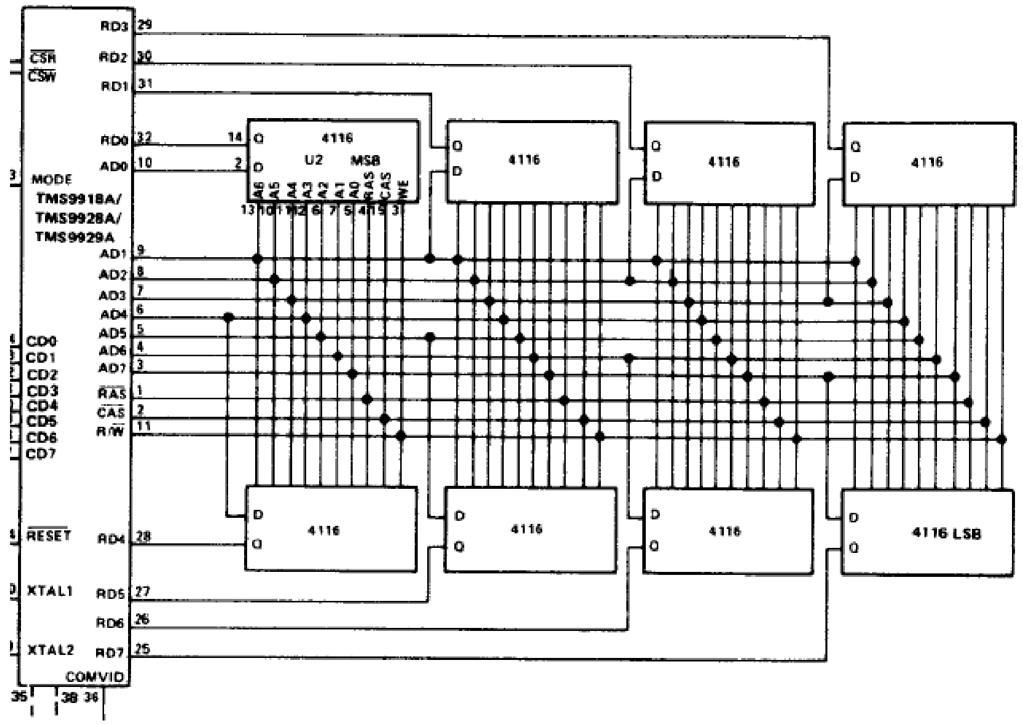
SRAM Replacement for TMS99x8 VRAM

The TMS99x8 Video Processing Unit is intended to directly interface with a bank of 4116 type DRAM (16K x 1 bit). The drawback is that 'package density' is quite poor by modern standards; eight DIP16 packages to implement 16KB of memory. Furthermore, the 4116 DRAM requires a negative voltage bias rail, which is typically no longer present in modern-day designs (and if so, not intended for logic use).



Therefore, replacing a bank of eight 4116 DRAM packages with a single SRAM package is beneficial. This document describes an implementation that has been tested by several builders, using TMS9918 and TMS9928 devices. The intent of this writeup is to describe the circuitry, as well as present some observations that were made along the way to help others that might want to implement same.

SRAM Selection

A logical choice for SRAM is a 32Kx8 "cache RAM" from older PC motherboards (e.g. 80386 and 80486-based). These SRAMs are very fast (more than adequate for use with the TMS99x8), are TTL compatible, and are readily found at very low prices. They are typically in 0.3" pitch ("skinny") DIP28 packages, and have part numbers like "60256" or "62256". Avoid parts that have the letter 'V' or 'W' sandwiched in the part number (e.g. 62V256) as these are *typically 3.3V parts*.

Address Demultiplexing

The TMS99x8 is intended to directly interface with 4116 DRAM, and DRAM uses a multiplexed address bus, divided into a "row address" and a "column address". To address a location within the DRAM, first the row address is driven onto the DRAM address lines and then the /RAS (Row Address Select) signal is asserted (driven low). Next, the column address is driven onto the DRAM address lines and then the /CAS (Column Address Select) signal is asserted.

In contrast, the 32Kx8 SRAM does not have a multiplexed address bus, and each address line must be driven during the entire read cycle. If the /RAS and /CAS signals are inverted and called ROW and COL, respectively, an octal D type Flip-Flop (e.g. 74LS574) could be used to capture the contents of AD1-AD7 on the rising edge of ROW and COL, thus yielding a set of demultiplexed address signals to apply to the SRAM A0-A14 inputs.

Because the address space is 16K, only 14 bits are required, and so the signals that must be applied to the D FF inputs are AD1-AD7. Furthermore, SRAM does not require any sort of refresh, so the relationship between the TMS99x8 AD1 and AD7 bus and the SRAM A0-A14 address inputs can be completely arbitrary. Whatever works well in terms of aesthetics and PCB layout is acceptable.

VRAM Read Timing Analysis

The VRAM read timing diagram from the TMS9918 VDP datasheet illustrates the relationship between all of these signals during a VRAM read cycle.

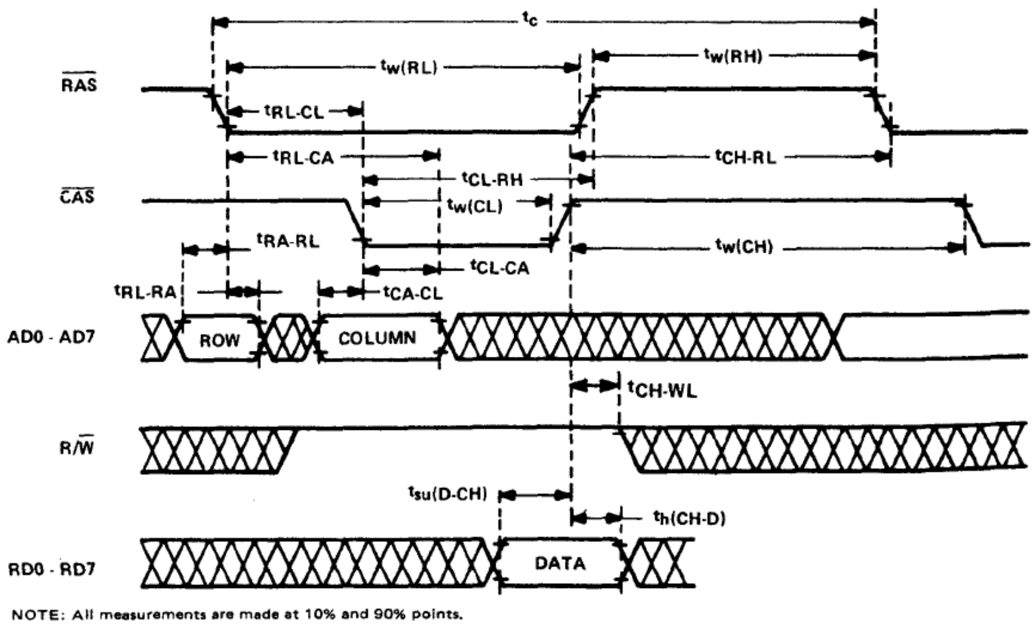


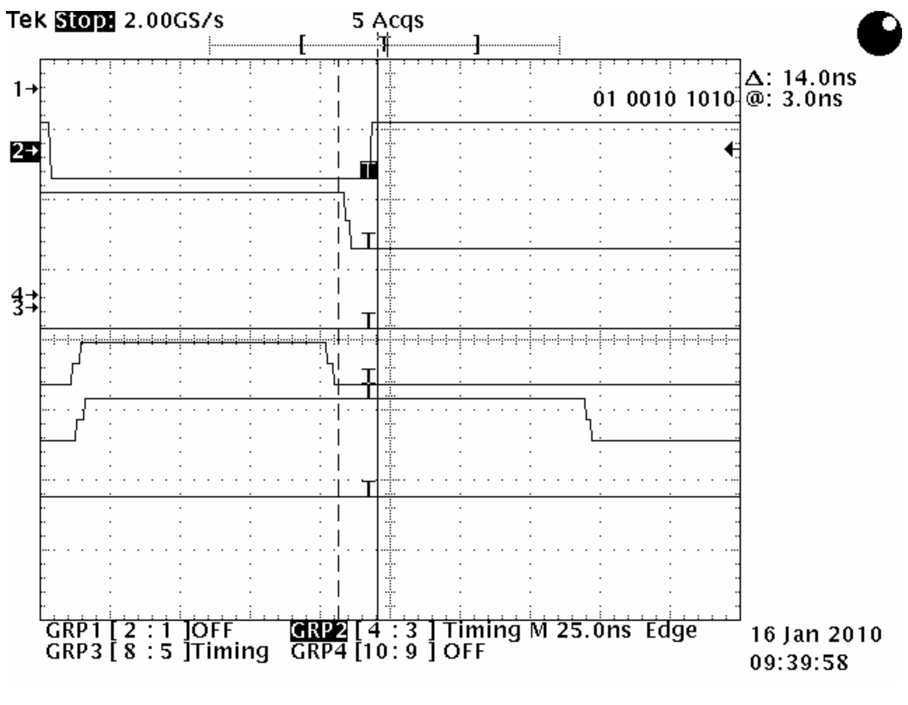
FIGURE 5-6 – VRAM READ CYCLE

To properly implement address demultiplexing, care must be taken that the AD1-AD7 signals are stable prior to the rising edge of the ROW and COL signals (i.e. conform to the input setup time of the D FF). For a 74LS574, the setup time on the D0-D7 inputs prior to CLK rising edge is 20ns.

First, consider the row address timing. The TMS9918 specification for $t_{RA\text{-}RL}$ is between 20 and 30ns, and so the additional t_{PD} of an inverter (to make ROW from /RAS) ensures that in all cases the 20ns setup time of the 74LS574 is met. For a 74HCT574, the setup time is 12ns, so once again there is no possibility of a violation.

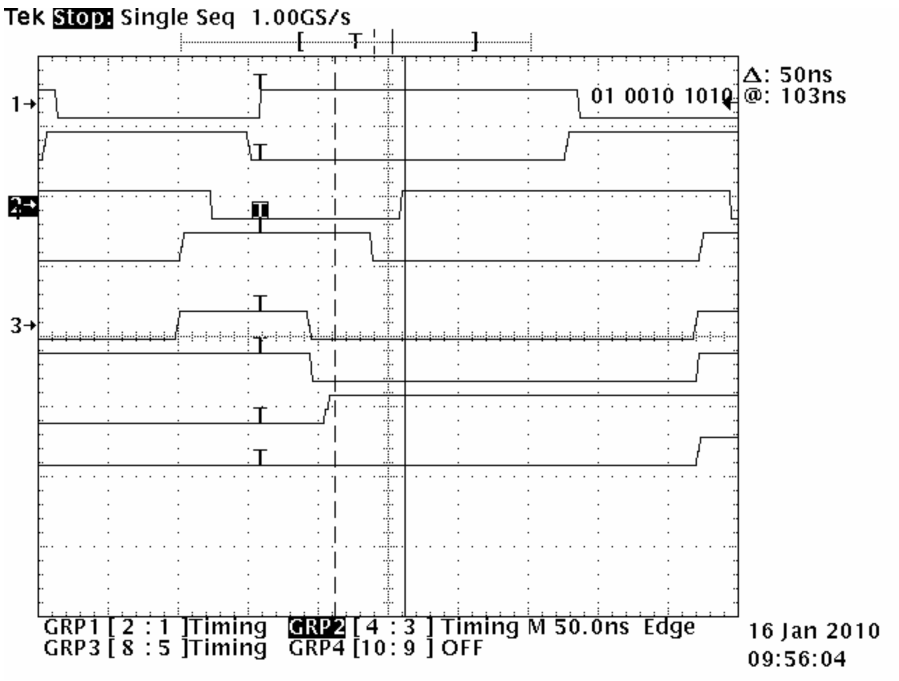
Next, consider the column address timing. The TMS9918 specification for $t_{CA\text{-}CL}$ is no less than -10ns, with a typical of -2ns. The negative value means that /CAS is falling *prior* to the AD1-AD7 lines being stable! If a 74LS04 inverter is used to create COL from /CAS, its t_{PLH} of between 9ns and 15ns brings the worst-case setup time prior to COL rising to -1ns.

A snapshot from a logic analyzer shows this issue clearly. The traces are (from top to bottom) COL, /CAS, and AD4-AD7. The cursor lines are positioned to illustrate that in this particular case, there are only 14ns of setup time from AD7-AD0 becoming stable (in this case AD5 falling) and the rising edge of COL.



This situation resulted in improper TMS9918 operation on my prototype; quite frequently the row address would appear on the outputs of the column address latch. Clearly what was needed was more delay between falling edge of /CAS and rising edge of COL. I accomplished this by creating COL from /CAS using *three inverter stages* in series, relying upon the propagation delay of the additional two inverters to ensure the necessary setup time.

The analyzer snapshot below shows the improved timing, the traces (from top to bottom) are ROW, /RAS, COL, /CAS, and AD4-AD7.



The time from AD7-AD0 settling and COL rising is now approximately 50ns; adequate to meet the setup of the 74LS574 D-FF. Because of this critical timing, *it is advisable to use the same logic family for the inverters and the row and column address D-FF devices*.

SRAM Read Access Time Requirements

When reading from VRAM, the TMS9918 requires $t_{SU(D-CH)}$ of setup time during which the data on the RD7-RD0 inputs must be stable prior to the rising edge of /CAS. The specification for $t_{SU(D-CH)}$ is 60ns minimum. Refer to the analyzer snapshot we used for verification of the 3-inverter COL delay. We can see that the column address is clocked about 210ns prior to the rising edge of /CAS. A cache SRAM typically has 70ns access time or better, and the worst-case t_{PD} of the 74LS574 is 28ns. Taking all timing into account, the setup time of the RD7-RD0 inputs is met in all cases (210ns - 60ns - 70ns - 28ns = 52ns of margin) with LSTTL logic, even more so with HCT logic.

VRAM Write Cycle Analysis

During a VRAM write cycle, the AD7-AD0 bus will contain the row address, the column address, and finally the write data. The /RAS, /CAS, and R/W signals coordinate the transfer into DRAM. The VRAM write timing diagram from the TMS9918 VDP datasheet shows the signal behavior during the cycle.

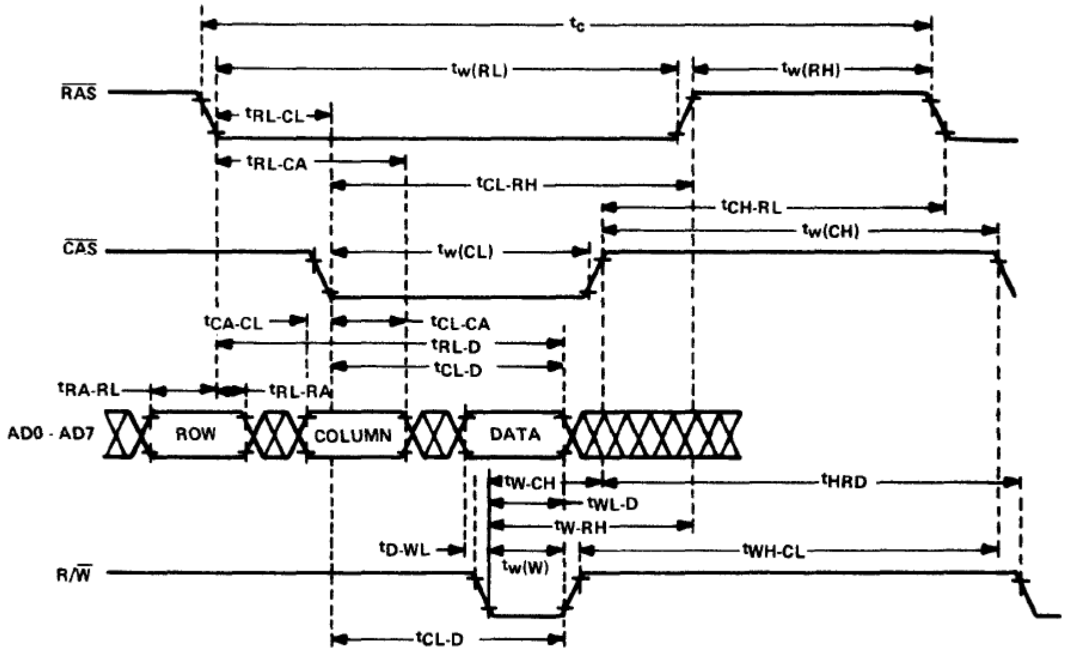
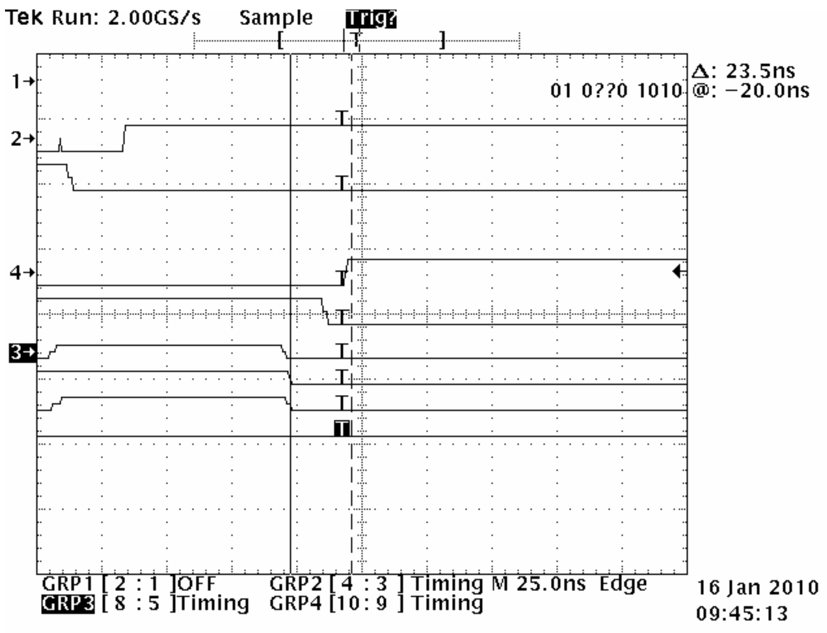


FIGURE 5-5 – VRAM WRITE CYCLE

It is not clear if the AD7-AD0 signals are not guaranteed to be stable throughout the entire write cycle: $t_{w(W)}$ is specified to be between 170ns and 210ns, and $t_{WL\text{-}D}$ between 135ns and 195ns. So, yet another octal D-FF connected to AD7-AD0 is required, clocked after the write data is stable on AD7-AD0. We need to make a signal with a rising edge, call it WR, by inverting R/W.

This logic analyzer snapshot taking during a write cycle shows how all these signals work together; (from top to bottom) COL, /CAS, WR, R/W, and AD4-AD7.

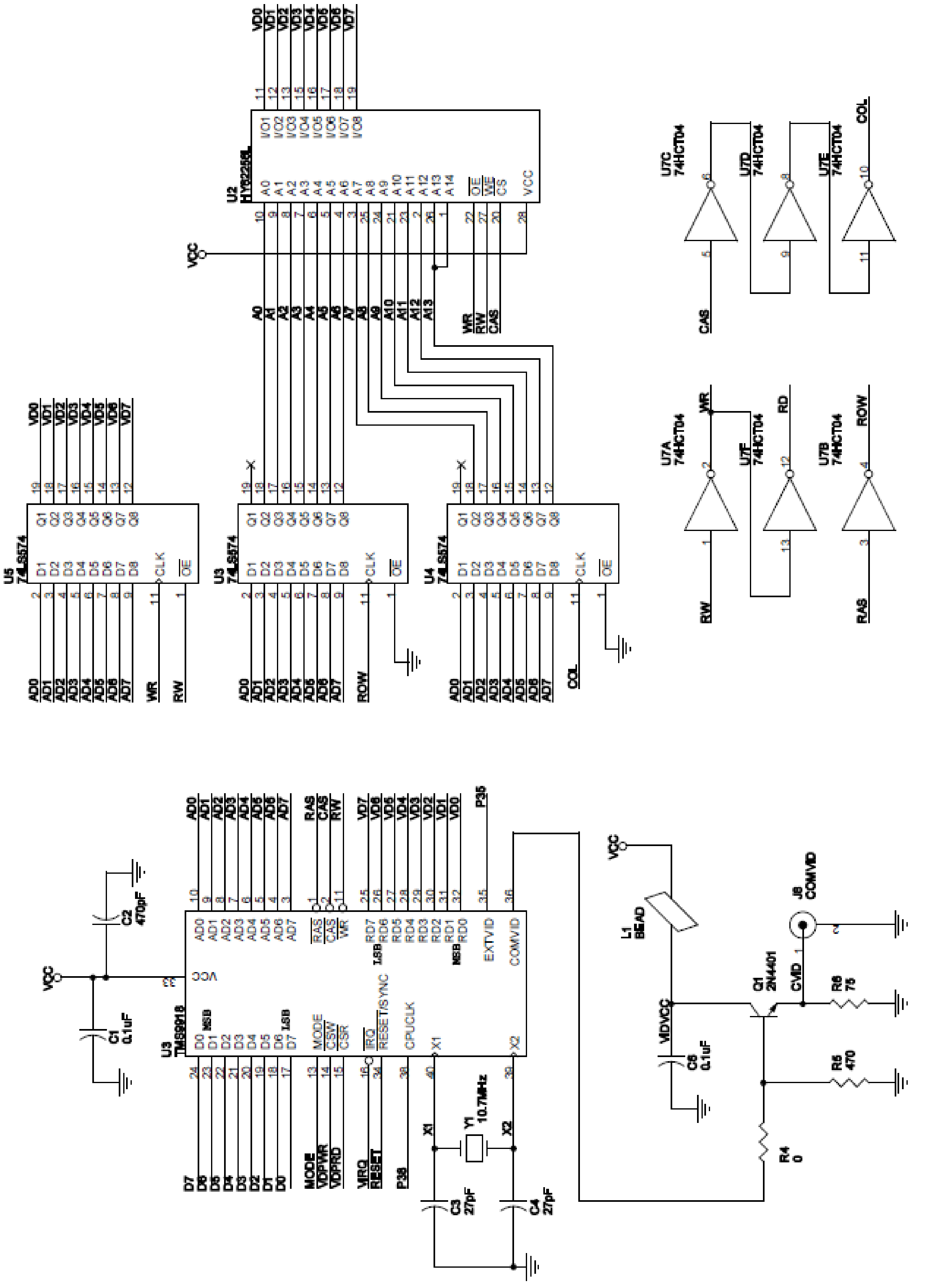


Here, the cursors are positioned on the rising edge of WR relative to AD4-AD7 becoming stable, showing 23ns in between.

The specification for t_{D-WL} is between 0ns and 20ns, and a 74LS04 inverter will add between 9 and 15ns. The 20ns setup time requirement for the 74LS574 input is 'cheated' by 5ns with worst-case timing, so adding two additional inverter stages to provide more setup time would guarantee operation. I used a 74HCT574 D-FF and one section of a 74HCT02 as an inverter, so the shorter HCT setup time was 'cheated' by only 3ns. Using an LSTTL inverter and an HCT574 D-FF could be another possible workaround.

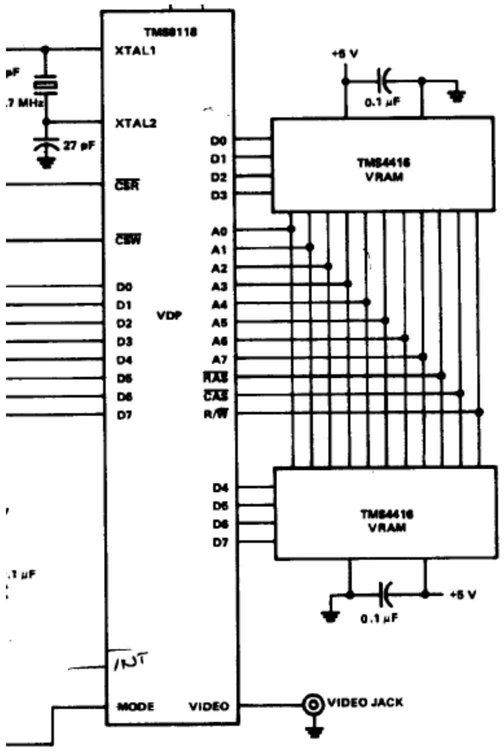
Finally, recall that there was no need to pay any special attention to the relationship between AD7-AD1 as applied to the row and column address flip-flops and A0-A14 of the 32Kx8 SRAM. This is *not the case*, however, with the relationship between AD7-AD0 and RD7-RD0. More specifically, there are no special requirements as to which SRAM D0-D7 terminal is connected to which RD[n] and which D-FF Q[n] line, but it is absolutely *critical* that there is a path between corresponding AD[n] and RD[n} terminals of the TMS99x8! The unorthodox weighting that TI used for the TMS9000 processor and peripherals makes it easy to get lost and make an error in connections across the various busses. (Ask me how I know this…I spent literally *days* trying to figure out why my test code resulted in only garbage on the screen!)

SRAM Replacement Schematic



SRAM Replacement for TMS91x8 Video DRAM

The TMS91x8 devices were released after the initial TMS99x8 family members. The only noticeable difference between the TMS91x8 and its TMS99x8 counterpart is in the VRAM interface. The TMS91x8 devices were redesigned to interface to a pair of TMS4416 (16Kx4) DRAM, rather than a bank of eight 4116 DRAM. The TMS4416 does not suffer from the drawbacks of the 4116: two DIP18 packages versus eight DIP16 for a 16KB array, and no need for a negative bias rail. To accommodate the DIP18 package, the TMS4416 has four common data input and output terminals; there are no longer separate DI and DO terminals. This, in turn, means that the rather odd interface arrangement between the TMS9918 and the 4116 DRAM cannot be used with the TMS4416 DRAM. What Texas Instruments did with the TMS91x8 was to simply make the RD7-RD0 bus a bidirectional data bus (rather than input only bus), and no longer place the write data on AD7-AD0 during a VRAM write cycle.



The motivation to replace two TMS4416 DRAM with several D-FF and an SRAM is somewhat reduced, but the good news is that the TMS99x8 SRAM Replacement circuitry works with the TMS91x8 device also. The only modification required is that the Write Data D-FF is no longer required, and so if this FF is in a socket, it can simply be removed for compatibility with a TMS91x8. The row address timing remains the same, and the column address timing improves significantly. Therefore, the TMS99x8 and TMS91x8 can be used interchangeably when this SRAM Replacement technique is used.

Acknowledgements

I would like to thank Andrew Lynch, creator of the wonderful N8VEM Z80 SBC project, for the motivation to work on the topic of SRAM replacement for the 4116 VRAM of the TMS9918. This had been on my own “back burner” for many years until he gave me the ‘nudge’ I needed to get working on it. I would also like to thank Dan Werner, who started working on this topic around the same time, and arrived at a working prototype while I was still trying to get my own PCB working. Dan provided a good deal of moral support and things to think about when debugging. Dan and I eventually ended up with the same circuit through our collaboration, which Andrew ultimately ended up building also.

Tom LeMense
tlemense@yahoo.com
January 24, 2010