = 36, plus the 6 horizontal vectors making the total possible ball vectors a total of 42.

Again the actual speed would depend on the size of the monitor or VDU, the calculations for simplicity were normalised for a 4 x 3 meter raster so they can be easily found for any sized VDU.

(The total number of 42 has some interesting meanings in popular culture, that it is: “the answer to life, the universe and everything”, but this answer had limited utility because nobody knew what the question was, but now we know; it is the velocity vectors of Pong!)

Some additional interesting points on BALL motion:

In this text below referring to the “ball position” I refer to the leading edge of the ball to locate it, and not the ball’s height or width.

One might first think that since in this digital system because the ball can only move a fixed number of counts, say up or down say 1, 2 or three counts on the screen every vertical period, that the speed at which the ball would move upwards should be identical to the speed it would move down downwards (and right should equal left motion too) as the positions of the pulses are always a fixed distance apart and two ball positions are a fixed distance apart on the screen image which has a fixed timing.

As seen from the data above the complimentary right vs left or up vs down motions are *not exactly identical*, albeit very close. The reason for this is the relative timing of the counting sequence.

When the ball moves upwards (or left) it creates the illusion of moving back in time arriving earlier and earlier in the video image which is scanned from top to bottom (and left to right). On an oscilloscope locked to the vertical sync for example, the ball pulse on channel 2 is seen moving to the left which makes it look like its going back in time compared to most signals seen on a scope as the scope is scanned from left to right in time.

For example if the VVID (or vertical component of the ball) moves downward say 3 clock pulses, or 3 horizontal intervals, it will be sitting three scanning lines below the position it was previously in.

For this to have occurred the *effective* count in the ball position circuit must have been 262+3 over a vertical interval (which is 262 sync counts) so the ball moves to a point three counts after the position it was previously in. The term effective count was used again because the actual count of the vertical ball position circuit would be 246 +3 as blanking accounts for 16.

For it to move upwards 3 counts, or 3 lines, the vertical ball position counter would have to effectively count 262 - 3 counts so on the next field interval the ball is back where it was, in the original position, three counts above.

Looking at the distance the ball has moved up and down 3 counts or three horizontal lines, a fixed physical distance on the screen, spaced by a fixed time period of the scan, one would think that it would have taken the same time to move both ways, but this is not so!

The explanation for this is that the forward time or count progression of the scan (sync system) is *relatively* different while the ball is moving up or down over its fixed pulse positions. It's a classic relativity problem created by the fact that the ball changes position with respect to a progressive field of 262 counts or 262 horizontal periods or lines.

If the ball is in any screen vertical position, it will be a fixed number of counts, say any number like 100, below or after (not)vertical blanking goes high and the 9316 counter begins counting. It therefore takes 100 counts of horizontal timing pulses to generate that particular ball position. Also to get to the next (not)vertical blanking pulse going high(which enables the vertical ball position 9316 counters at the start of the next field again) it takes $262 - 100 = 162$ pulses of horizontal time from where the ball is currently located to start a new count.

If the ball moves downward, say 3 counts during a field (at its max downward speed load value 7) it will be sitting at count 103 after vertical blanking. So it takes a total of $162 + 103$ pulses of time to finish positioning the ball down 3 counts down the screen. This is $262 + 3 = 265$ counts or simply one field + 3 counts to go downwards 3 counts. The ball is now on count 103.

Then to move the ball back to where it was, 3 counts above, the 9316 counter is loaded with 13. But this time it takes $262 - 103 = 159$ pulses of horizontal time to get to (not) vertical blanking going high and starting the count in the vertical 9316 counter, then another 100 counts to position the ball, a total of $159 + 100 = 262 - 3 = 259$ counts.

So the time taken to position the ball upward is 259 counts, but to position it down the same amount is 265 counts. So the ball moves a little faster upward than downward. This fact was also discovered on the

ball velocity tables where *frequency differences* were used for the calculation.

Ball Movement In Summary:

*The time interval to move the ball up or down (or left or right) is **not** the time or space that exists in the scan interval **between** the two ball positions on the video image as one might intuitively think, but rather the time referenced to the scan itself **around** the two ball positions which is the 262 progressive pulse counting system of the vertical sync generators or the 455 pulse system of the horizontal sync generator.*

COUNTING & DISPLAYING THE SCORES IN PONG:

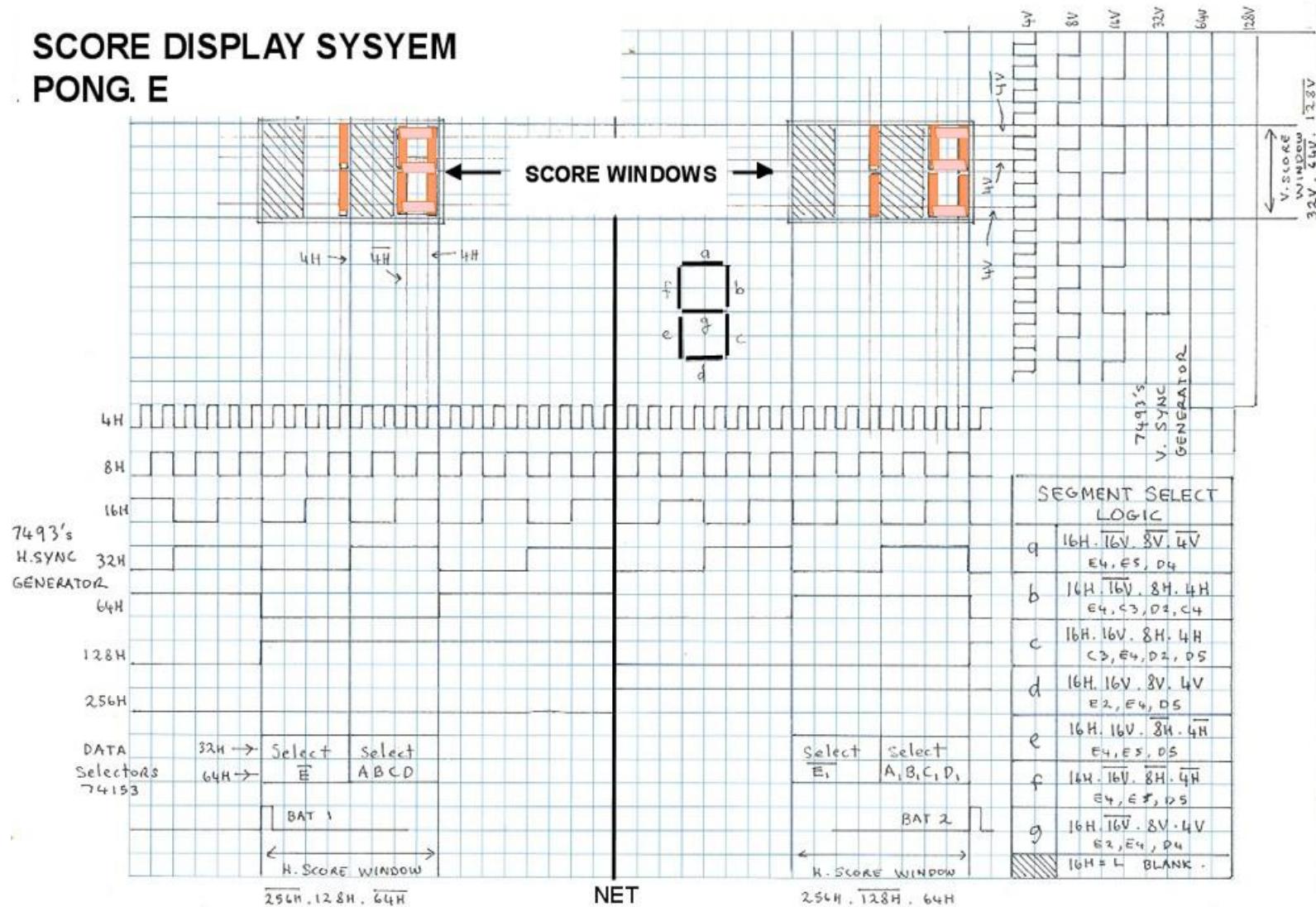
The scores are counted by two 7490 decade counter IC's. Attached to each one is an additional flip flop to provide an additional data bit to create the "1" in the tens column of the seven segment display that makes up each player's score number.

The segments that compose the 7 segments and the two segments that make up the "1" are gated or multiplexed via data selector IC's and a "Gate Array" into the score window area.

Probably it is best to look initially at the way the pulses which make up the score segments are gated into the video.

The chart below show how these are gated in. It is done with a gate array which responds to specific H and V pulses from the H and V sync generators. In general each displayed vertical segment is 4H wide (the same width as a bat) and the horizontal width of the segment is 16H.

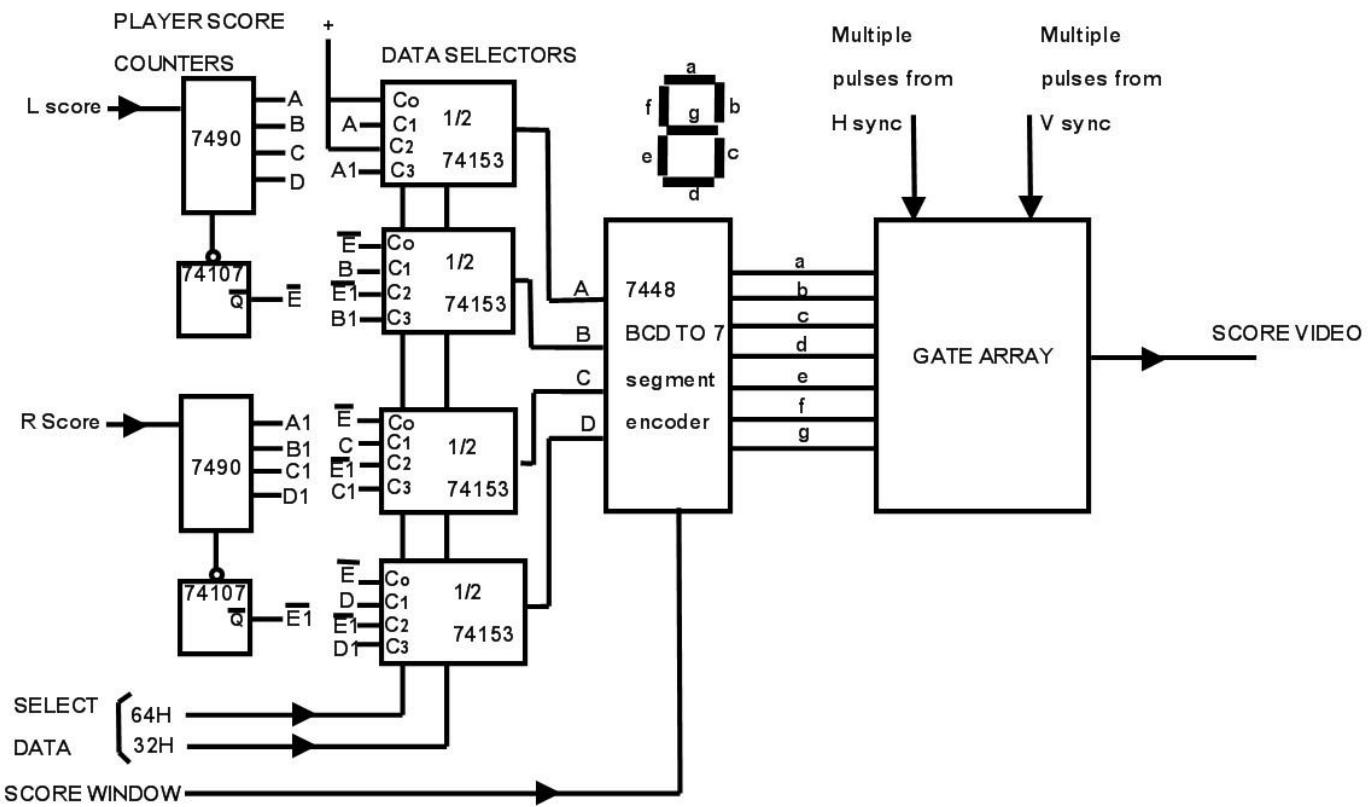
SCORE DISPLAY SYSTEM PONG. E



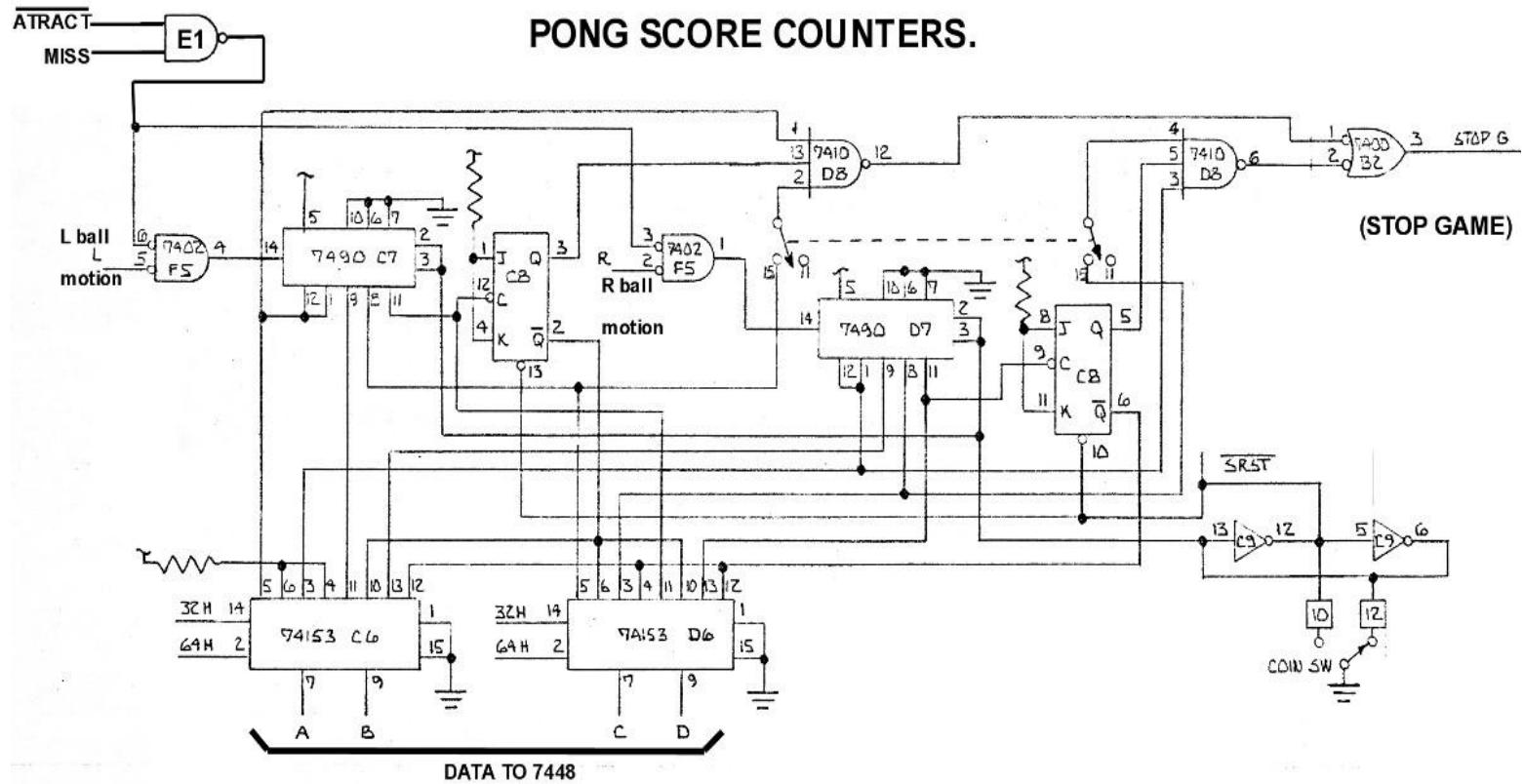
The 64H and 32H pulses are used to gate in data from the score counters into the data selector IC's (two 74153's).

How this is done is shown in the next two diagrams. The block diagram helps to show how the 74153 data selector IC's are configured with the 7490 counter system which is not immediately obvious from the schematic. The count of either player scoring 11 or 15 (depending on a pcb switch) stops the game. The (not)SRST comes from the game latch circuit, which goes low briefly when the coin passes the mechanism.

SCORE VIDEO GENERATOR - Pong.



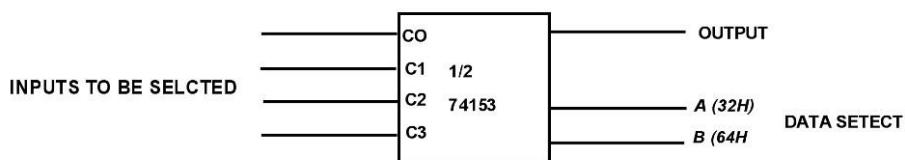
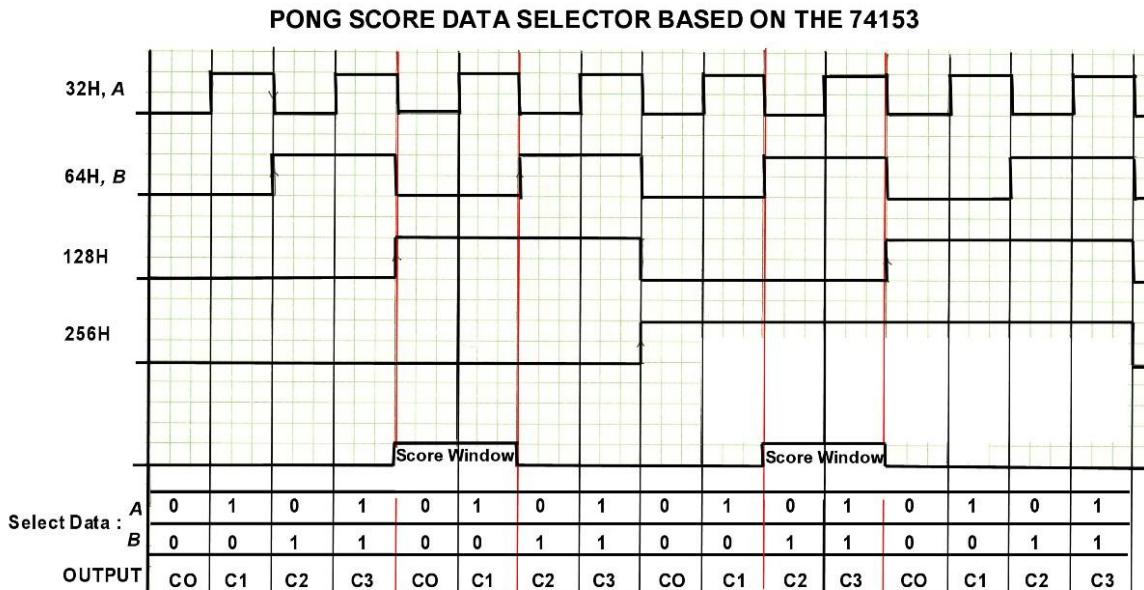
PONG SCORE COUNTERS.



The 75153's are controlled by the 62H and 32H pulses so as to select the data to be displayed from the score counters. A clever trick is used here to be able to display the “1” in the 10’s column:

The (not)E falls low when the score count is 10 or over. When (not) E is selected, for the left hand score window it is transferred from the Co inputs, by the data selector, to the B, C and D output. The A output has a high logic level on Co, so A is high, and B, C and D are low. With A High, the 7448 BCD to seven segment encoder generates the “1”.

When the (not)E is high (eg the score is less than 10) then when it is time to display(select) the left hand side of the left score window again the output data from the data selectors is A &B & C & D are all High, and this causes the 7448 to blank its display, making the 1 vanish. This is a function of the 7448. The same process happens for the other player with (not)E1, for the left hand side of the right score window for the other player. To see how the 62H and 32H selector pulses select the areas within the score window, refer to the following chart:

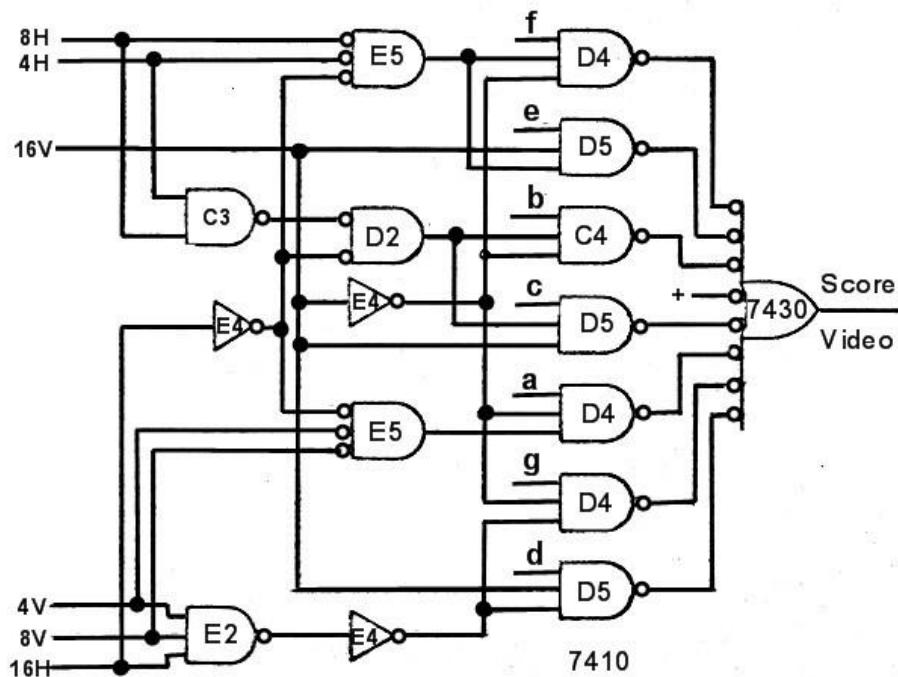


Select data (the 32H and 64H pulses) selects the count value to be displayed in the left side of the score window timing when $A=0, B=0$, this is where the “1” is displayed, then in the right hand side of the left score window with $A=1, B=0$ where the digits 0 to 9 are displayed. The process repeats on the right side of the image, selecting the left side, then right side of the R score window counter data for the other player.

The two player windows are distinguished by the fact that 64H is low for the left hand side and high for the right hand side of the video image.

The gate array:

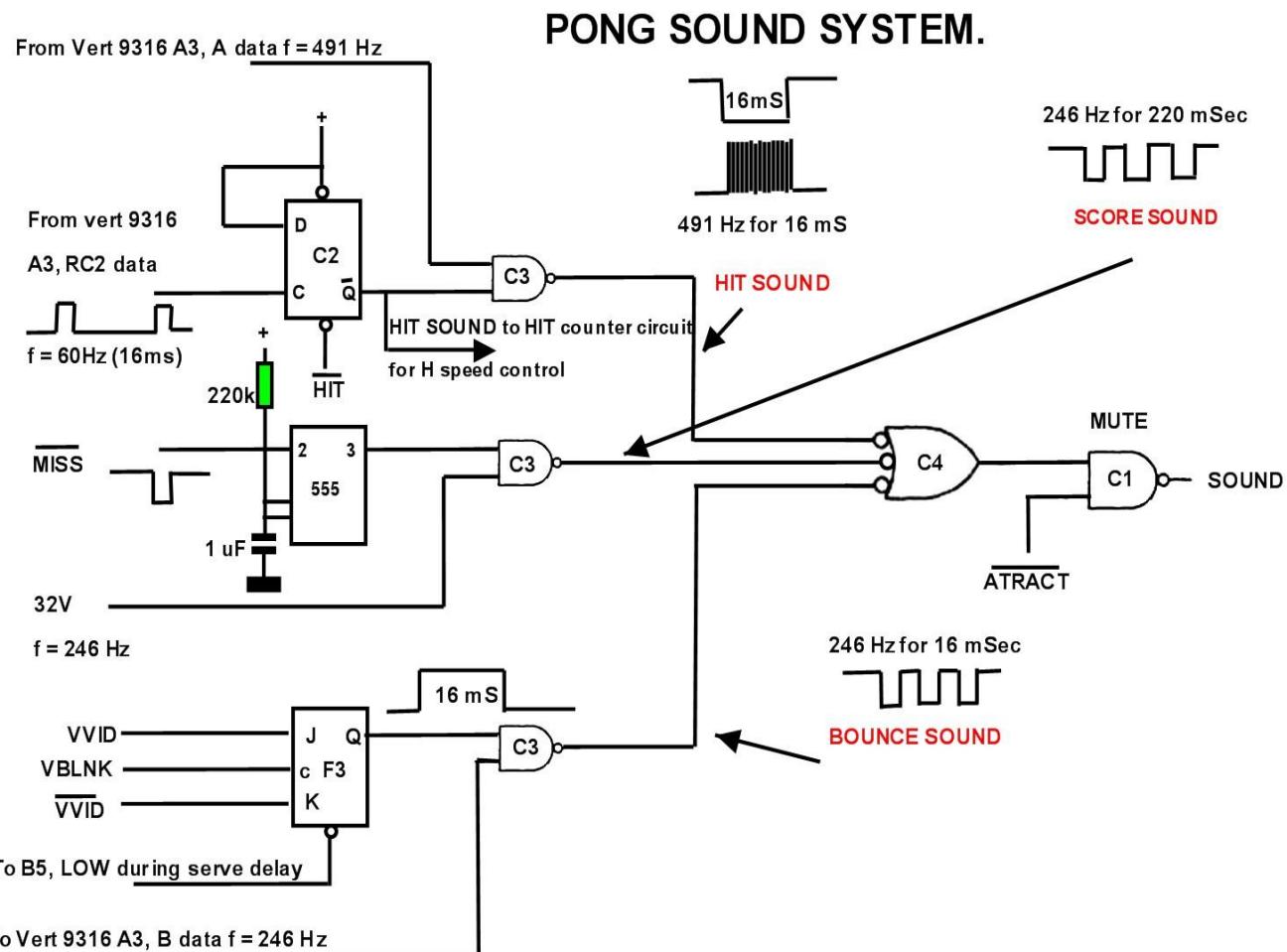
The gate array selects pulses from the H and V sync generators to create the video segments that make up the 7 segment display seen in the score window on the screen:



For example to select pulses to configure the “d” segment or the bottom bar of the seven segment display requires pulses 16H, 16V, 8V and 4V (see chart on score display system above). As can be seen these are the pulses on the lower gate D5 which gates segment d.

PONG SOUND SYSTEM:

Three types of sounds are generated during game play; a Hit sound when the bat and ball are coincident, a Score sound when the ball passes a bat and hits horizontal blanking and a Bounce sound when the ball hits vertical blanking during play.



A score sound timer IC (555) times a burst of 246 Hz triggered by (not) MISS, which goes low when HVID and HBLNK are coincident, this is done by one of the Nand gates of E6 not shown.

For the bounce, if VVID is high (and (not)VVID low) when VBLNK goes low then F3 will have a high clocked to Q. On the next VBLNK pulse VVID will be low and (not)VVID high (the ball has moved away) so Q will be clocked low, the process taking a vertical interval or 16 mSec and a burst of 246 Hz is gated to the output.

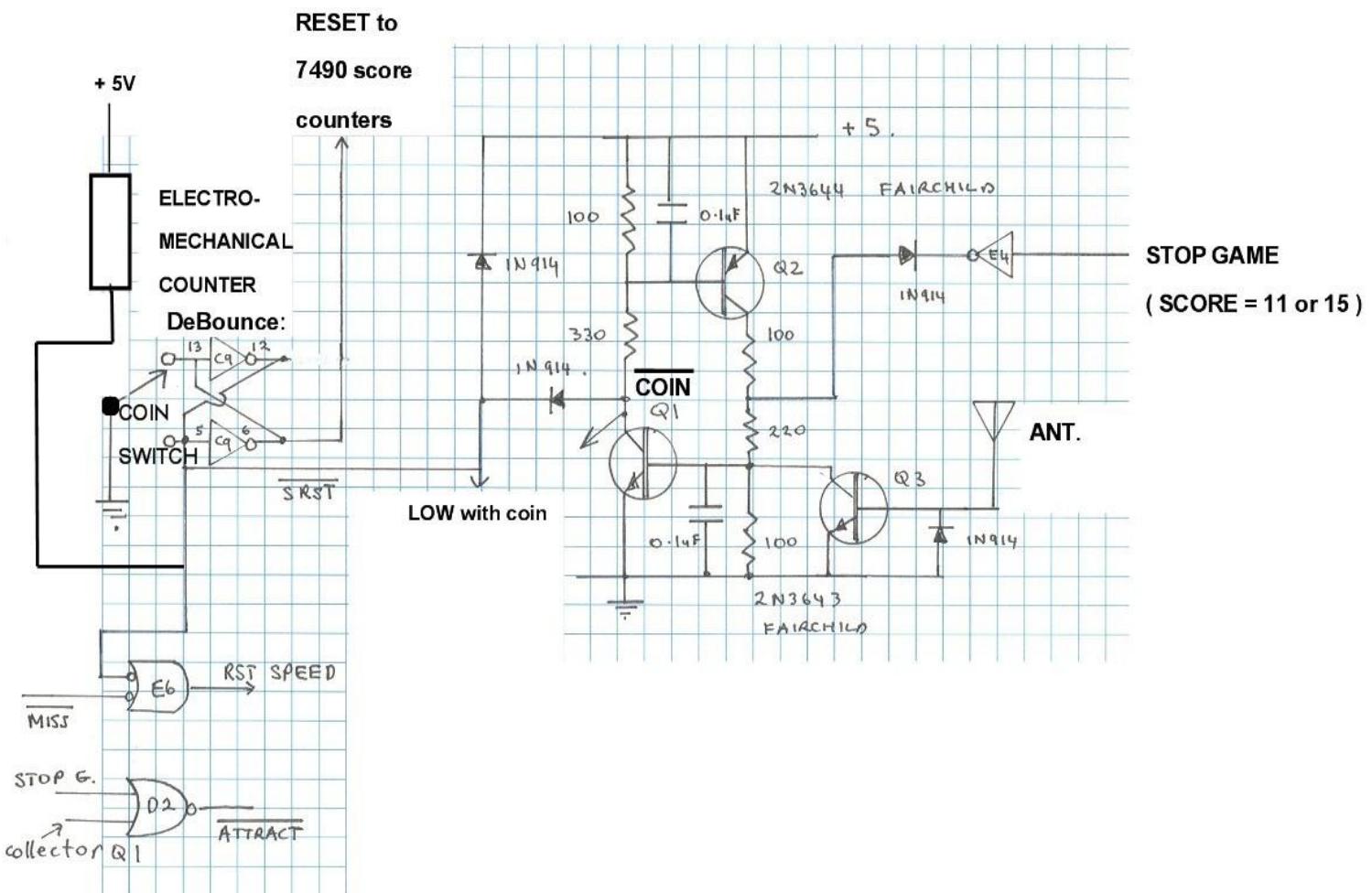
For the Hit sound; (not)HIT goes low when either of the players bats and ball are coincident. When used to reset the D flip flop C2 this makes Q bar (or notQ) high gating the 491 Hz sound through. The vertical part of the ball VVID is generated around the trailing edge of the RC2 terminal or pin 15 of the 9316 location A3, in the vertical ball position circuit. Therefore it takes 16 mSec for the next leading edge of the RC2 pulse to clock C2, setting Q high and Qbar low, stopping the hit sound.

GAME START STOP LATCH & STATIC DISCHARGE & RFI IMMUNITY:

It was realised by the early electric game designers, that cluey teenagers might try to get a free game credit by feeding bursts of static electric discharge, or perhaps RF into or near the machine, potentially triggering a credit or perhaps get the game started in some mode, without actually putting a coin in the mechanism. This problem was addressed with a single transistor, with a diode in its base and its base connected to a free wire in the machine called an “antenna”. A strong RF field or static discharge would cause the transistor to conduct, resetting the game to the game over or Atract mode. (later Atari changed the spelling to Attract in documentation on TANK)

The circuit below is of the game latch of Atari Pong. Again most other manufacturers copied this arrangement:

GAME STOP- START LATCH PONG



At the time the game is powered up, Q1 and Q2 are not conducting so the collector of Q1 is high, and Q2 low. The coin switch is briefly closed by the coin passing by this causes Q1 and Q2 to latch into a conducting state, until the maximum score is produced, either 11 or 15 depending on where the PCB switch was set. At this point STOP GAME (STOP G)

goes high, resetting the latch. Any powerful RF in the antenna, or a static discharge, is rectified by the 1N914 diode & detected by Q3 which will also reset the latch.

When the coin passes through and the coin switch closes briefly, the horizontal speed is also reset so the game always starts at the lowest horizontal ball speed. A (not)MISS also resets the speed. Also when the collector of Q1 is high OR STOP GAME is high, the game is in the ATRACT mode and the current, or last set of scores, remain on the screen. The 7490 player score counters are reset with a HIGH when the coin passes the switch and (not)SRST falls transiently low.

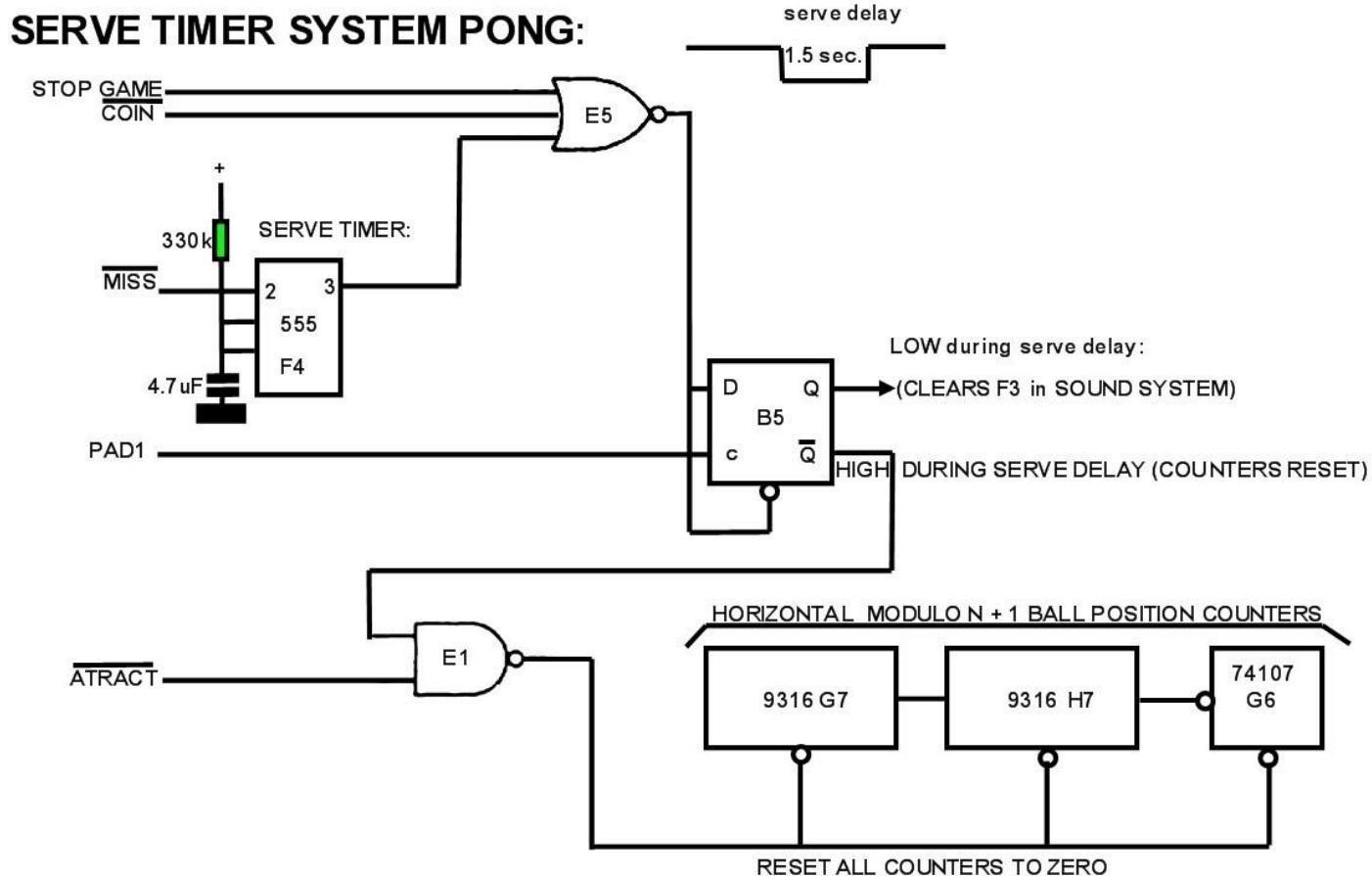
With the (not)ATTRACT low: The bats go off via H3, the sound is muted via C1. The Score sound (SC) is able to clock H3, via D1, to allow the ball (HVID) to bounce off H blanking so the ball bounces silently around the screen attracting the next would be players.

Sometimes the back emf from the electro-mechanical coin counter would damage gates E6 and C9, despite the back emf suppression 1N4001 diode. A factory modification was made to the rear of Atari's PCB by cutting the track and adding a resistor and capacitor to help prevent this problem.

PONG SERVE TIMER SYSTEM:

This system is designed to create a brief delay after the coin is inserted before the ball is "served" and to control the position in which the ball appears on the screen when it is served. The ball is served from the net area, just to the right of the net.

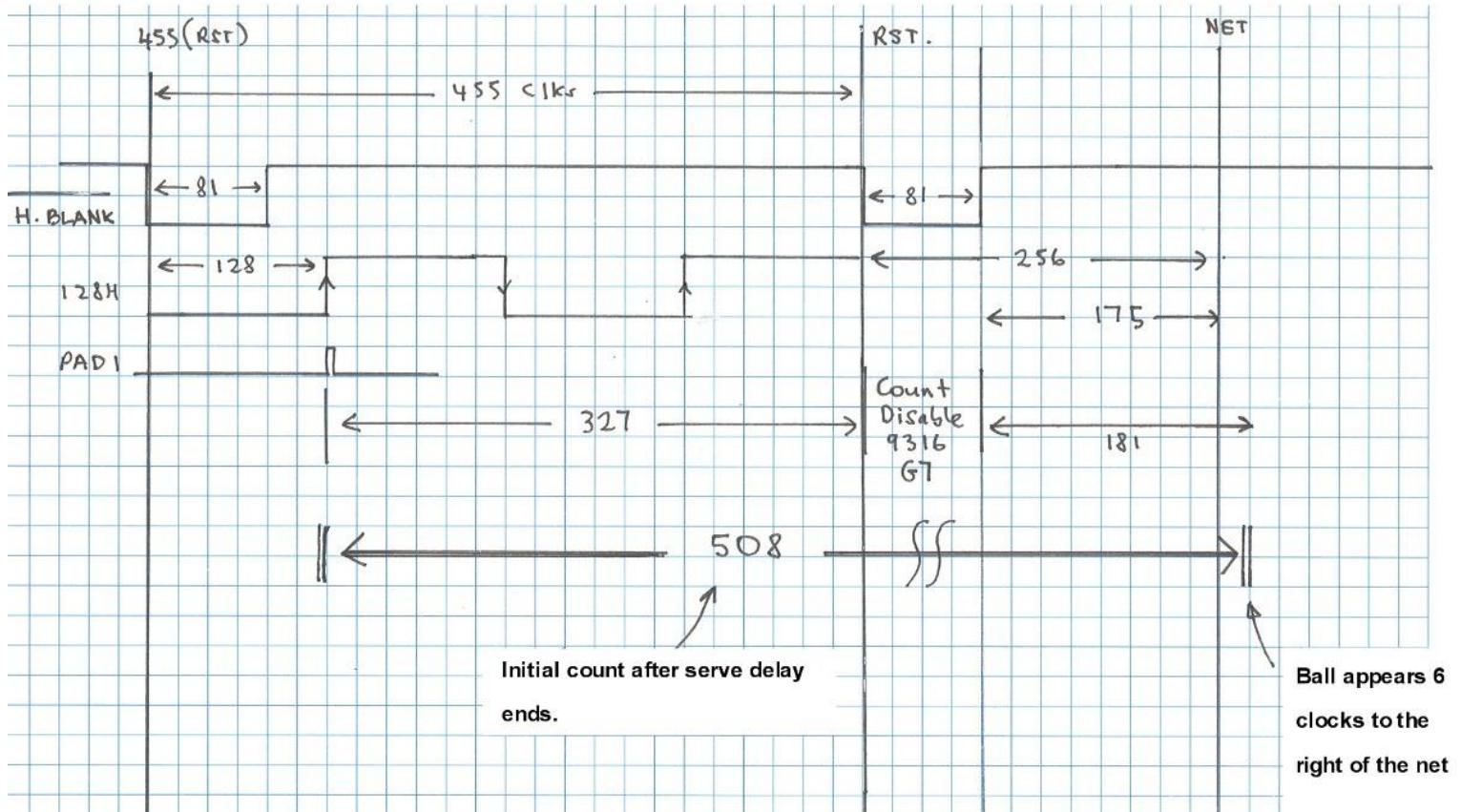
The diagram below shows the basic circuit elements:



PAD1 is the player 1 (left screen) bat signal after it has been gated horizontally to create the bat video image, it is a 60Hz pulse. This pulse clocks a high level to B5's Q output, ant a low to the Qbar. During the serve delay (afer a player misses a hit) the logic state of Q and Qbar reverse, for about 1.5 seconds. This zero's the horizontal ball position counters.

The counting processes are best summarised on a chart. The position of the ball appearance at the serve has been synchronised to the position of the left hand players bat (paddle) by using pulse PAD1:

PONG BALL SERVE SYSTEM



The PAD1 pulse synchronises the time that the 9316 counters are allowed to start counting after the serve delay time ends.

Then to produce any horizontal ball signal the H 9316 counters have to count $256 + RC(\text{ripple count} = 240) + 8 + 4 = 508$ clock pulse counts after PAD1. These pulses are fed to the input of H6 a 4 input Nand gate

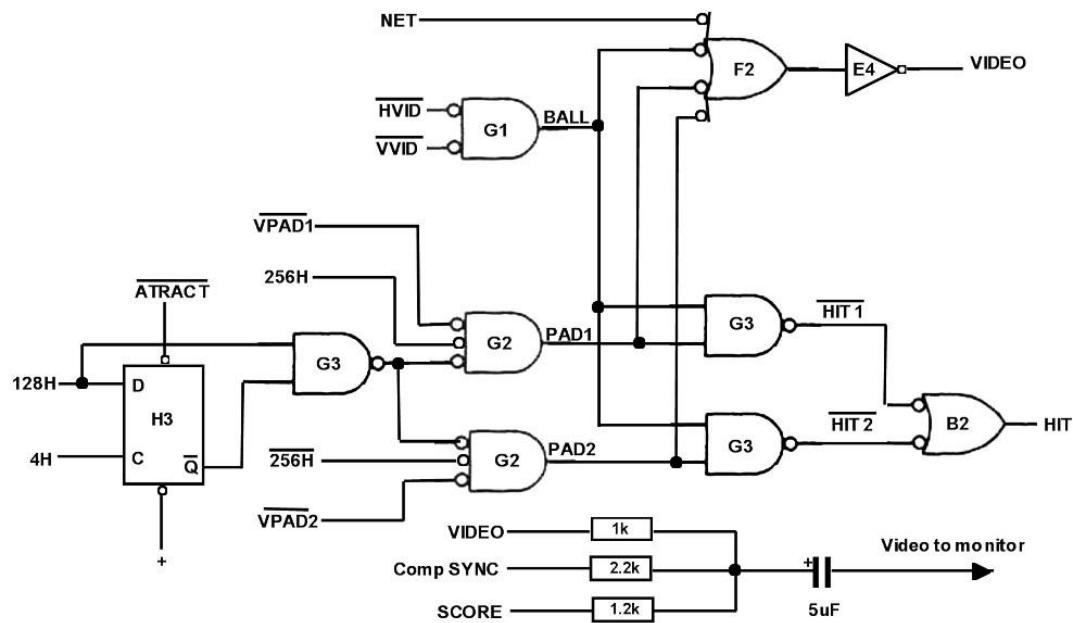
which produces the (not)HVID pulse. On the 4th of another 4 counts later (which determines the horizontal width of the ball) the horizontal 9316 counters load. Then after that it counts every 374 counts in the usual way (assuming a load data value A= 0, B = 2 at G7).

Therefore, as seen from the chart above the ball is served from a horizontal position very close to the net. Its vertical velocity value is whatever value it had at the time of the miss when it strikes horizontal blanking. In addition, the direction of the horizontal motion is the direction it was at the time of the miss, so that the ball is served to the player who missed it.

ASSEMBLING THE PULSES OF THE VIDEO & the HIT DETECTOR:

Most of Pong's circuitry has been described above. There is a gate array used to combine the (not)VPAD 1 & 2 pulses for the bat, the net pulses and the ball to produce a combined set of pulses called video. Video is combined with the composite sync and the score by mixing with 3 resistors. The gates which assemble the combined video signal:

PONG: VIDEO MIXING & HIT DETECTOR



The flip flop H3 is clocked by 4H and with 128H as the data positions the bats(paddles) in the correct location on the video image and ensures they are 4H wide.

The bat vertical rate pulses (not)VPAD1 and (not)VPAD2 from the players bat generators are gated to the left and right hand side of the screen by the 256H and (not)256H pulses. The PAD1 and PAD2 pulses represent the bat as it is seen in the video signal and on the screen.

When the BALL pulse is coincident with either PAD1 or PAD2 then a HIT is detected. The HIT, as noted above in this article, is used to transfer and latch the bat data from the position on the bat, where the ball hit the bat, to the vertical velocity encoder to determine the vertical velocity when the ball reflects from the bat.

MODIFICATIONS TO PONG:

NOTE: THESE MODIFICATIONS ARE SUGGESTED FOR THOSE BUILDING NEW TTL PONG GAMES. GIVEN THAT ATARI ARCADE PONG BOARDS ARE BECOMING RARE, DO NOT DAMAGE THEM WITH MODIFICATIONS UNLESS ABSOLUTELY NECESSARY. THEY SHOULD BE PRESERVED FOR FUTURE HISTORIANS.

This section addresses various cures for some of the original game's problems. It should be pointed out that in the scheme of things these 5 problems were not too severe and do not reflect negatively on the genius of the game's overall design. Also note in any of these circuits if a

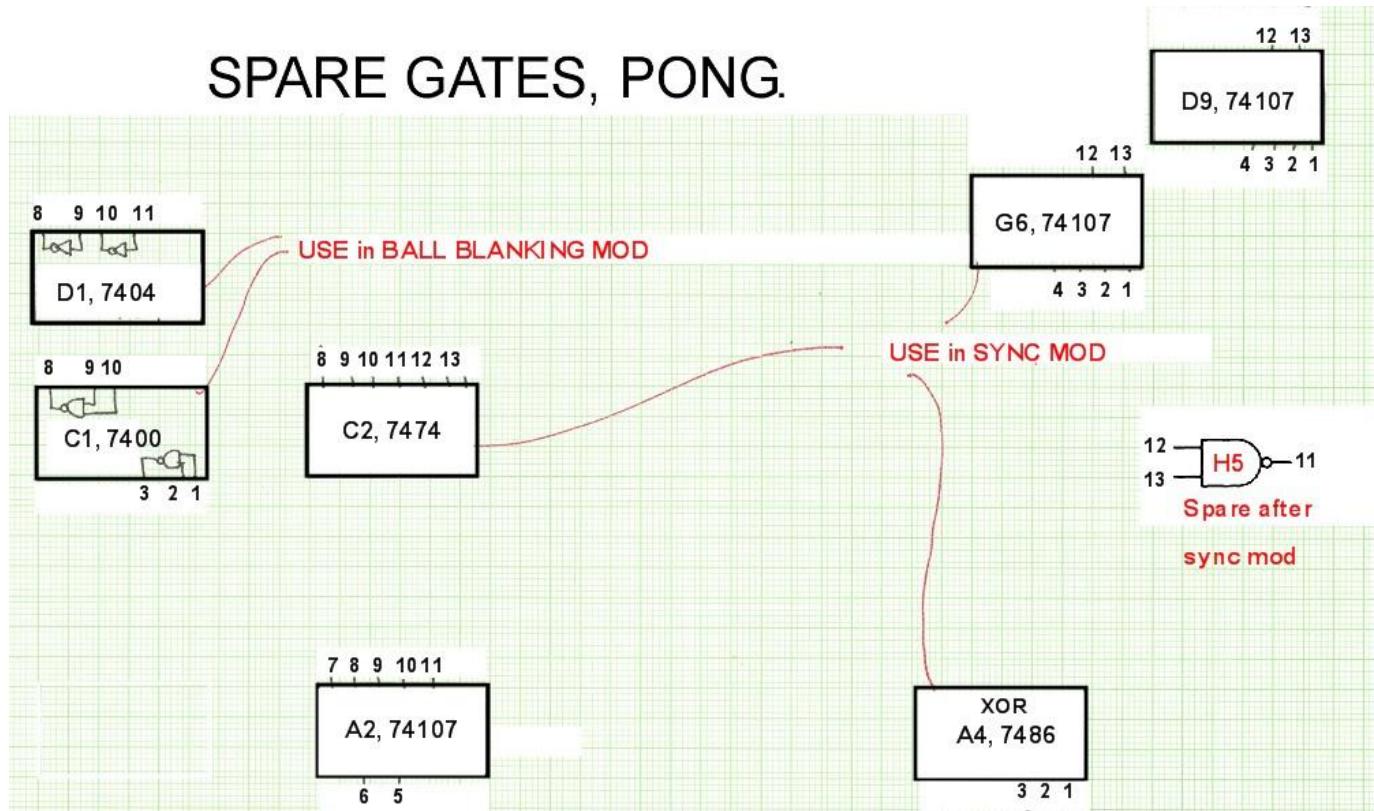
connection is shown to + then this means it is connected to 5V via the common pull up resistor on the pcb.

The problems are:

- 1) Picture centres to the left, making monitor H hold setting difficult.
- 2) Weak net signal in later pcb's
- 3) Ball video signal appears in blanking upsetting the monitor's sync a little at times.
- 4) Bat (paddle) range limited, frustrating some players.
- 5) Ball latches in vertical blanking.

Before making any modifications it is useful to note there are a number of unused gates and flip flops on the original pcb:

SPARE GATES, PONG.



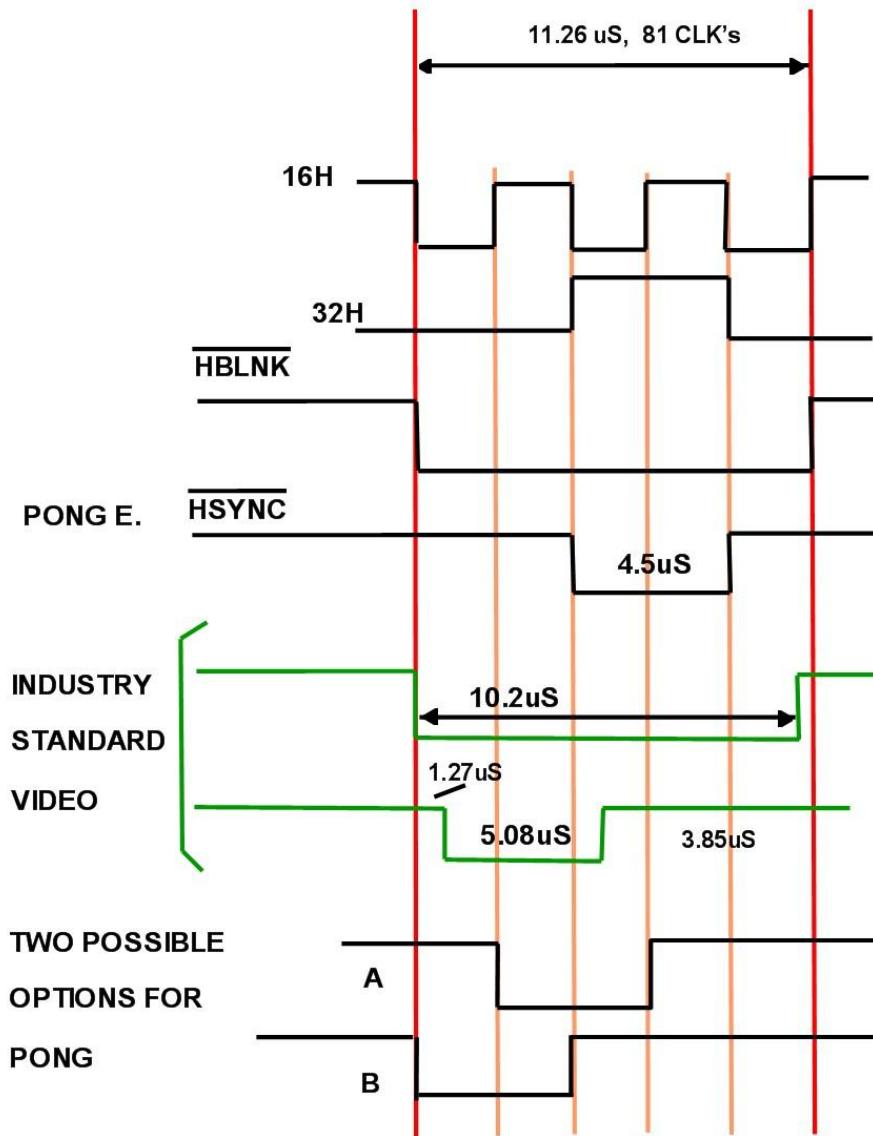
So the game could have been done with one less 74107 IC package, but the pcb would have been more difficult as they are a way apart.

1) THE PICTURE CENTRES TO THE LEFT ON THE MONITOR:

The horizontal sync pulse in the incorrect position within horizontal blanking time.

(This problem was fixed in PONG doubles, a 4 player version)

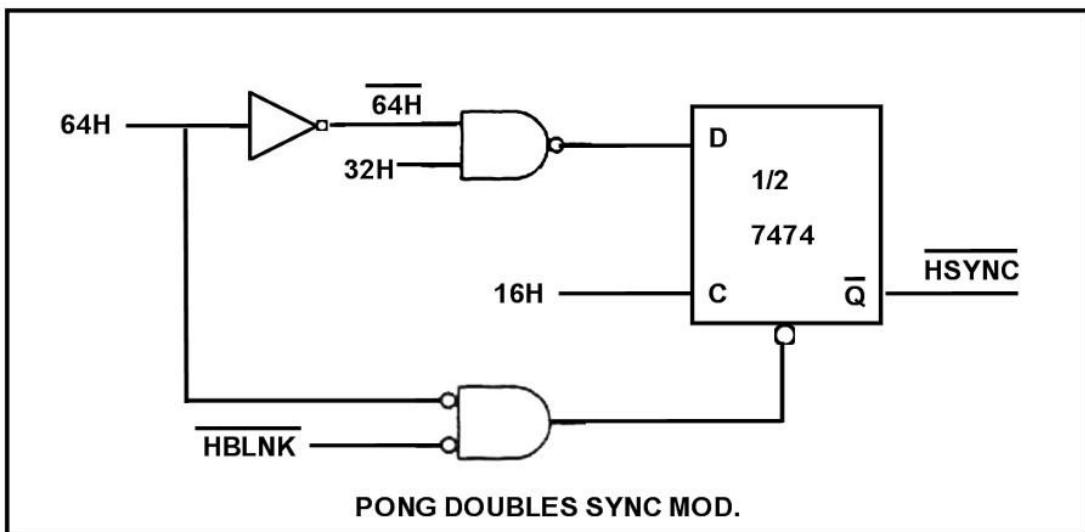
The original PONG E horizontal sync pulse timing was as follows:



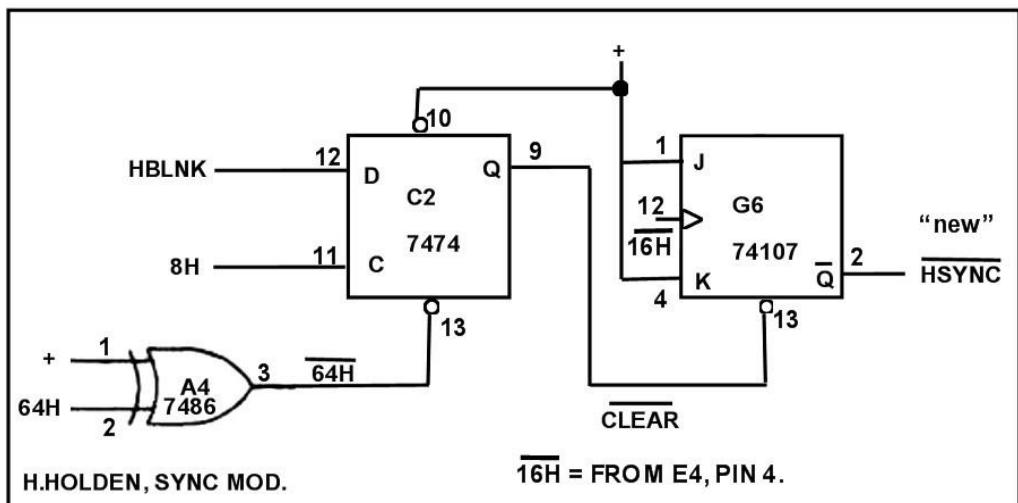
The original pong sync was displaced to the right in the blanking interval compared to standard video sync signals. This meant that the central part of the video image, ie the net, was closer in time to the sync pulse, therefore the net and screen image was left shifted. This could be compensated for by adjusting the H. Hold control on the TV(monitor).

Due to the horizontal flywheel circuit in the monitor's H hold circuit, turning the H hold control results in right to left shifting of the image. However, when this is used to "centre the picture" it is more likely the set will go out of horizontal hold at times.

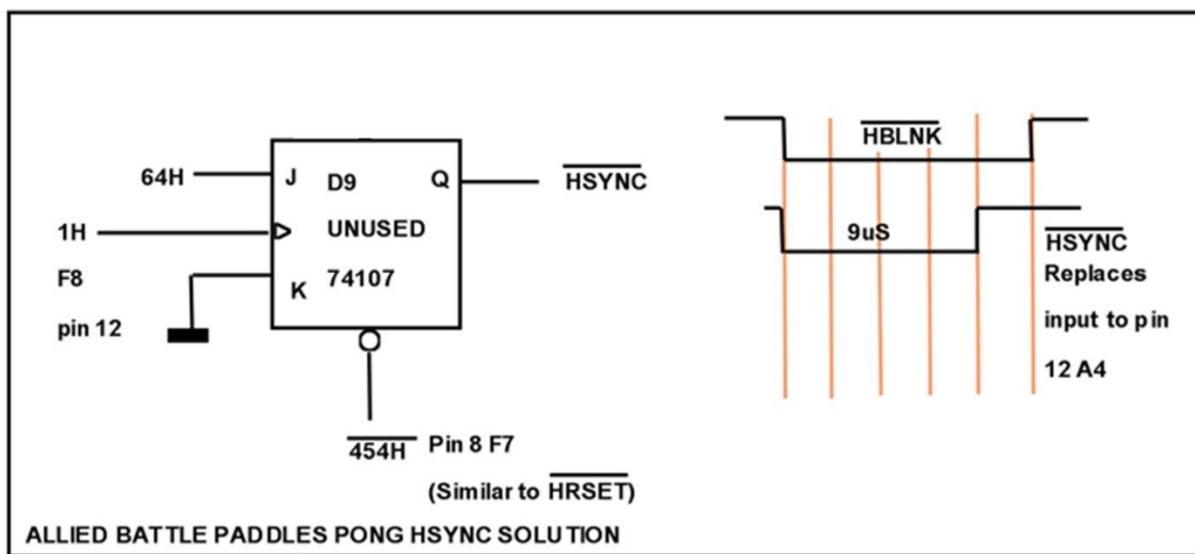
In Pong doubles, Atari had modified the H sync circuit to this version which gives the timing Option A in the above diagram:



There were a number of unused gates in original Pong and it is possible to use some of the original unused gates get the same result as option A. The spare XOR gate is used as an inverter:

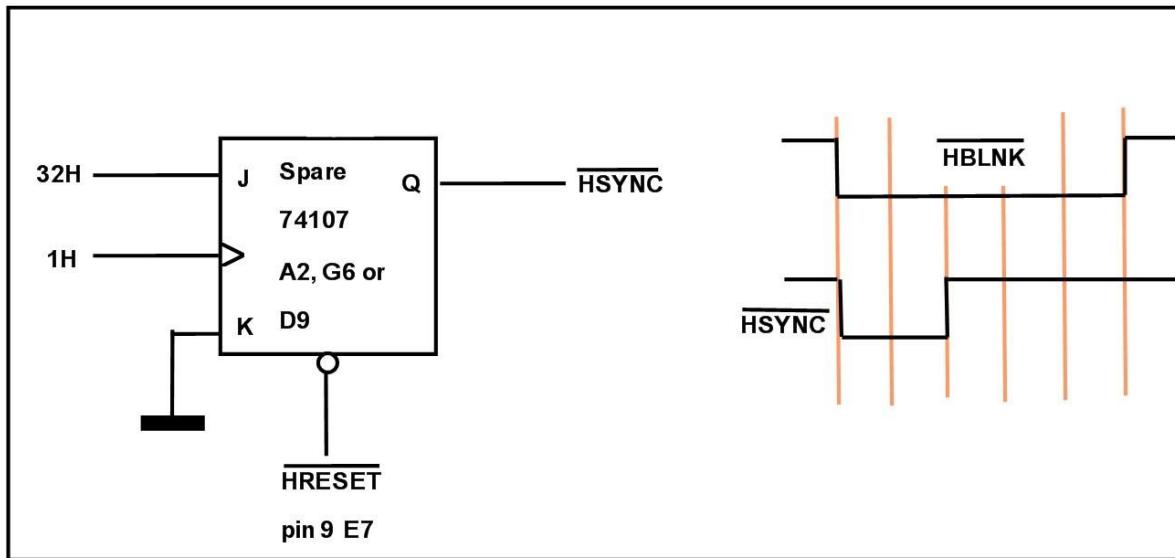


Other variations include one by ALLIED BATTLE PADDLE which was found wired as a factory looking modification giving a broader than standard HSYNC pulse starting right at the beginning of vertical blanking:



The Allied modification, although a non standard H sync pulse width, does give improved picture centering.

A variant of this is probably the most simple option of all:



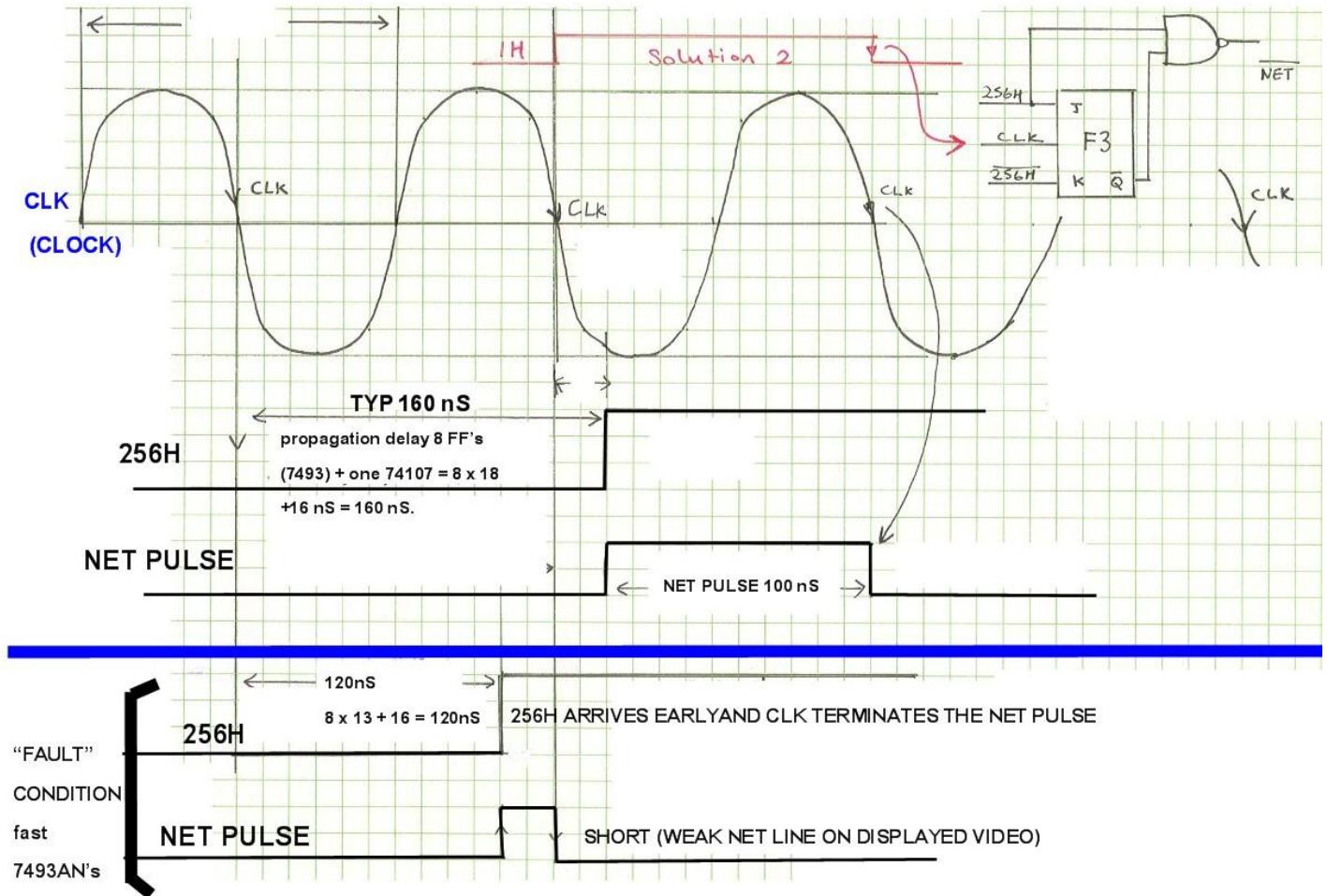
However the simple version above can result in over correction, moving the image a little too far to the right.

2) WEAK NET VIDEO SIGNAL.

This problem was detected in a later model pong E PCB which has SN7493AN IC's in the H sync pulse generator, each with a minimal transmission delay, just on the inside of the specs for this type of IC's.

However due to cumulative delays in a ripple counter system the 256 H pulse was arriving earlier than typically usual and upsetting the NET generator. This is explained in the following diagram:

WEAK NET PULSE ON SOME LATE PONG PCB's WITH 7493AN IC's IN H SYNC CIRCUIT



In an early Pong E PCB the total delay measured in the 7493 horizontal sync counter chain was 160nS. This was accounted for by 8 flip flops each with a 18nS delay and one 74107 delay making a total of about 160nS delay. In a late Atari PONG E pcb the IC's were SN7493AN. In this case each internal IC flip flop had a 13nS delay, each type of 7493 still within the manufacturers specs. However with these “faster IC's” the NET circuit malfunctioned producing a weak net pulse.

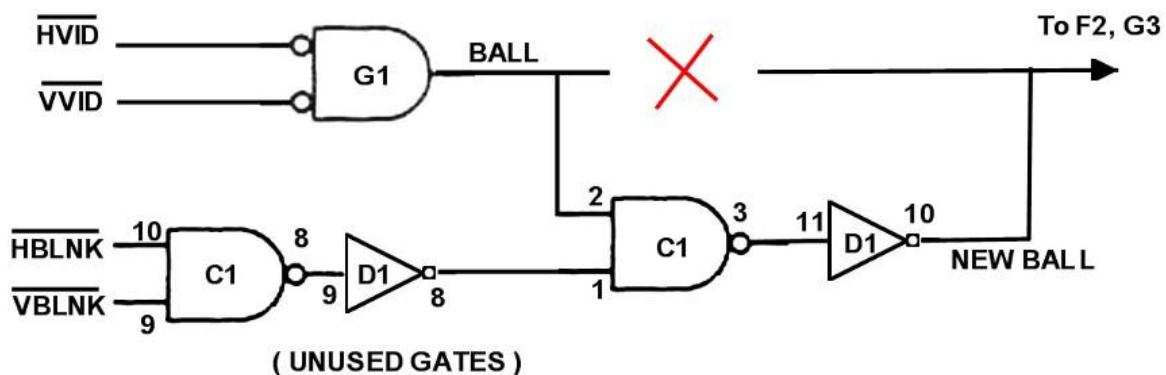
There are 2 solutions that fix this: One is to add a 220pF capacitor on two of the H 7493 counter outputs to ground. This creates enough added delay to get the situation to normal. The other (Solution 2) is to use 1H

as the clock signal for the net generating flip flop F3. The 1H is half the frequency of the CLK, so if the resultant overall length of the NET pulse ends up being about 160nS, which gives a normal looking net on the screen.

3) BALL VIDEO SIGNAL APPEARS IN BLANKING:

This makes the monitor image jump a little (depending on the brand of monitor) when the ball bounces off vertical blanking. The ball video signal however shouldn't really be in the blanking area which is reserved for sync pulses. The ball video signal can be gated out of blanking using spare gates:

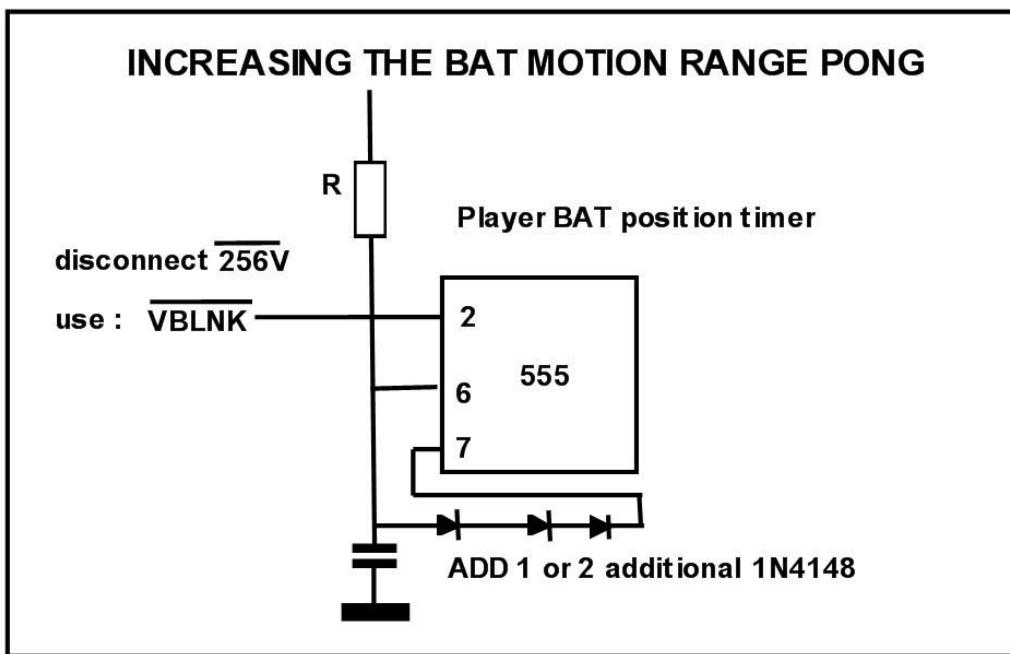
GATING BALL OUT OF BLANKING:



4) LIMITED BAT (PADDLE) RANGE:

On the original game the range of motion of the bat (or paddle) is limited, even when set ideally on the adjustment pot and the ball can pass just above and below the bat. Some players complain about this as they are unable to avoid losing a point even when they believed they have the skill to do so.

A modification which improves the bat range is as follows:



It should be noted that increasing (adding one diode) and using (not)VBLNK, the bat range does increase the chance of the bat being able to “push” the Vertical component of ball (VVID) into vertical blanking where it can remain “trapped” disabling the game and requiring an on-off power reset. This might have been why the bat range was limited in the first instance. Only the original designer Mr Alcorn would know the answer to that question. Further bat range can be acquired by

adding the 2nd diode. This makes it inevitable that the ball will be pushed into blanking at some stage unless the ball capture/lockup problem is corrected with the modification outlined below.

(If a 555 is being replaced on a Pong pcb, it is important it is an original NE555N, not a 555V or any other 555 versions such as Cmos as they do not work correctly).

5) BALL & VERTICAL BLANKING LOCKUP PROBLEM:

This moderately complex problem turned out to have a simple solution, though it was a tricky one to solve. The problem comes about due to a number of interactions creating an inescapable loop.

The lockup problem tends to occur if the bat range is enough to “push” the ball into the vertical blanking interval, either through maladjustment of the bat preset pots, or by modifications to gain extended bat range. On occasions it is possible for the game to start in this condition, although that is uncommon but it has been observed.

Firstly the vertical ball signal VVID goes high during blanking, because at the moment the ball position 9316 counters are inhibited (by vertical blanking) the count value is such that VVID is high. The VVID pulse is “widened” during this time to a longer duration than vertical blanking which is one factor.

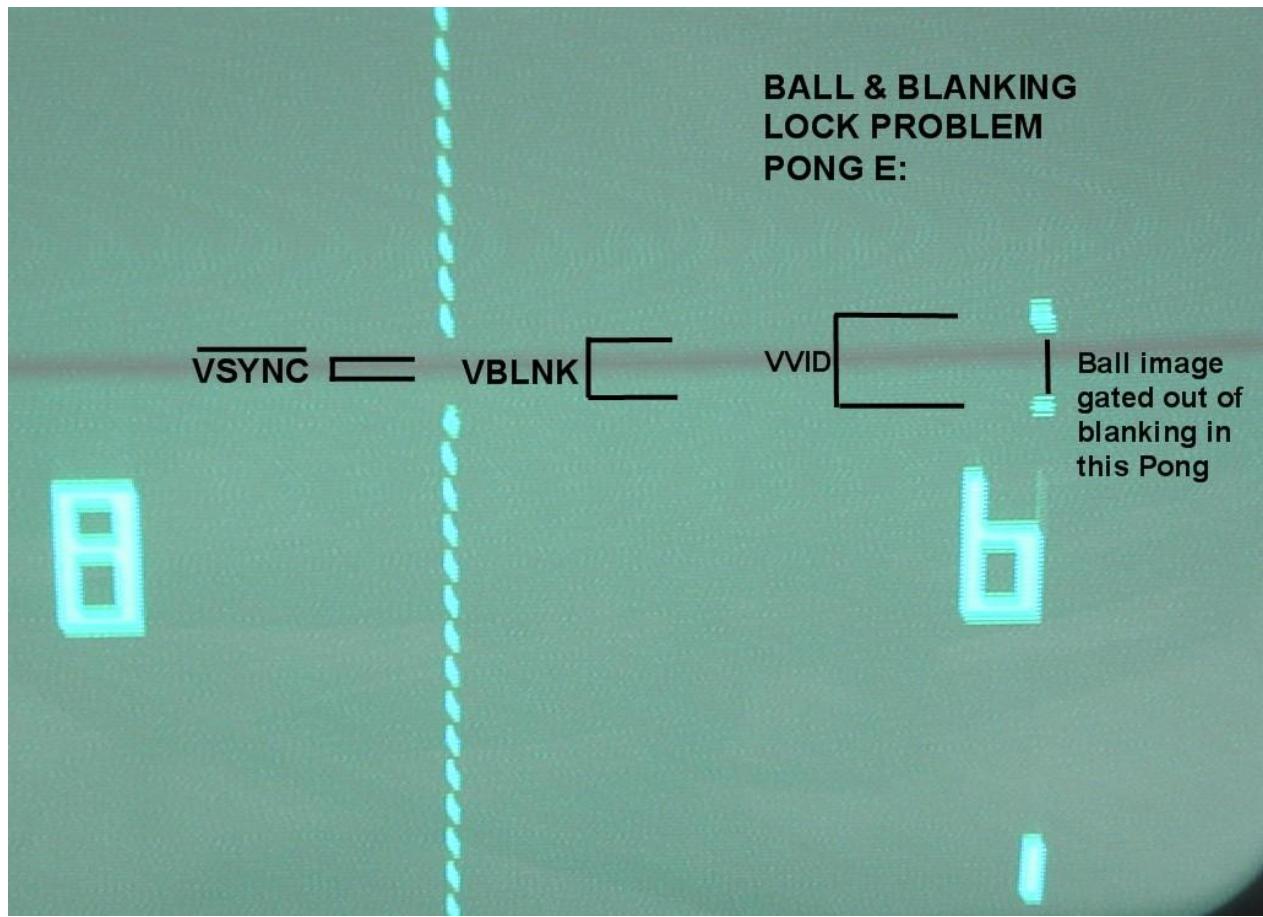
When the bat hits the “ball” flip flop A2 is cleared (reset) so that the balls motion corresponds to the bat/ball intersection data. Ignoring or disabling any horizontal motion, if the ball was between the top of the screen and the top of the bat, and intersecting with the top of the bat at least once, it will have velocity encoder value 13 and bounce off the top of the bat toward the top of the screen and towards vertical blanking

VBLNK. When VVID becomes coincident with VBLNK's falling edge, then 74107 flip flopA2 is clocked (Toggles) to reverse the ball's vertical motion back toward the top of the bat. So the ball sits there moving between VBLNK and the top of the bat. The closer the bat is to VBLNK, then the faster the ball motion and the higher the frequency of vertical direction reversal. If this happens with each VBLNK pulse or at the vertical rate, then the ball reversals have the same frequency as the vertical interval.

If the ball oscillation rate between the bat and vertical blanking becomes equal to the vertical interval itself, then VBLNK can take over from the bat pulse in controlling the ball reversals if the data (VVID) on the J&K inputs of the 74106 remain high when VBLNK falls low. They can remain high over consecutive field as will be explained due to the range that the VVID is allowed to move in one vertical interval. Then when the bat is moved away the VBLNK pulse falling edge acts to redirect the "ball" back into blanking and a lockup state is sustained.

For investigation the problem can be generated by stopping the ball's horizontal motion by loading IC H4 pin 3 to ground with a 56 ohm resistor and making the ball stop above, or below a bat. The bat (from its control and preset adjustment) can be used to push the ball toward and into the vertical blanking area. Initially the ball will oscillate more and more rapidly as the space between the bat edge and blanking decreases.

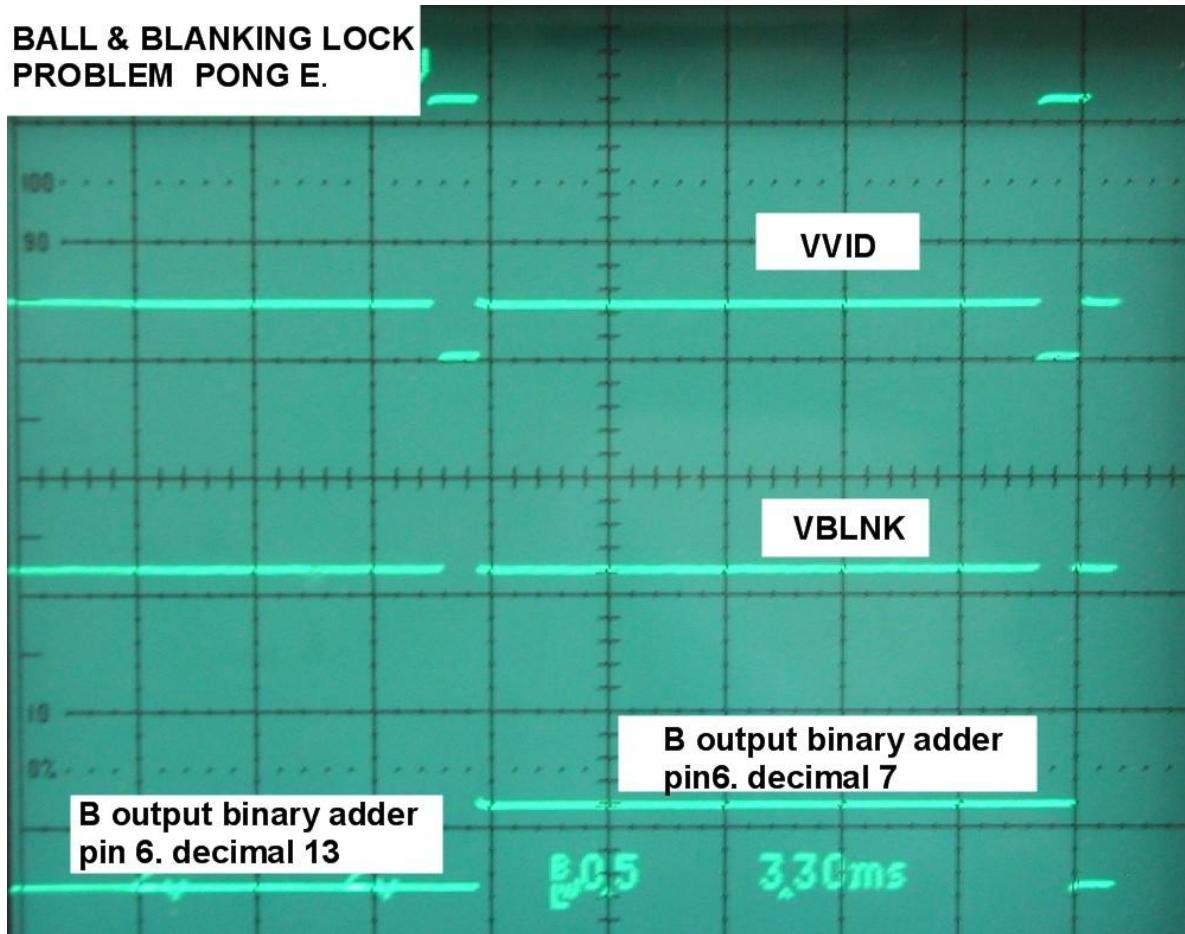
Ultimately when the bat is forcing the ball to oscillate across vertical blanking it will lockup there in sustained oscillation, even when the bat is moved away. The flowing oscilloscopes and photos show what is happening:



In the photo above, the ball is locked around vertical blanking (after the bat has been moved away) and looks different than the original PONG pcb video image because in the version photographed above the ball signal has been gated out of the video signal during blanking, however this does not relate to the problem under discussion. On the original circuit (ATARI PONG E) the ball is seen to extend right across vertical blanking and travelling horizontally. (The horizontal motion was disabled for the photo).

Looking at the important signals during this lockup condition:

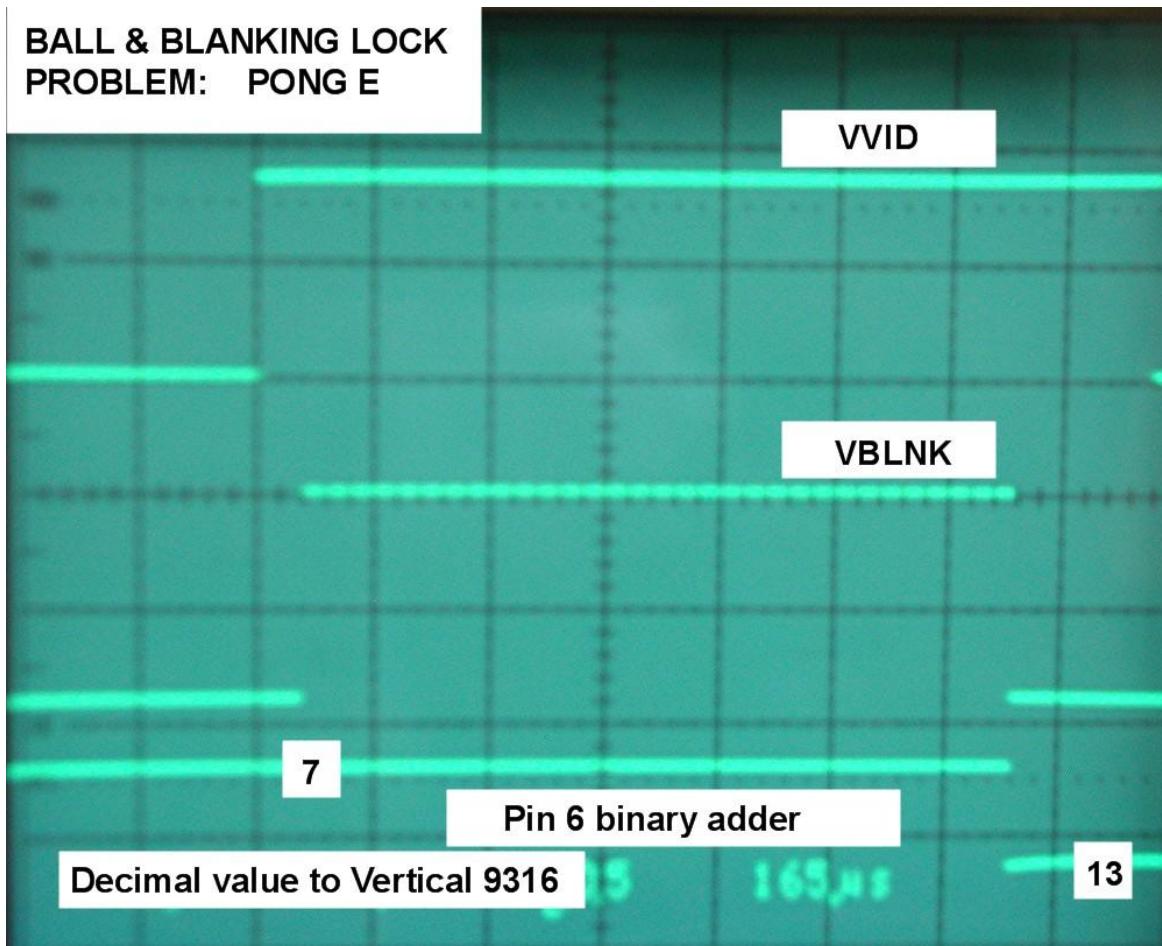
**BALL & BLANKING LOCK
PROBLEM PONG E.**



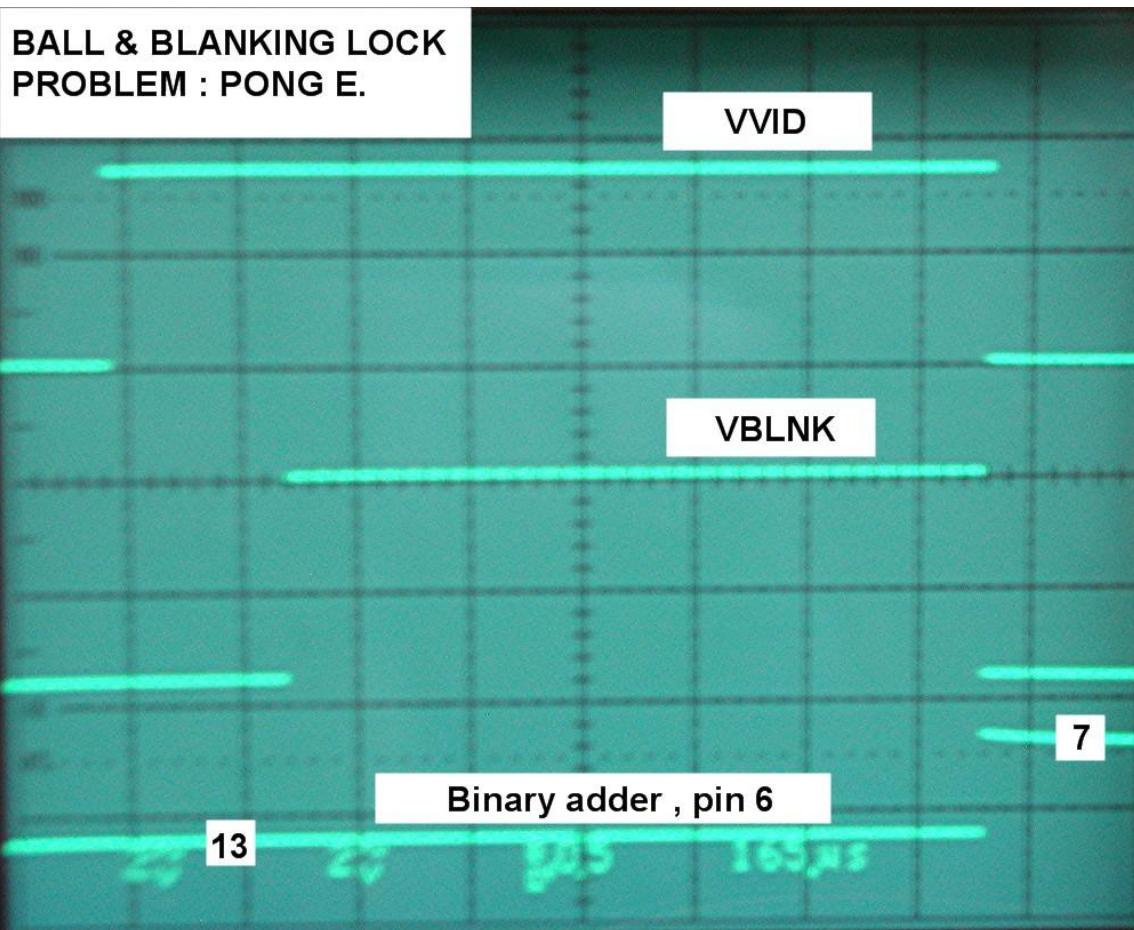
The VVID and VBLNK pulses are locked together in this anomalous condition. The vertical direction flip flop A2 is functioning normally reversing the data from the two flip flops A5 and B5. The output of the binary adder is behaving correctly with the value on pin 6 (the B output) reversing and corresponding to ball motion of load values 13 and complimentary value 7.

To get a better look at the timing:

**BALL & BLANKING LOCK
PROBLEM: PONG E**



When the vertical load value is 7 the elongated VVID pulse *has not moved enough(to the left on the oscilloscope) after 1 vertical interval* to have escaped VBLNK at the moment VBLNK falls low and clocks A2 pin 12, so the flip flop A2 toggles as VVID is high when VBLNK falls low. Looking at the next vertical blanking pulse in time after *one* vertical interval:



Again, during the vertical interval when the vertical counter load value had been 13, the ball has not moved far enough to the left (upwards on the game's video screen) and VVID is still high when VBLNK falls low so the flip flop A2 again toggles reversing the direction of the ball. At the next horizontal interval the problem recurs and the VVID pulse simply oscillates in position about the vertical blanking area. Difficult to see in the above oscillogram; VBLNK does fall low prior to VVID falling low so A2 is toggled.

When VVID is outside vertical blanking area its pulse width is only about 250uS (4 horizontal lines or intervals).The elongated VVID pulse

comes about because of the inhibition of the vertical counters during blanking over 16 counts or 16 horizontal intervals or 1016 uS.

Cause of the lockup:

Over one vertical blanking interval, the ball, or VVID component of it, can only move an amount corresponding to a fixed number of counts of the (not)HYN and at the load value of 7 or 13. These define the maximum up and down ball speeds (or positions) that the ball pulse can move; 3 counts left or right, which is $3 \times 63.5\text{uS}$ or 189.6 uS . This is the amount that VVID moves left or right around the VBLNK over one vertical interval in the two oscilloscopes above in the lockup condition. This is not enough for VVID to “escape” the blanking area, in that in both cases VVID is still high at the moment VBLNK falls low, which causes the vertical velocity direction flip flop A2 to relentlessly toggle at a *vertical rate* with each vertical interval controlling the ball’s (or VVID’s) position around the vertical interval, and “trapping” it there.

The falling edge of VBLNK effectively takes over from the (not)HIT pulse, so when the bat moves away the motion around VBLNK is sustained.

Normally during a game the flip flop A2 only toggles at a much lower rate when the ball and blanking are coincident during a bounce, or flip flop A2 is cleared by a (not)HIT on its pin 13 when the vertical velocity is encoded by the bat data.

With the bats are spaced away from the vertical blanking interval (as they were in the original design) the ball oscillation frequency between the bats and blanking, could not approach the vertical frequency rate, as there was a good gap between the bat and the vertical blanking interval. This of course allowed the ball to pass between the bat and the upper or lower edges of the screen image, upsetting some players.

In summary:

The cause of the problem is the ball's (or the vertical component of it VVID) bouncing rate (between the bat and VBLNK) increasing to match the vertical rate and synchronising with it. This is due to the fact that the vertical ball position circuit's range is set to the maximum of 3 horizontal counts above or below the vertical sync rate limiting how far the VVID pulse can move over one vertical period with respect to VBLNK. Also the VVID pulse width is elongated by vertical blanking as the vertical ball position counters are inhibited during vertical blanking and the VVID itself is 4 horizontal counts wide, not 3. The VVID and VBLNK pulses and the circuit configuration present at the vertical direction reversal flip flop A2 are such that this is not enough for the ball to escape vertical blanking over *one* vertical period allowing VBLNK to take over from the bat in the control of A2 and reverse the ball back toward blanking on the next vertical pulse, VBLNK. The VVID (the ball) remains trapped in the absence of any interaction with the bat under this condition.

Solutions:

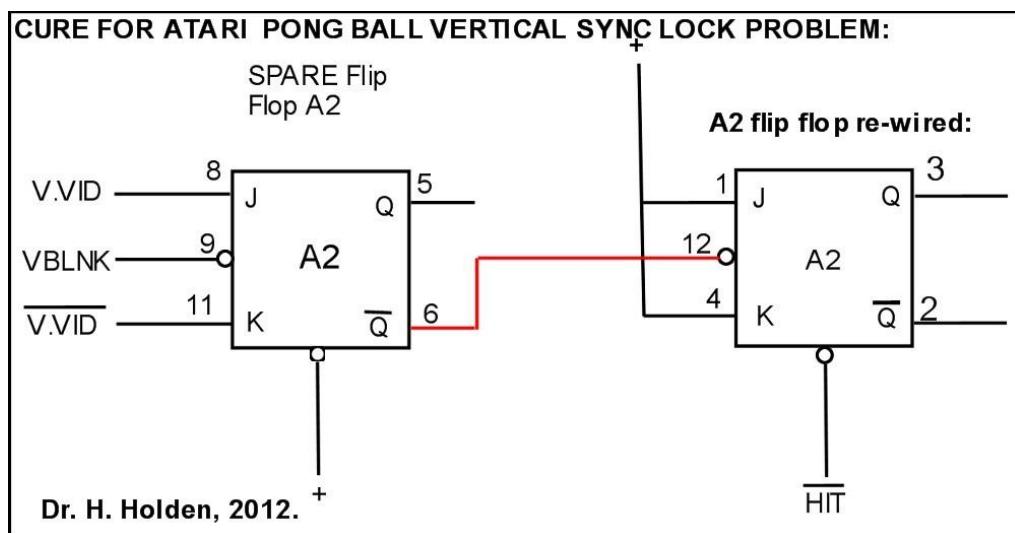
One way to fix this problem would be to have the ball bounce off a border created by a set of pulses crafted before and after vertical blanking. This was done on some Pong clones where a thin white horizontal border was placed on the top and bottom of the raster and the ball bounced off that. This prevents the ball entering the vertical blanking area and the vertical component of the ball pulse becoming elongated at that moment too. It also would require that the bounce sound circuit that uses VVID and (not)VVID and VBLNK were modified because with no coincidence of VVID and VBLNK that circuit would stop working.

Another option to cure this anomaly I have developed which is simple to execute is to make use of the other unused half of the flip flop A2. The idea is to make the vertical velocity reversal flip flop A2 unresponsive to a new reversal event for a period after each bounce off vertical blanking, so that another ball reversal can only occur after an added vertical period (0.0166 sec) for example. Then VVID could move 6 counts, not 3 after a bounce and be definitely sure to escape vertical blanking.

As it turns out, one of the flip flops in the A2 package was not used in the original design, so the vertical position reversal flip flop A2 and connections can be configured to include it.

Referring to the schematic below, the negative edge of VBLNK applied to the spare flip flop pin 9 clocks a low to its Qbar terminal pin 6 when the ball (VVID) is coincident with VBLNK. As long as they stay coincident Qbar will remain low with each VBLNK. When the ball moves away from blanking, the *next* VBLNK clocks a high to QBar pin 6 and only after that can the system be re-triggered because the reversal flip flop A2 on the right only clocks after the negative edge of the pulse on its pin 12.

The circuit below shows the modified configuration:



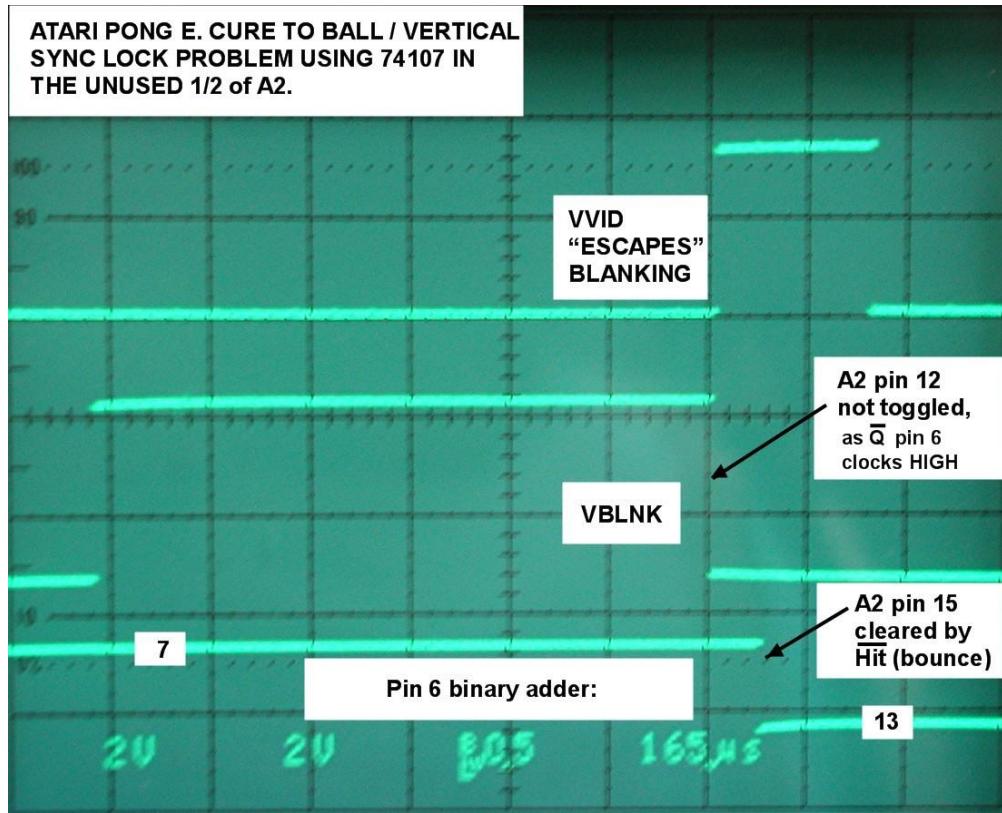
Pin 12 of A2 was previously connected to VBLNK and pin 1 & 4 were previously connected to VVID.

(The behaviour of the right flip flop is also improved as the J & K inputs are now tied high. Occasionally early JK flip flops can misbehave when the data on the J & K is changing during the clock being high (one's catching behaviour- see Horowitz & Hill, The Art of Electronics), although this is not the cause of the sync lock problem in this case).

The low going Qbar output (pin 6) toggles the right hand flip flop A2 (as J & K are high) reversing the vertical ball motion. On the next low going VBLNK pulse, the left flip flop has a high clocked to its Qbar output, only if the ball has moved away from blanking.

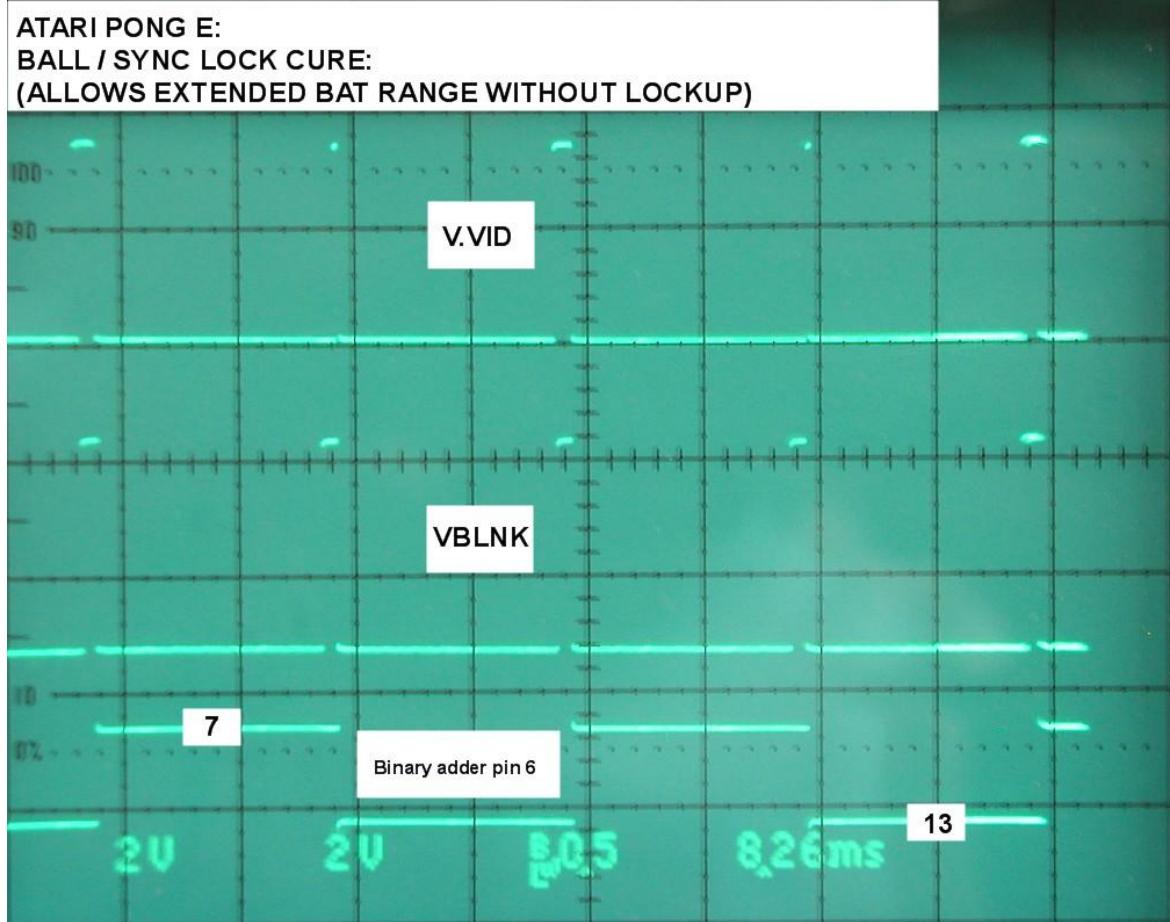
While one VBLNK pulse is used to set Qbar (low pin 6) the next one is required later to set it high after that, before it can fall low again and trigger the other A2 flip flop at pin 12. This provides the additional time for the ball to “escape” blanking and avoids the lockup condition. With this modification the bat can push the ball right through blanking without lockup or “capture” and into the next field where it remains “free”.

Below is an oscilloscope trace after this modification attempting to get the VVID to synchronise with VBLNK:



The oscilloscope above: when the bat is pushing the ball against the blanking area, at the moment the ball & bat are coincident A2 pin 15 falls low. This clears A2 and the vertical velocity encoder switches to the value encoded by 13 (fast up). With the modification, after the VVID was moving downwards in blanking (7) it has moved far enough to escape blanking. When VBLNK falls low, VVID is low (not high as it was before the modification) so the falling edge of VBLNK on the start of this field does not clock 12 pin 12 because a high is clocked to the pin 12, not a low. A2 pin 12 can only be clocked (toggled) on the next field.

The oscilloscope below shows the scan at a lower time-base rate:



The ball is being bounced at the vertical rate, by the presence of the bat and at the start of one field the ball(VVID) is elongated as usual around blanking, but at the start of the next field it has escaped blanking and is free at its normal pulse width of 254uS. If the bat is moved away the ball leaves blanking. If the bat attempts to push it further the ball jumps through blanking. For example it will appear at the other side (top or bottom) of the screen and remain “free”

Ideally the bat range is adjusted so the bat comes close to, or just reaches, vertical blanking so the ball can't slip by. If the bat can pass right across VBLNK it is possible that there could be a “10” clocked to the vertical velocity encoder and the ball could stay near vertical

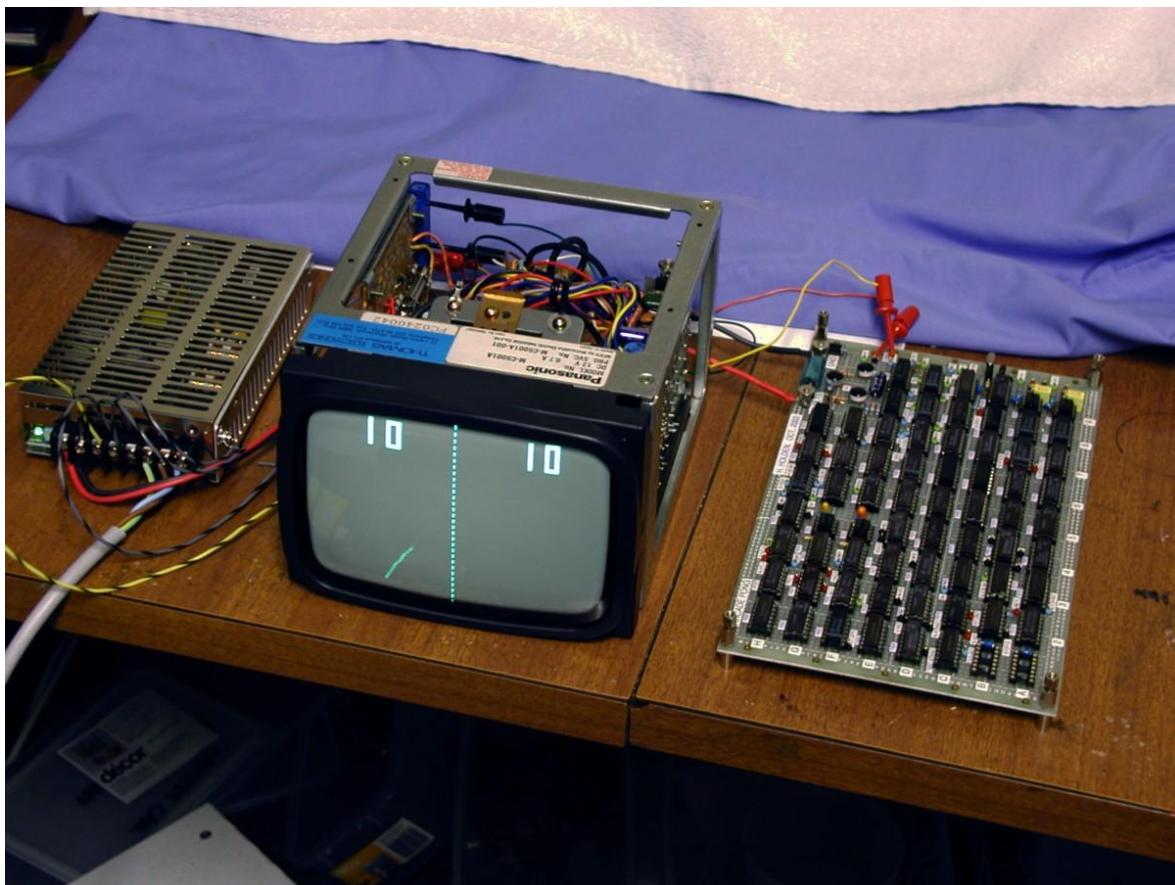
blanking with zero vertical velocity. Provided the monitor height was not excessive, and the ball out of view, then contact with the players bat would still be easy and game play as usual. Ideally the Pad1 and PAD2 bat pulses (like the ball pulse in one of the modifications) would have been gated out of the vertical blanking interval too, but this would have taken a few more gates.

MINIATURISING PONG & STILL USING TTL IC's:

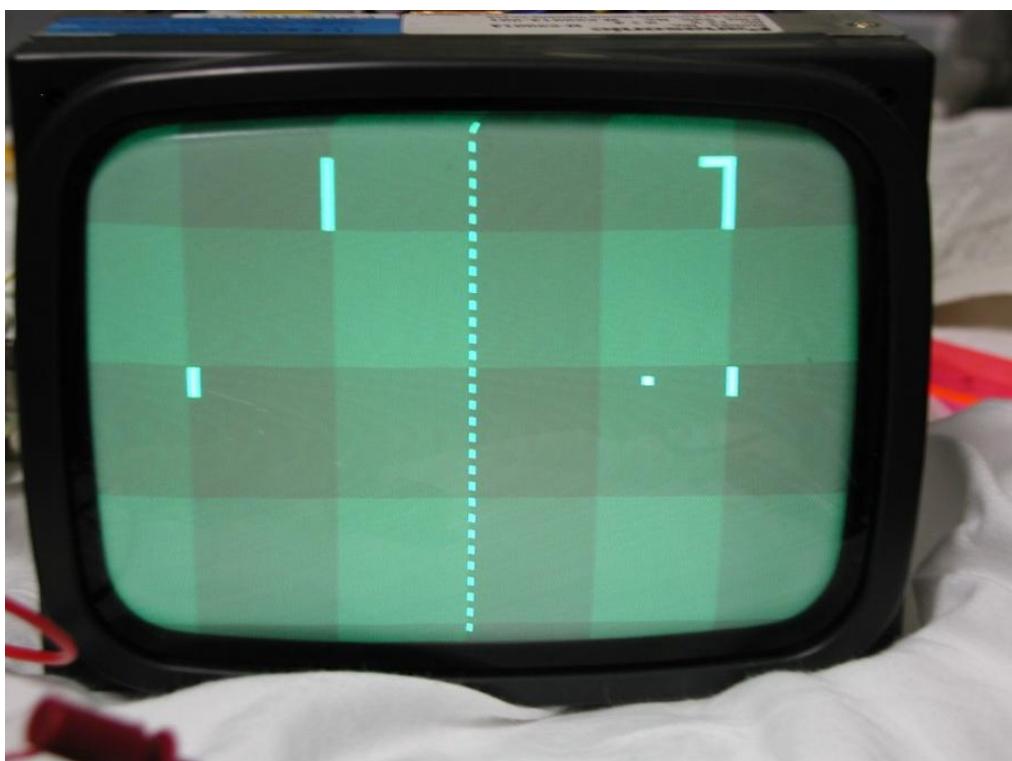
Atari's original arcade pcb was very large. The following photos are of a TTL PONG game built onto a small PCB and hand wired with over 900 connections. Some TIL311 hexadecimal displays were added to the pcb to show the hit counter and vertical velocity encoder outputs for experimental investigations.

The game was constructed using all ceramic package vintage mil spec 54 series TTL IC's which are undoubtedly the most reliable and robust integrated circuits ever made. Also signals from the H and V sync counters have been mixed into the video output to gain a patterned background to look like a mowed lawn. A green 5 inch computer VDU is used for the display. The wiring on the rear of the pcb is Teflon covered hook up wire, laced into a loom with silk thread.

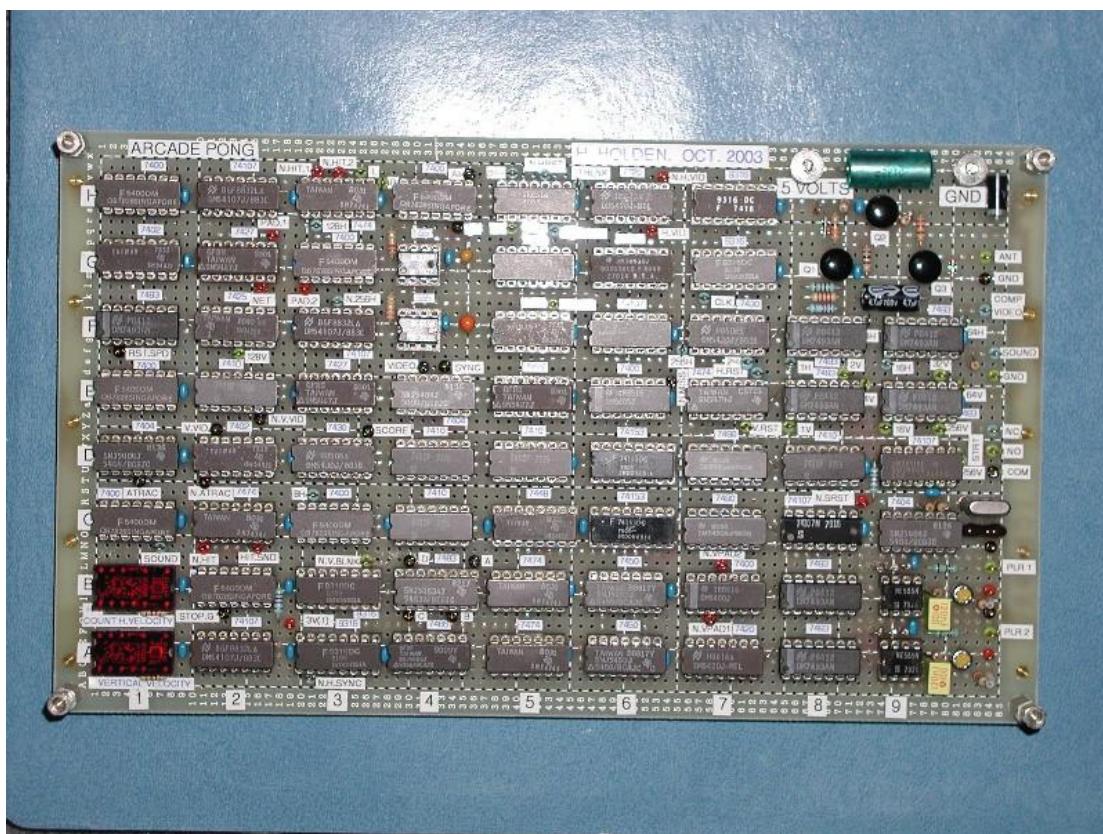
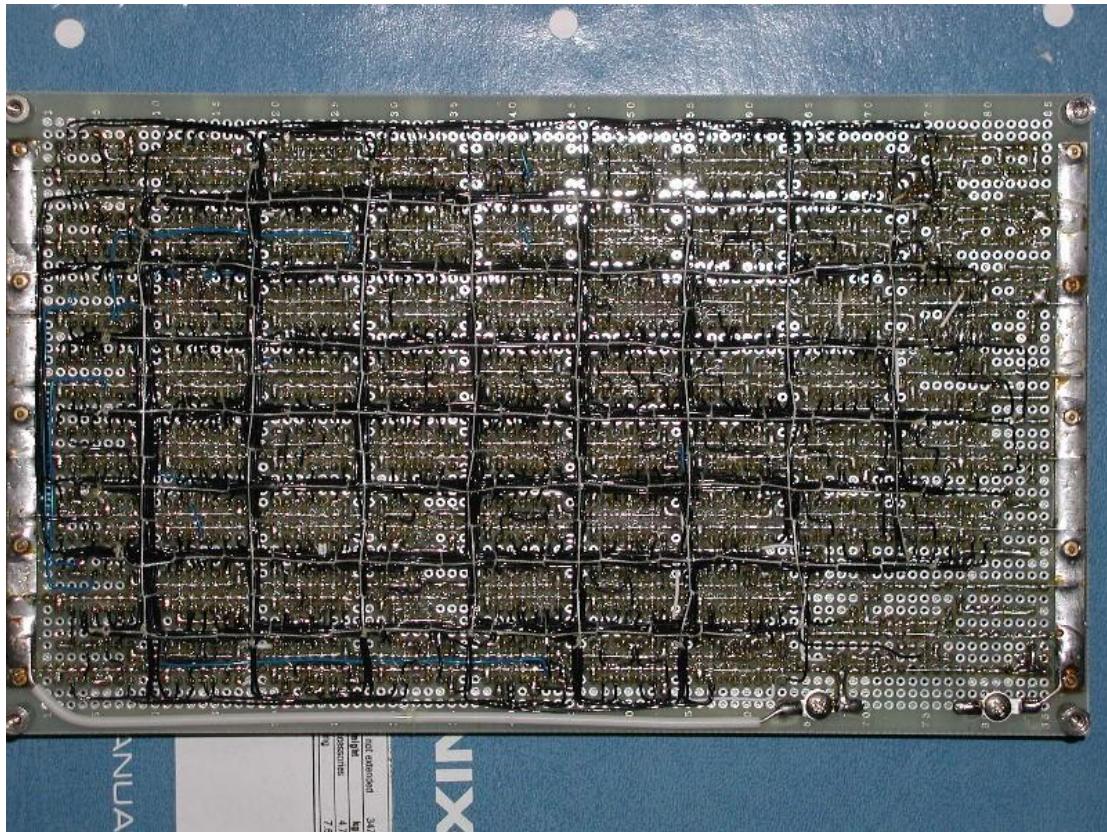
This Pong variant is called LAWN TENNIS:



The H & V motion was stopped to take the photos below:



The rear of the pcb which is 9 x 5&1/4 inches in size:



UPDATE 2013:

1) Questions & answers with Mr Allan Alcorn:

Q) Who was responsible for the design of the Pong Arcade cabinet?

A) Nolan specified the cabinet. He was tempted to use fiberglass like Computer Space but knew of the production constraints so chose a simple wooden cabinet. We were lucky to have a very capable maker in the area that could meet our production demands. Nolan wanted the cabinet do be somewhat subdued to fit in to a family location like a restaurant. His color choices were "yellow for excitement and black for mystery."

Q) The Attract mode was clever to make the game look interesting while not in use, who thought of that idea?

A) The attract mode was Nolan's idea. He told me that the controls must have no effect otherwise a child will sit there all day turning the knob. It took very few gates to make it work.

Q) Who was responsible for the design of the PCB ?

A) We were proud of the PCB layout, it was very well done. It was done by a contractor, Howard Cantin who did it by hand and was a true artist. Take a look at the layout on the Apple II PCB, it might look familiar.

Other remarks from Mr. Alcorn:

The problem you noticed about the paddle not going all the way to the top was left in because without it good players could monopolize the game. Our motto was "if you can't fix it call it a feature."

2) "The Ghost in the Machine"

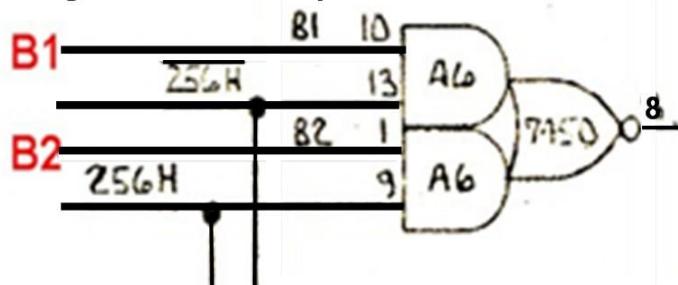
It has come to light after more than 40 years since its production that Pong E has a pcb track wiring anomaly which was not intended by the designer Mr Alcorn.

While the actual schematic of Pong E is correct showing gates & wiring, one of the IC's, A6, on the schematic had its pin numbers mislabeled. The pin 1 and pin 10 labels of IC A6 in the vertical velocity encoder circuit were switched.

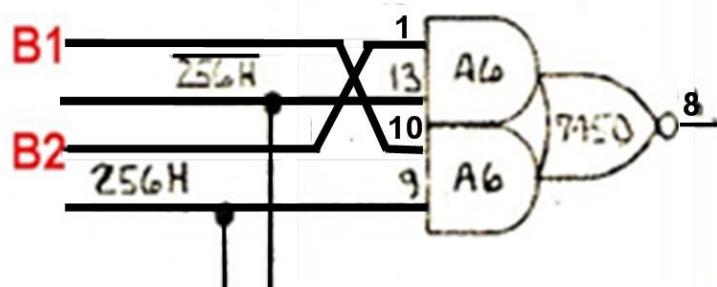
This numbering anomaly was recently detected by Mr William Buchholz who had also noticed that the interaction of the ball and paddle was not what one would expect at times.

When the pcb was created the pcb designer Mr. Howard Cantin presumably used the pin numbers to link the IC's to pin numbers on other IC's. As a result of this the paddle B data (B 1 & B2) from each player's paddle generators were inadvertently switched:

Pong E Schematic pin 10 & 1 mislabeled:



Effect of PCB design following the pin numbers:



The effect of this was significant. For example it meant that at the time of the collision of the ball with player 2's paddle, the position of player 1's paddle on the screen had an effect on the vertical velocity that the ball acquired after bouncing off player 2's paddle and visa-versa. It is analogous to a tennis player being able to move his racquet and be able to affect what was happening when his opponent was striking the ball.

In general the B data from one player's paddle generator will be high if it is above the level of the other player's paddle and low if below it. With near vertical alignment it could be high or low depending on the count within the lines that make up the paddle as the B data is alternately high and low within the paddle time.

The result is that the output of the vertical velocity encoder which corresponds to specific vertical velocities gets shifted up or down and this changes the paddle architecture. It also results in the exclusion of some of the possible vertical velocities at times and an increase of the area of the paddle, or number of lines of the paddle, corresponding to some of the vertical ball velocities.

It explains why for example, if the paddles are set up with the ball moving in horizontal motion, they are not sometimes aligned as one would expect to achieve this as the zero vertical motion zones are shifted up and down within the array of paddle lines. Also that by moving one player's paddle after the ball had left it in horizontal motion, with the other players paddle known to be in a position of horizontal motion previously, the ball will sometimes reflect with a vertical component.

Overall though, the effect was to make the game behaviour not what the designer intended, but it was still close enough at the time not to be noticed. Due to the fact that the paddle B data is the least significant bit of the 3 bit paddle data (the A data from the paddle generator was dropped) the effect was not as devastating as it would have been if the C or D data was switched.

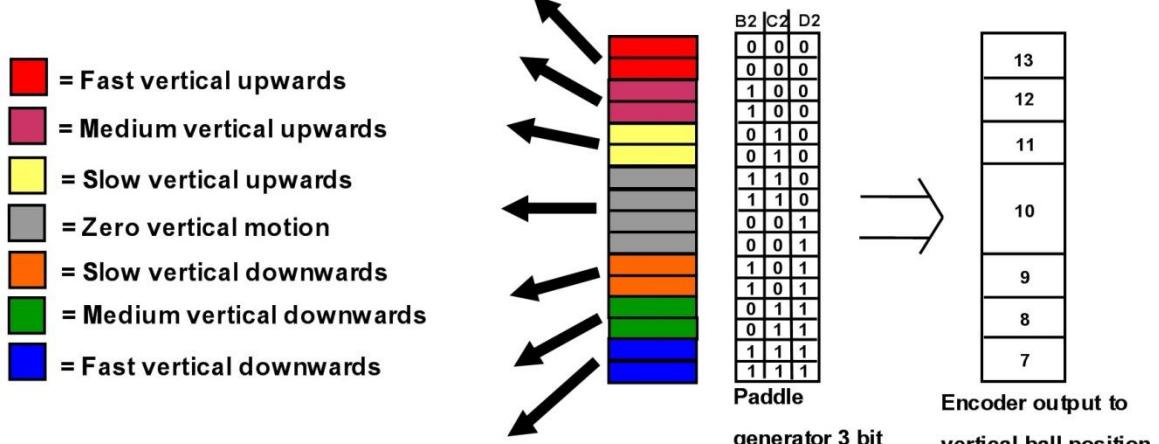
In later versions of Pong, such as pong doubles, the 7450 IC in this part of the vertical velocity encoder was replaced with a 74153 data selector IC to allow for selecting the 4 paddles. The anomaly was not present in that version.

To analyse the effects of this B data switch on the paddle data stream requires some diagrams:

Firstly to re-cap on the designer's intentions and using colors as markers for the paddles 16 lines of video:

ATARI PONG E: 16 LINE PADDLE ARCHITECTURE - DESIGNER'S INTENTION:

Vertical ball trajectories after ball & paddle 2 collision (coincident signals)



Note: Above the vertical level of the paddle, the paddle data B1, C1, D1 (or the other paddle B, C, D) is low level = 0. While below the vertical level the data is all high = 1. During the paddle time (paddle lines) the B data is exactly as indicated above in the table.

As can be seen the paddle generator data is composed of three bits. These are B2,C2 and D2 for player two's paddle, as indicated (or B1,C1,D1 for player 1).

The vertical velocities are encoded to a 4 bit binary number (represented above as its decimal value over 7 to 13) so as to control the ball's vertical position or velocity. When the signal which represents video window enclosing the ball becomes coincident with the signal representing the video window enclosing paddle, the paddle data at that moment is clocked through the vertical velocity encoder circuitry to produce the appropriate ball vertical velocity as shown in the

diagram. The intention was to give realistic game play and this feature was unsurpassed by any other paddle game at that point in history.

The following series of diagrams show what happens when the B data is switched:

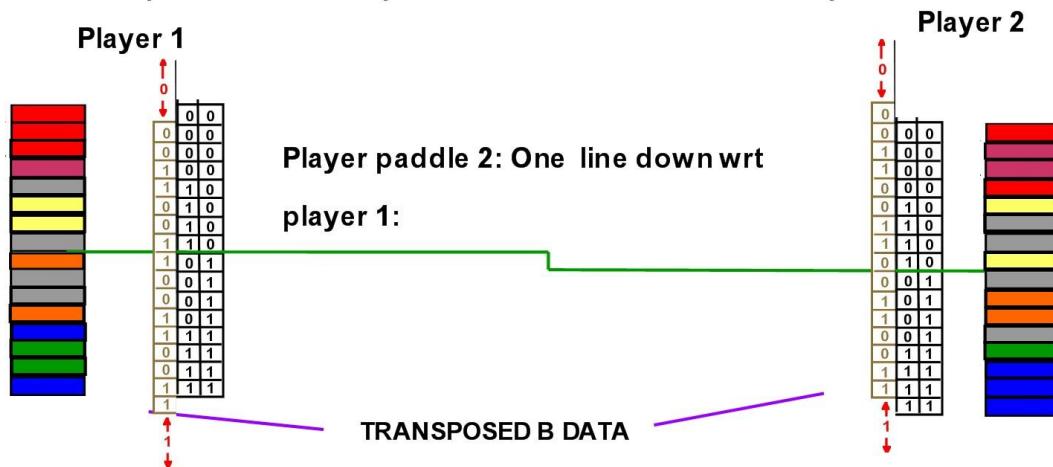
B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:



With perfect vertical alignment the paddle functional architecture is not changed.

With one horizontal line displacement:

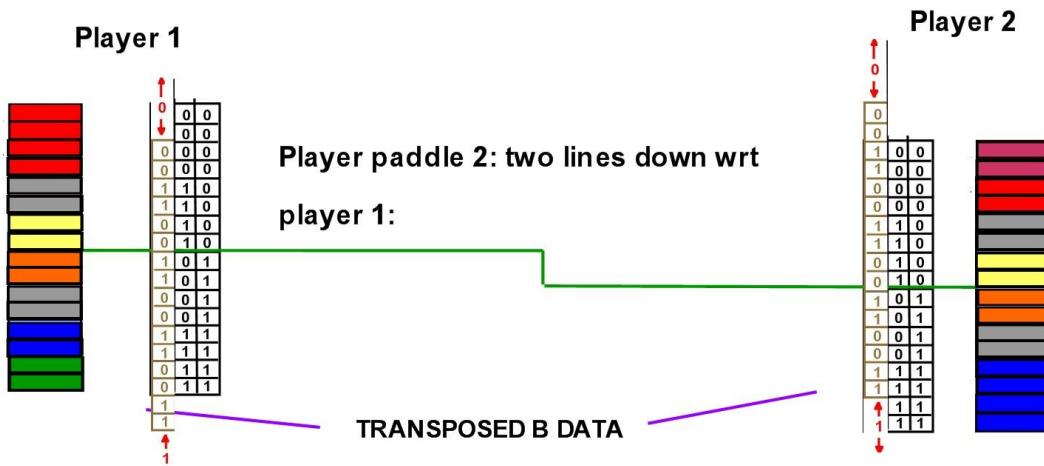
B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:



The paddle architecture is now jumbled. The player paddles are complimentary images of each other in that the complimentary up vs down vertical velocities, eg blue/red and orange/yellow etc are stacked in a similar format and inverted. The zones of the paddle corresponding to zero vertical motion (eg only horizontal ball motion) are spread apart in the paddle center. At this point all the possible vertical velocities are still present but the probability of medium vertical upwards motion (magenta color) and medium vertical downwards motion (green) are reduced in favor of fast vertical upwards motion player 1 or fast vertical downward motion player 2.

With a two line displacement:

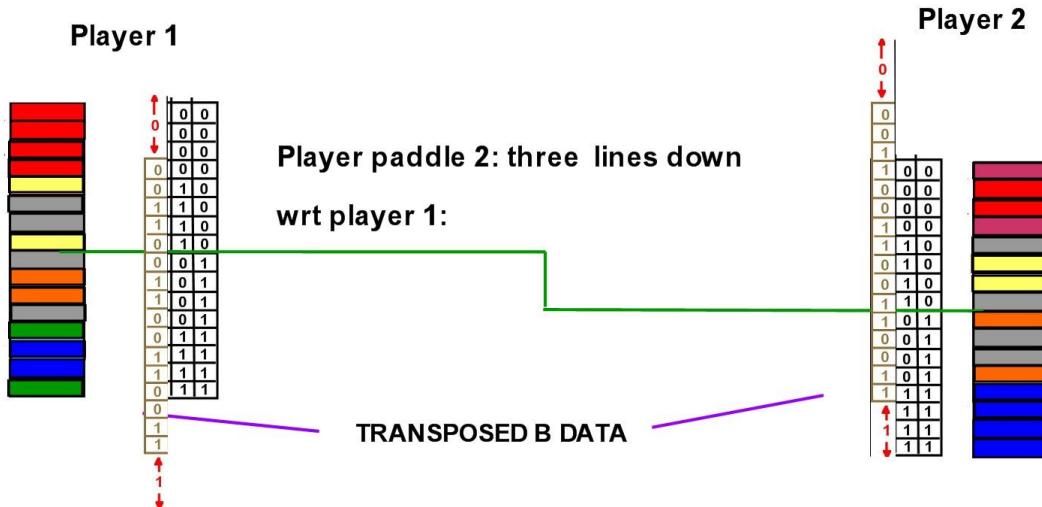
B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:



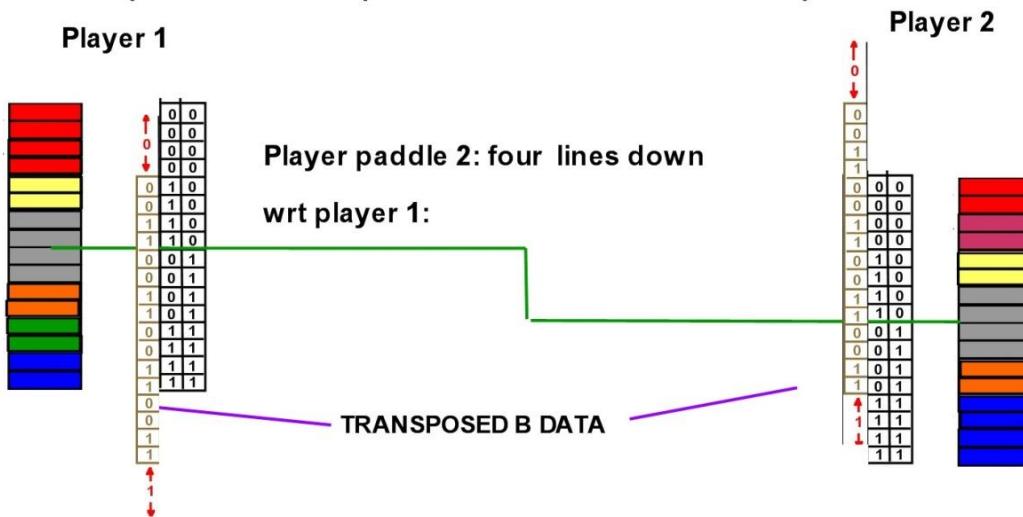
With the two line displacement a possible state of vertical motion from each player's paddle vanishes in favor of either fast upward motion player 1 or fast down player 2. Again the zero motion state is spread away from the paddle center which is now occupied by the complimentary slow vertical motion states (orange & yellow).

Moving on to 3 lines displacement or 4, 5, 6 etc various combinations crop up. Up to nine lines of displacement is shown below. After 16 lines displacement the paddles no longer share any vertical alignment and the paddle architecture settles on a final result.

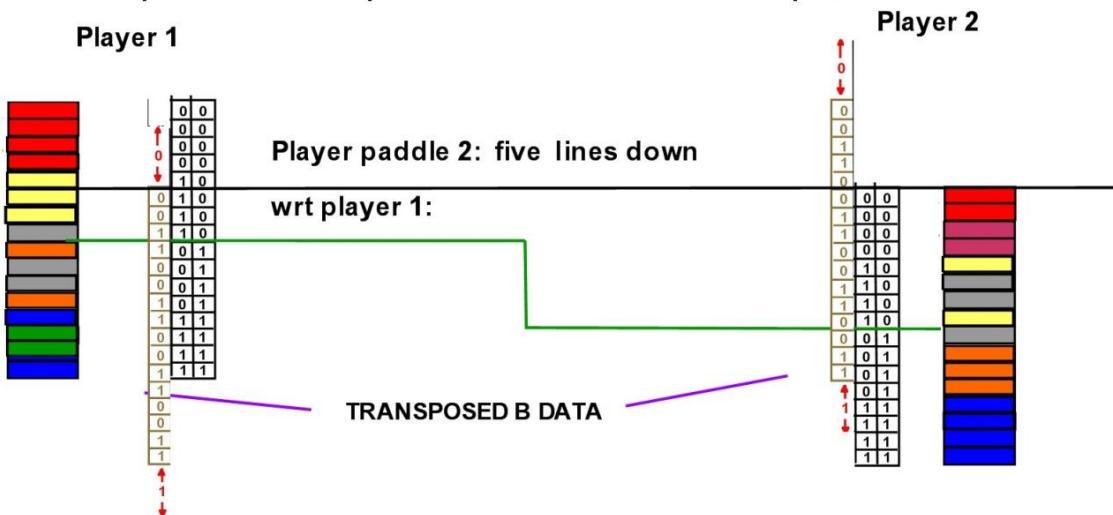
B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:



B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:

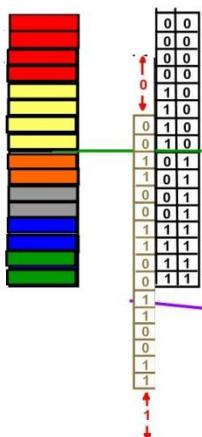


B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:



B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:

Player 1

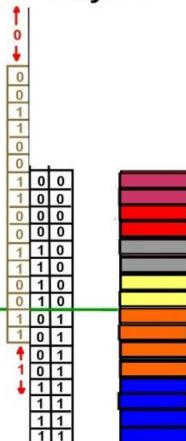


Player paddle 2: 6 lines down wrt

player 1:

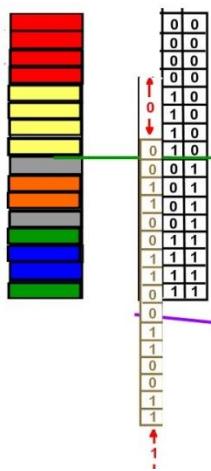
TRANSPOSED B DATA

Player 2



B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:

Player 1

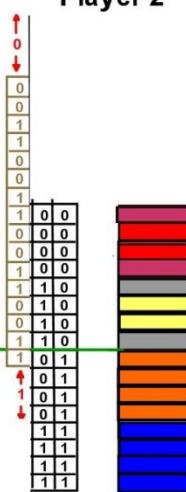


Player paddle 2: seven lines down

wrt player 1:

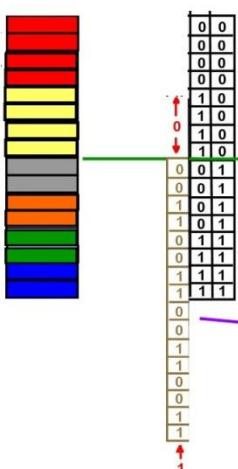
TRANSPOSED B DATA

Player 2



B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:

Player 1

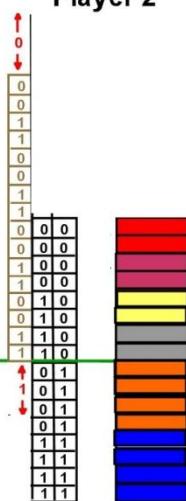


Player paddle 2: eight lines down

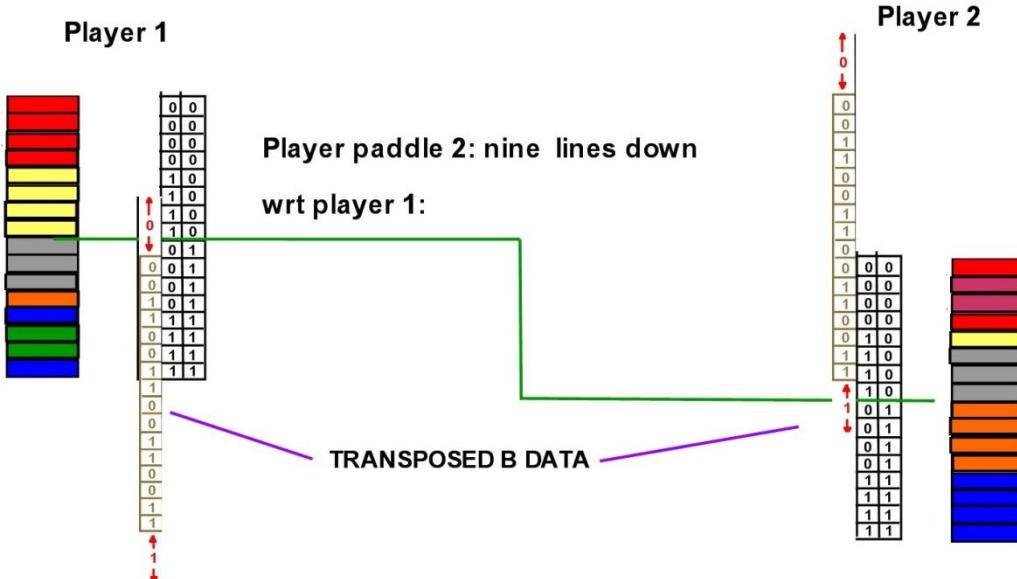
wrt player 1:

TRANSPOSED B DATA

Player 2

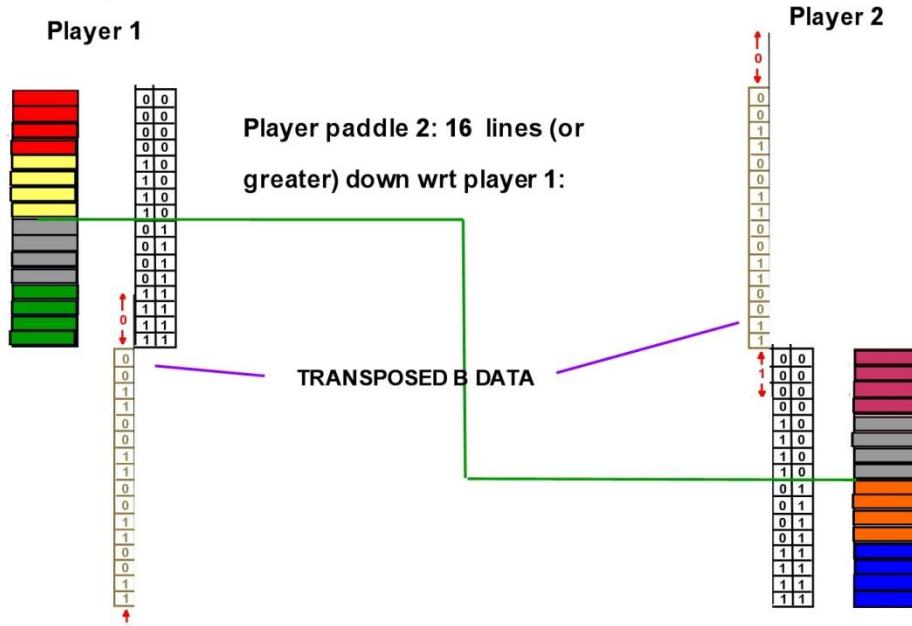


B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:



As can be seen the paddle overlap with transposed B data results in multiple paddle functional architectures. When the paddles are not sharing vertical alignment, with an offset of 16 H lines or greater, the following occurs:

B 1 and B2 paddle data transposition scenarios and effect on paddle functional architecture:



This would be the status during most game play time when the paddles were not aligned. Each paddle now has only 3 possible states of vertical motion occupying a larger paddle area, with the centre or zero vertical motion state is either above the paddle centre (lower paddle) or below the paddle centre (upper paddle).

The diagrams explain why this anomaly was not picked up at the time of the manufacture of the Pong E PCB. The reason is that the overall paddle architecture still had up going vertical velocity of one kind or another, mostly in its upper parts, and down going vertical velocity of one kind or another mostly in its lower parts and a zone somewhere near or around the centre of the paddle which gave horizontal only motion. The fact that some of the vertical velocity possibilities were not there, especially with paddles not in vertical alignment, was simply not noticed and the game appeared to play as designed.

If there are any errors detected in this article or corrections that the reader thinks should be made to this article, please email me with as much supporting technical information as possible outlining the case for the corrections.

Dr. Hugo R. Holden.

UPDATED Jan 2013.

Hugo.holden9@gmail.com
