EE 52

Spring 2017

Digital Oscilloscope Documentation

Sophia Liu

December 16, 2017

Contents

1 User Manual

1.1 System Description

The System on Programmable Chip (SoPC) Digital Oscilloscope is an FPGA/microprocessor-based system capable of capturing and displaying up to 5 MHz signals. The analog input can range from 0 V to 3.8 V. The system has all the features of a standard oscilloscope with the exception of input signal scaling. Two rotary encoders are used to control the settings, and a LCD display is used to display the captured signal.

1.2 How to Use the System

The system will begin immediately upon powering up. [1] The system starts in one-shot trace mode with a sampling rate of 100 ns, a mid-level trigger (halfway between the minimum and maximum trigger levels, at 1.9 V) with no delay and positive slope, and with the menu displayed and scale set to axes. The probe can be attached to the desired input source. The oscilloscope settings are described in section 1.3. The reset button can be used to restart the system.

1.3 System Settings

1.3.1 Menu

The scope parameter menu is located in the upper right corner of the LCD and contains the following entries in order:

```
Mode
Scale
Sweep
Trigger
    Level
    Slope
    Delay
```

The menu display can be toggled on and off by pressing the menu rotary encoder. The user can cursor to any of the entries by turning the menu rotary encoder and can change a setting by turning the secondary rotary encoder. Changes take effect

[1] The serial configuration device currently does not work, so the FPGA must first be programmed through the JTAG debugger.

immediately. More rotary details can be found in section 1.3.2. The menu entries are described in more detail below.

Mode : The Mode menu entry can be set to Mode Normal, Mode Automatic, or Mode One-Shot. In Mode Normal the scope waits for another trigger after every retrace. In this mode, new traces are captured as fast as the scope can redraw the screen. Mode Automatic works the same as Mode Normal if there are trigger events. But if no trigger event occurs after a specified delay (typically significantly longer than the time represented by a screen of data) the scope triggers automatically without a trigger event occurring. In Mode One-Shot the scope triggers only once and then holds that trace on the screen. It does not look for another trigger event until the Trigger menu item is selected and secondary rotary encoder is turned.

Scale : The Scale menu entry can be set to Scale Axes, Scale Grid, or Scale Off. If the scale is set to Scale Axes, the x and y axes are displayed along with the trace. If the scale is set to Scale Grid, an x-y grid is displayed along with the trace. If the scale is set to Scale Off, no axes or grid are displayed.

Sweep : The Sweep menu entry sets the sweep rate (in time per sample) for the scope. Possible settings are: 100, 200, and 500 nanoseconds, and 1, 2, 5, 10, 20, 50, 100, 200, and 500 microseconds, and 1, 2, 5, 10, and 20 milliseconds (per sample).

Trigger : The Trigger menu entry re-arms the trigger for the scope in one-shot mode. Any time it is selected and the secondary rotary encoder is turned the scope trigger is re-armed and a new trace will then be captured once the trigger conditions (level and slope) are met.

Level : The Level menu entry sets the trigger level. It can be set to any value from the most negative input voltage to the most positive in 128 steps. Additionally, the trigger level is displayed as a line on the screen when the trigger level is being changed.

Slope : The Slope menu entry is either Slope + or Slope - and determines whether the scope is triggered on a positive or negative slope respectively.

Delay : The Delay menu entry determines the trigger delay. It sets the time after the trigger event at which the trace will start. It may be set to any value from the minimum delay to the maximum delay times the sample rate and it is displayed as a time.

1.3.2 Rotary Encoder Functionality

The user can change the scope configuration via two rotary encoders (seen in **??**). All of the scope parameters are set via an on-screen menu. The rotary actions are detailed below:

Press encoder 1 : Turns the menu on/off. If the menu is off it is not displayed and turning the rotary encoders have no effect on the settings.

Turn encoder 1 CW : Moves the cursor down, if not already at the bottom menu item. If at the bottom, the cursor does not move.

Turn encoder 1 CCW : Moves the cursor up, if not already at the top menu item. If at the top, the cursor does not move.

Turn encoder 2 CW : Changes the currently selected (with cursor) menu item. Goes ”forward” through the list of possible settings. If at the ”end” of the list, doesn’t change the current selection.

Turn encoder 2 CCW : Changes the currently selected (with cursor) menu item. Goes ”backward” through the list of possible settings. If at the ”beginning” of the list, doesn’t change the current selection.

2 Technical Documentation

2.1 Hardware

2.1.1 Hardware System Overview

The oscilloscope system is centered around a Cyclone III FPGA, which includes a Nios II CPU and controllers for the VRAM, LCD, rotary encoders, and analog input and triggering. Upon startup, the bitmap is loaded from the serial EEPROM. The is then loaded from the ROM.

Receiving a Signal: When the probe is connected to a desired input, the analog signal is sent to an Analog to Digital Converter chip. 8 bits of data are then sent in parallel to the FPGA to an analog controller.

The various trigger setting inputs (auto triggering, trigger enable, trigger slope, trigger level) are used along with the 8 bit signal to determine if the system should begin sampling. Once triggering has occurred, the signal is written to a FIFO buffer at the sampling rate. Once the FIFO is full, the CPU clocks out and stores the buffer data to be displayed.

Displaying a Signal: In order to display the signal on the LCD, data is written to the VRAM, which is then sent out to the LCD. Several control signals (write enable, chip select, address, wait) are sent from the CPU to the VRAM controller to perform read and write operations on the VRAM. A (moore??) state machine is used to generate

2.1.2 Power

A din4 connector (PWR1) is used to supply the board with +/- 12 V and 5 V. All chips are locally bypassed (...?)

5 V is used to power the reset, VRAM, SRAM, ROM, and ADC chips.

Several LM1086 regulators are used to generate the other required voltages. Specifically, U14 uses the 5 V supply to generate 2.5 V for analog power to the FPGA PLLs, and U15 uses the 5 V supply to generate 1.2 V for FPGA power supply and digital power to the FPGA PLLs. U16 generates 3.3 V from 5 V, which is used to power the FPGA I/O supply pins, buffers, JTAG, serial configuration device, clock, and for pulling high the rotary encoders. (??)

Regulators also generate 5 V from 12 V (U24) and 4 V from 5 V (U23) for ADC reference voltages. (more analog stuff in analog section?)

A boost converter is used to generate the 25 V necessary for the LED backlight. U21 (part number AP3012) steps up from 5 V and outputs 25 V. (more detail? was used because?)

2.1.3 Analog System

The analog input can range from 0 to 4 V. An analog-to-digital converter (U17, TLC5510A) was used to digitize the input signal to 8 bits. This was chosen because of the simplicity of the circuit needed in order to meet the 0-3 V requirement.

Several analog supply voltages were used to isolate the analog and digital circuitry. An external 4 V analog reference was generated and used for a 0-4 V range for the input signal. A 5 V analog supply voltage was also required, along with an analog ground reference connected to digital ground through an inductor.

Analog logic:

Analog inputs: Several values are taken as outputs from the software and are used in the hardware logic. Specifically, an 8 bit trigger level, 1 bit auto triggering, 1 bit trigger slope, 1 bit trigger enable, 19 bit sampling rate, and 16 bit trigger delay are used.

Trigger logic: First, the system determines whether or not a trigger has occurred. A Moore state machine is used to determine whether or not a trigger has occurred. It sets the the trigger event signal high when the trigger slope is positive and the signal has transitioned from below to above the trigger level, or if the trigger slope is negative and the signal has transitioned from above to below the trigger level. The code can be found in Appendix **??**.

Thus, the 8-bit signal is compared to the trigger level, and the less than and equal to outputs are sent to the trigger state machine along with the trigger slope.

The system also generates a trigger event if auto trigger is enabled after 512 clocks, which was arbitrarily chosen as a short amount of time.

A counter and comparator are used to create a clock at the given sampling rate. Once a trigger event has occurred, another counter and comparator are used to create the trigger delay. The signal is then written into a FIFO buffer at the sampling rate, which is clocked out through the software when it is full. The FIFO has a size of 480 words, or the length of the LCD.

2.1.4 Rotary Encoders

Two rotary encoders are used for the user input. These include and A and B output for the encoder and a single pull single throw push button switch.

Pins A and B are pulled high[2], while the common pin is grounded

[2] 1 K resistors were used in place of the 10 K resistors on the schematic because the voltage high output was too low for the buffers.

to create out-of-phase pulses from outputs A and B.

Rotary decoder logic: The rotary logic block consists of a debouncer for the push button and a decoder for the rotary encoder.

The debouncer (AnalogDebounce) takes as inputs the active-low button input (Rotary1_P and Rotary2_P on the schematic), clock, and active-low reset signal. It outputs a single active-high clock pulse whenever the button input signal is low for longer than 100000 clocks, which was chosen through testing the switches.

The decoder (RotaryController) takes as inputs the two active-low outputs from the encoder (Rotary1_A and Rotary1_B, and Rotary2_A and Rotary2_B on the schematic), and the reset and clock signals, and outputs a single active-high clock pulse for each counter-clockwise turn of the rotary encoder, where output A leads B.

This is accomplished by first debouncing the input using a counter and comparator to eliminate glitches. Several DFFs are used in series in order to read inputs A and B from two consecutive clocks to determine when edges have occurred. An SRFF is set low when input B goes from low to high while input A is still low, and is set high otherwise. The inverse of this is returned as the output, resulting in a decoder for one direction. Two decoders are used for each rotary encoder to distinguish clockwise and counterclockwise turns.

2.1.5 VRAM and LCD

Two 256Kx4 DRAM with 512x4 SAM (serial access memory) chips (U19, U22, MT42C4225) were used for a total of 8 bits of data, with a 480x272 RGB TFT LCD (Connector J2, ER-TFT043-3).

VRAM: Each VRAM sends 4 bits of serial data to the LCD, and receives 4 bits of data from the CPU, along with a 9 bit address and row address strobe (RAS), column address strobe (CAS), write enable (WE),and output enable (OE) signals from the VRAM controller. A serial clock signal is also sent to the VRAM. The LCD controller generates the various clock signals that are sent to synchronize the VRAM and LCD.

The VRAM consists of the DRAM, transfer circuitry, and SAM. Read and write cycles are performed to read and write to the

DRAM, and read transfer cycles are performed to transfer a row from the DRAM to SAM. The VRAM is refreshed when no other operations are being performed to retain data.

The two chips are sent the same control and address signals, and receive and output different data signals.

memory,sam stuff

LCD: The LCD receives 8 bits of data in parallel from the DRAM, 3 bits of red, 3 bits of green, and 2 bits of blue, with the unused color bits pulled down. It also receives a horizontal sync, vertical sync, clock, and data enable signal from the LCD controller. The touch inputs were unused.

The LCD also required a 25V backlight supply (VLED on the schematic). A step-up converter (U21) was used, which is discussed in more detail in Section 2.1.2.

Logic overview: The VRAM-LCD system consists of a VRAM controller and LCD controller that interact with each other, the CPU, and the VRAM and LCD.

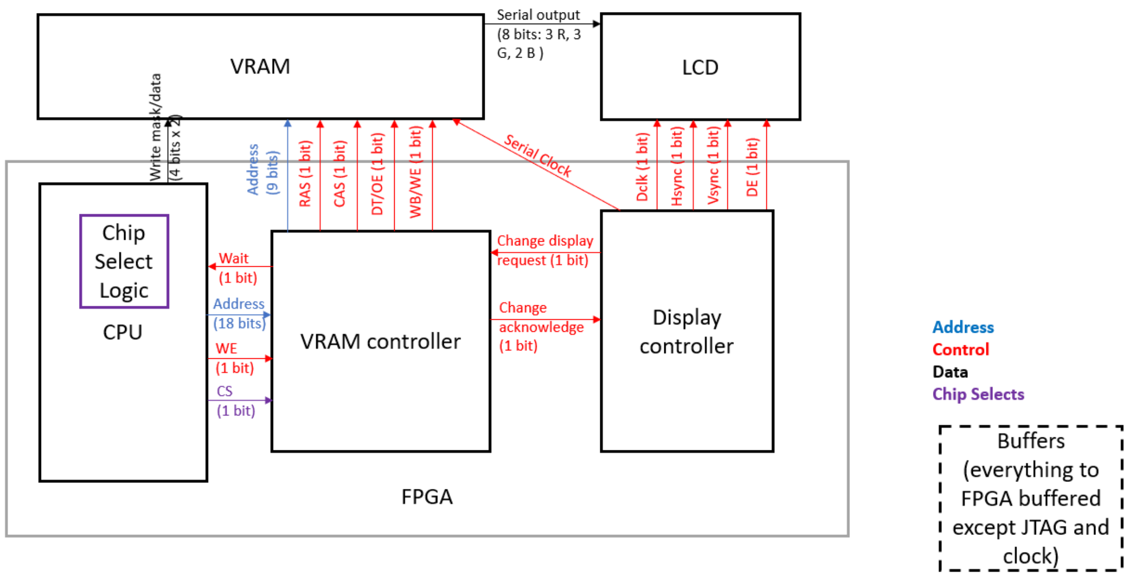

Figure 1: Block Diagram for VRAM and LCD.

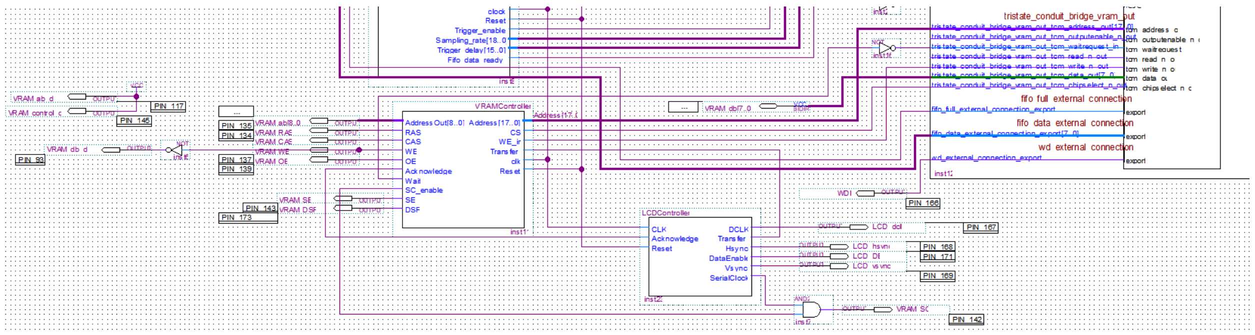

Figure 2: VRAM and LCD Controllers.

VRAM Controller: The VRAM controller takes several inputs from the CPU: 1-bit chip select and write enable signals, and an 18-bit address. It also takes as an input a transfer request signal from the display controller. A state machine is used to generate the output signals, the RAS, CAS, address selector, WE, OE, serial clock, CPU wait output, and transfer acknowledge signals.

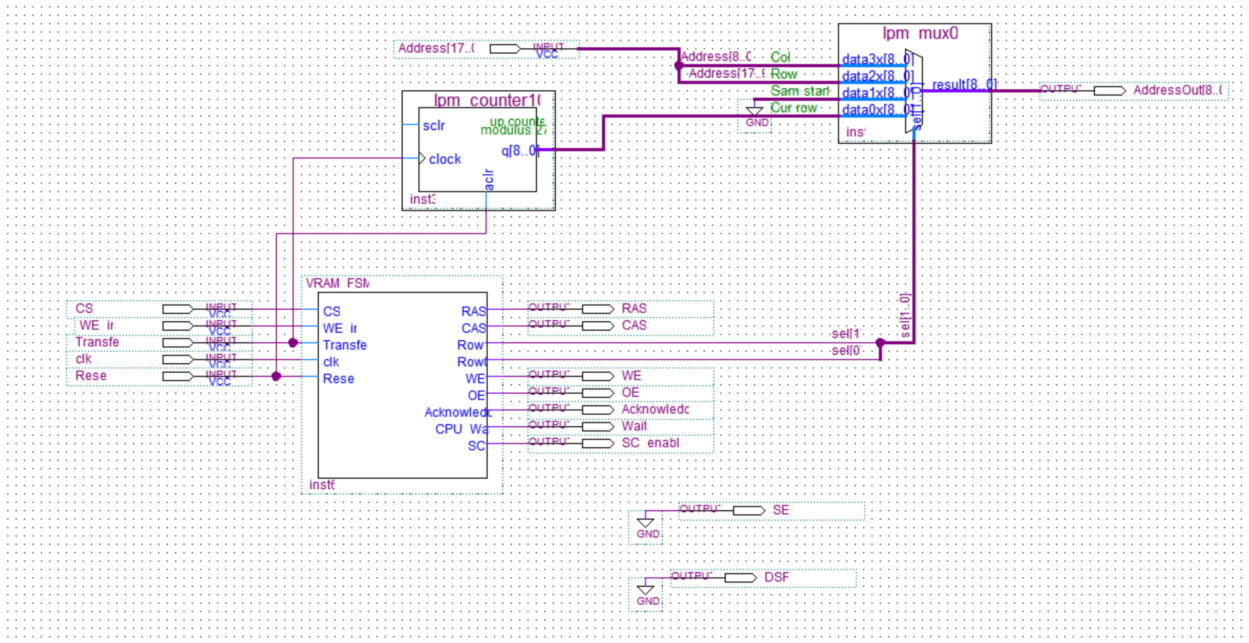

Figure 3: VRAM Controller.

The output address is selected and outputted using a multiplexer. For either the column or row, the first or second half of the input address, is returned. In the case of a row transfer the SAM start address, 0 for the beginning of the SAM, and the current row, are outputted at the appropriate times. A counter is used to increment the current row for after each row transfer cycle, from row 0 to 272, or the total width of the LCD.

The read, write, row transfer, and refresh cycles are performed with a Moore state machine. transfer/acknowledge with row transfer

The state machine begins at an idle state, and goes back to the idle state after each cycle is completed. If there is a row transfer request, when the transfer flag is active, the row transfer cycle is performed. If there is a write request, when the chip select and write enable inputs are active, then the write cycle is performed. If there is a read select, when the chip select is enabled but write enable is not, then the read cycle is performed. If none of these cycles are to be completed when in the idle state, a refresh cycle is performed.

Timing diagrams were used to create the state machine with the correct outputs, found in Appendix B.

LCD Controller: The LCD controller is used to generate the clock signals for the LCD.

The Vertical Sync (VSYNC) signal is used for changing rows, and Horizontal Sync (HSYNC) for changing columns. The Data Enable (DE) signal is also generated for when input data is valid within the VSYNC AND HSYNC signals. A 12 MHz clock signal is also sent to the LCD, while a serial clock signal is generated for the VRAM clock input to the serial address counter for the SAM registers.

The timing diagrams can be found in Appendix B, Figures 11 and 12.

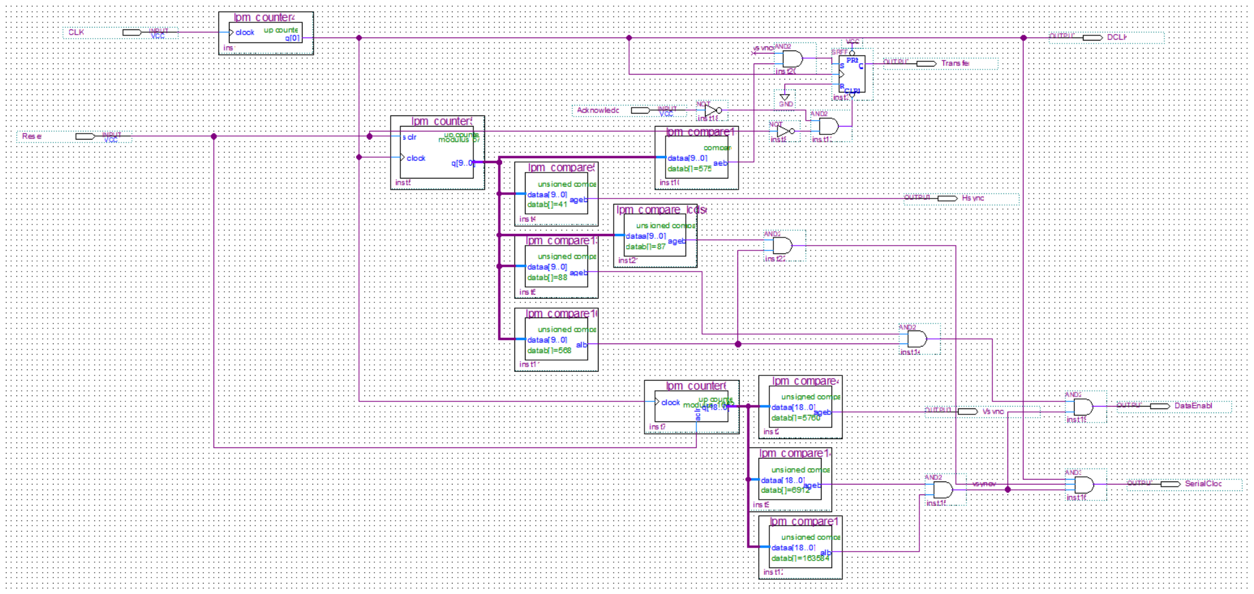
Figure 4: LCD Controller, with comparators for defining valid regions in the signals.

2.1.6 Memory

Memory Map:

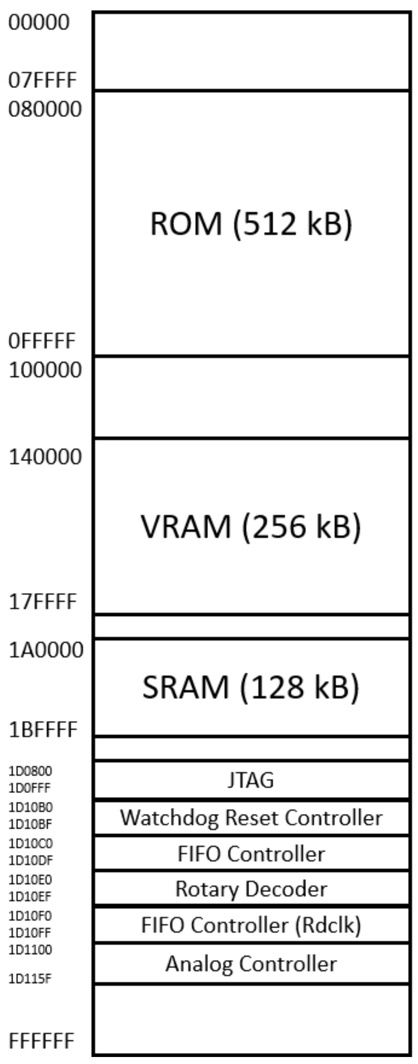

Figure 5: Memory map diagram, used by the CPU(?)

ROM: 1 512 K x 8 bit EEROM (U2, Am29F040) was used. 8 data bits are connected in parallel through the buffers to the CPU, along with 19 address bits, an active-low chip enable (CE), and output enable (OE) signal. The write enable (WE) signal was pulled up, as the ROM was written to with a dedicated ROM programmer.

The ROM timing diagram for the read cycle can be seen at Appendix B, Figure 13. 3 read wait states were used.

The executable file from the Nios II IDE was loaded into it using the ROM programmer.

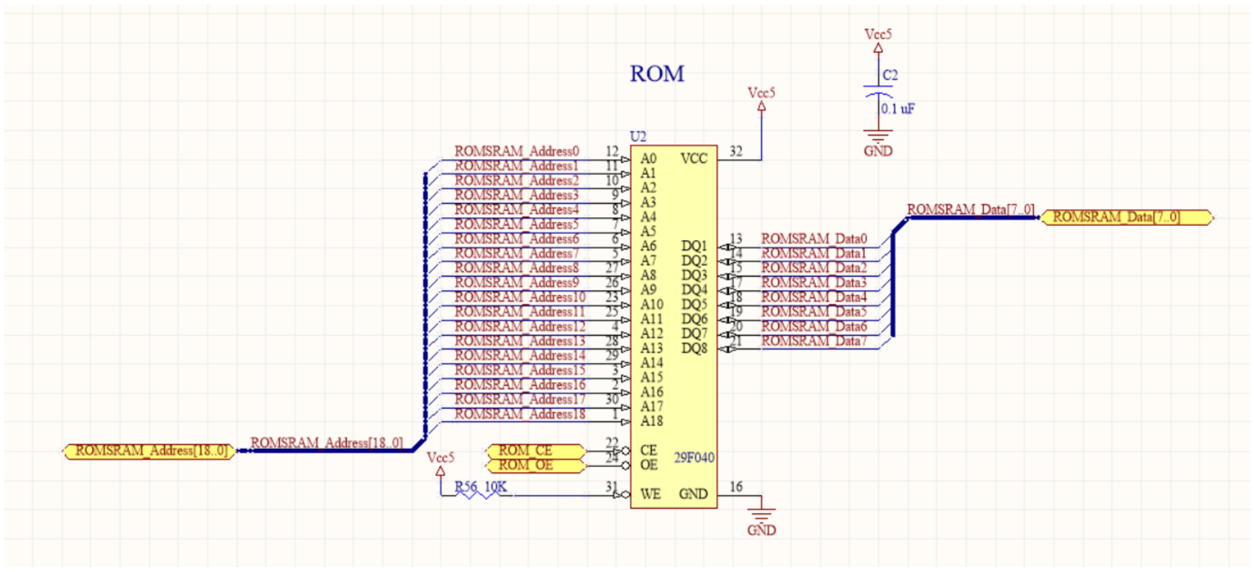

Figure 6: ROM Schematic.

SRAM: 1 128 K x 8 bit CMOS static RAM was used (U1, HM628128B)

The address line is shared between the ROM and SRAM, with only the 17 least significant bits used for RAM addressing. The 8-bit data bus is also shared between the ROM and SRAM. The active-low output enable (OE), write enable (WE), and chip select (CS1) are also connected between the SRAM and CPU, with the second active high chip select (CS2) pulled high as required by the read and write cycles.

The SRAM timing diagrams can be seen at Appendix B, Figures 14 and 15. 2 read wait states and 1 write wait state was used.

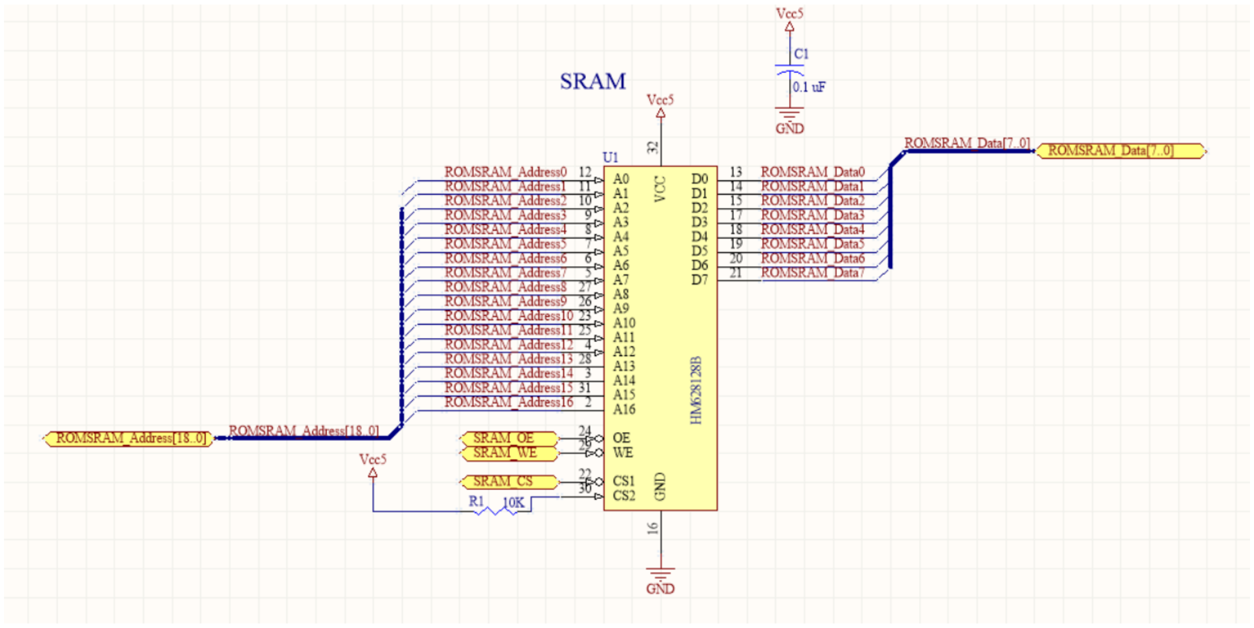

Figure 7: SRAM Schematic.

Serial EEPROM: A serial configuration device (U4, EPCS16), a 2 MB flash memory device, was used to store FPGA configuration data [3].

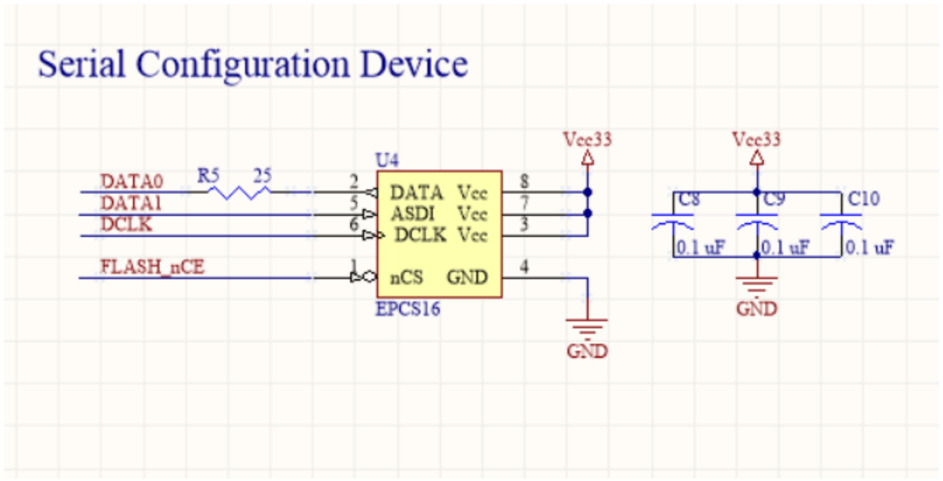

Figure 8: EEPROM Schematic.

[3] Was unable to successfully use serial device for unknown reasons, possibly due to incorrect FPGA configurations.

2.1.7 FPGA

An Altera Cyclone III FPGA was used (U9, EP3C25Q240). The three required voltage levels were generated with regulators, and each power pin was locally bypassed.

Buffers, or level shifters, were used to connect the buses and control signals from the peripheral chips to the FPGA to go between the low FPGA voltage and mainly 5V environment (U5, U6, U7, U8, U10, U12, 74LVT16245). A separate buffer was used to interface with the ADC (U17, TLC5510A), which had a larger input voltage range requirement (U18, SN74HCT240).

The pins were connected on the PCB as listed in Figure **??**.

The final product included several changes. Pin 162, an output for device configuration, was left floating, and the reset input was changed to Pin 187. One buffer was also left unusable, resulting in rewiring to extra I/O pins.

To Direction Location I/O Bank VREF Group Fitter Location ADC_clk Output PIN_240 8 B8_N0 PIN_240 ADC_db[7] Input PIN_230 8 B8_N0 PIN_230 ADC_db[6] Input PIN_233 8 B8_N0 PIN_233 ADC_db[5] Input PIN_234 8 B8_N0 PIN_234 ADC_db[4] Input PIN_235 8 B8_N0 PIN_235 ADC_db[3] Input PIN_236 8 B8_N0 PIN_236 ADC_db[2] Input PIN_237 8 B8_N0 PIN_237 ADC_db[1] Input PIN_238 8 B8_N0 PIN_238 ADC_db[0] Input PIN_239 8 B8_N0 PIN_239 altera_reserved_tck Input PIN_27 altera_reserved_tdi Input PIN_26 altera_reserved_tdo Output PIN_29 altera_reserved_tms Input PIN_28 Clock Input PIN_31 1 B1_N0 PIN_31 LCD_dclk Output PIN_167 6 B6_N0 PIN_167 LCD_DE Output PIN_171 6 B6_N0 PIN_171 LCD_hsync Output PIN_168 6 B6_N0 PIN_168 LCD_vsync Output PIN_169 6 B6_N0 PIN_169 PFO Input PIN_164 6 B6_N0 PIN_164 RAMROM_ab[18] Output PIN_39 2 B2_N0 PIN_39 RAMROM_ab[17] Output PIN_37 2 B2_N0 PIN_37 RAMROM_ab[16] Output PIN_38 2 B2_N0 PIN_38 RAMROM_ab[15] Output PIN_44 2 B2_N0 PIN_44 RAMROM_ab[14] Output PIN_41 2 B2_N0 PIN_41 RAMROM_ab[13] Output PIN_49 2 B2_N0 PIN_49 RAMROM_ab[12] Output PIN_43 2 B2_N0 PIN_43 RAMROM_ab[11] Output PIN_52 2 B2_N0 PIN_52 RAMROM_ab[10] Output PIN_55 2 B2_N0 PIN_55 RAMROM_ab[9] Output PIN_51 2 B2_N0 PIN_51 RAMROM_ab[8] Output PIN_50 2 B2_N0 PIN_50 RAMROM_ab[7] Output PIN_45 2 B2_N0 PIN_45 RAMROM_ab[6] Output PIN_46 2 B2_N0 PIN_46 RAMROM_ab[5] Output PIN_6 1 B1_N0 PIN_6 RAMROM_ab[4] Output PIN_9 1 B1_N0 PIN_9

RAMROM_ab[3] Output PIN_13 1 B1_N0 PIN_13 RAMROM_ab[2] Output PIN_18 1 B1_N0 PIN_18 RAMROM_ab[1] Output PIN_21 1 B1_N0 PIN_21 RAMROM_ab[0] Output PIN_22 1 B1_N0 PIN_22 RAMROM_db[7] Bidir PIN_69 3 B3_N0 PIN_69 RAMROM_db[6] Bidir PIN_70 3 B3_N0 PIN_70 RAMROM_db[5] Bidir PIN_71 3 B3_N0 PIN_71 RAMROM_db[4] Bidir PIN_72 3 B3_N0 PIN_72 RAMROM_db[3] Bidir PIN_73 3 B3_N0 PIN_73 RAMROM_db[2] Bidir PIN_68 3 B3_N0 PIN_68 RAMROM_db[1] Bidir PIN_65 3 B3_N0 PIN_65 RAMROM_db[0] Bidir PIN_64 3 B3_N0 PIN_64 Reset Input PIN_187 7 B7_N0 PIN_187 ROM_CS Output PIN_5 1 B1_N0 PIN_5 ROM_OE Output PIN_4 1 B1_N0 PIN_4 romram_db_dir Output PIN_76 3 B3_N0 PIN_76 rotary_1_A Input PIN_181 7 B7_N0 PIN_181 rotary_1_B Input PIN_182 7 B7_N0 PIN_182 rotary_1_P Input PIN_183 7 B7_N0 PIN_183 rotary_2_A Input PIN_184 7 B7_N0 PIN_184 rotary_2_B Input PIN_185 7 B7_N0 PIN_185 rotary_2_P Input PIN_186 7 B7_N0 PIN_186 SRAM_CS Output PIN_63 3 B3_N0 PIN_63 SRAM_OE Output PIN_57 2 B2_N0 PIN_57 SRAM_WE Output PIN_56 2 B2_N0 PIN_56 VRAM_ab[8] Output PIN_132 5 B5_N0 PIN_132 VRAM_ab[7] Output PIN_126 5 B5_N0 PIN_126 VRAM_ab[6] Output PIN_131 5 B5_N0 PIN_131 VRAM_ab[5] Output PIN_128 5 B5_N0 PIN_128 VRAM_ab[4] Output PIN_127 5 B5_N0 PIN_127 VRAM_ab[3] Output PIN_120 4 B4_N0 PIN_120 VRAM_ab[2] Output PIN_119 4 B4_N0 PIN_119 VRAM_ab[1] Output PIN_118 4 B4_N0 PIN_118 VRAM_ab[0] Output PIN_133 5 B5_N0 PIN_133 VRAM_ab_dir Output PIN_117 4 B4_N0 PIN_117 VRAM_CAS Output PIN_134 5 B5_N0 PIN_134 VRAM_control_dir Output PIN_145 5 B5_N0 PIN_145 VRAM_db[7] Bidir PIN_80 3 B3_N0 PIN_80 VRAM_db[6] Bidir PIN_78 3 B3_N0 PIN_78 VRAM_db[5] Bidir PIN_82 3 B3_N0 PIN_82 VRAM_db[4] Bidir PIN_81 3 B3_N0 PIN_81 VRAM_db[3] Bidir PIN_84 3 B3_N0 PIN_84 VRAM_db[2] Bidir PIN_83 3 B3_N0 PIN_83 VRAM_db[1] Bidir PIN_88 3 B3_N0 PIN_88 VRAM_db[0] Bidir PIN_87 3 B3_N0 PIN_87 VRAM_db_dir Output PIN_93 4 B4_N0 PIN_93 VRAM_DSF Output PIN_173 6 B6_N0 PIN_173 VRAM_OE Output PIN_139 5 B5_N0 PIN_139 VRAM_RAS Output PIN_135 5 B5_N0 PIN_135 VRAM_SC Output PIN_142 5 B5_N0 PIN_142 VRAM_SE Output PIN_143 5 B5_N0 PIN_143 VRAM_WE Output PIN_137 5 B5_N0 PIN_137 WDI Output PIN_166 6 B6_N0 PIN_166 WDO Input PIN_161 6 B6_N0 PIN_161

2.1.8 JTAG, Reset, and Clock

pin 162 used during FPGA startup, can't use for reset

JTAG:

Reset: A reset circuit is used, which includes a microprocessor supervisory device (U3, MAX705). The watchdog input (WDI) is implemented in the main loop, and a voltage divider is used for a power-fail warning at 4 V from the 5 V supply.

Clock:

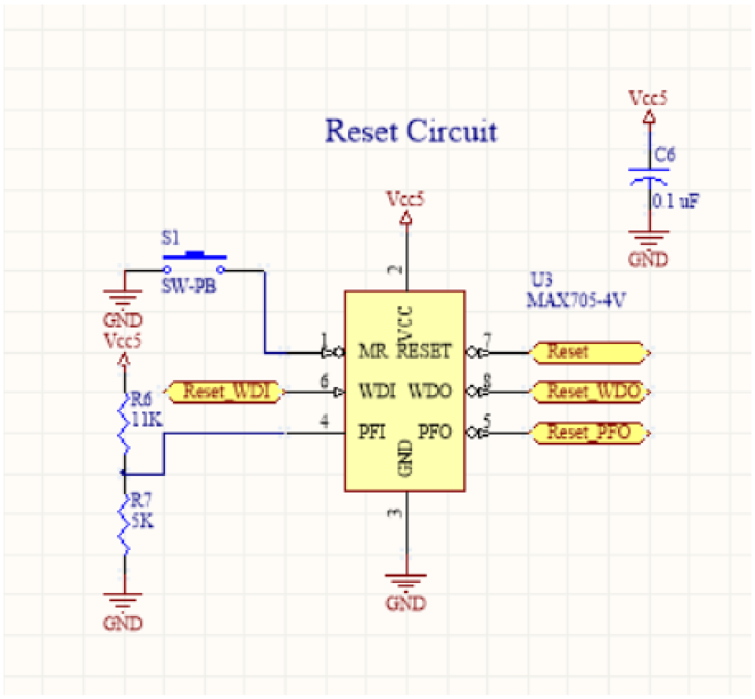

Figure 9: Reset Circuit Schematic.

The active-low watchdog output, power-fail output, and reset output, and manual reset output are combined for an overall reset signal that is sent to the CPU and various controllers.

2.1.9 Fixes

wrong adc footprint, buffer soldering error, backwards LCD connector

2.2 Software

2.2.1 Software System Overview

2.2.2

A Code

Listing 1: data/scoptrig.vhd

```
----------------------------------------------------------------------------
--
--  Oscilloscope Digital Trigger
--
--  This is an implementation of a trigger for a digital oscilloscope in
--  VHDL.  There are three inputs to the system, one selects the trigger
--  slope and the other two determine the relationship between the trigger
--  level and the signal level.  The only output is a trigger signal which
--  indicates a trigger event has occurred.
--
--  The file contains multiple architectures for a Moore state machine
--  implementation to demonstrate the different ways of building a state
--  machine.
--
--
--  Revision History:
--     13 Apr 04  Glen George       Initial revision.
--      4 Nov 05  Glen George       Updated comments.
--     17 Nov 07  Glen George       Updated comments.
--     13 Feb 10  Glen George       Added more example architectures.
--
----------------------------------------------------------------------------


-- bring in the necessary packages
library  ieee;
use  ieee.std_logic_1164.all;


--
--  Oscilloscope Digital Trigger entity declaration
--

entity  ScopeTrigger  is
    port (
        TS         :  in  std_logic;      -- trigger slope (1 -> negative, 0 -> positive)
        TEQ        :  in  std_logic;      -- signal and trigger levels equal
        TLT        :  in  std_logic;      -- signal level < trigger level
        clk        :  in  std_logic;      -- clock
        Reset      :  in  std_logic;      -- reset the system
        TrigEvent  :  out  std_logic      -- a trigger event has occurred
    );
end  ScopeTrigger;



--
--  Oscilloscope Digital Trigger Moore State Machine
--     State Assignment Architecture
--
--  This architecture just shows the basic state machine syntax when the state
--  assignments are made manually.  This is useful for minimizing output
--  decoding logic and avoiding glitches in the output (due to the decoding
--  logic).
--

architecture  assign_statebits  of  ScopeTrigger  is

    subtype  states  is  std_logic_vector(2 downto 0);     -- state type

    -- define the actual states as constants
    constant  IDLE      : states := "000";  -- waiting for start of trigger event
    constant  WAIT_POS  : states := "001";  -- waiting for positive slope trigger
    constant  WAIT_NEG  : states := "010";  -- waiting for negative slope trigger
    constant  TRIGGER   : states := "100";  -- got a trigger event


    signal  CurrentState  :  states;    -- current state
    signal  NextState     :  states;    -- next state

begin


    -- the output is always the high bit of the state encoding
    TrigEvent <= CurrentState(2);


    -- compute the next state (function of current state and inputs)

    transition:  process (Reset, TS, TEQ, TLT, CurrentState)
    begin

        case  CurrentState  is          -- do the state transition/output

            when  IDLE =>               -- in idle state, do transition
                if  (TS = '0' and TLT = '1' and TEQ = '0')  then
                    NextState <= WAIT_POS;      -- below trigger and + slope
                elsif  (TS = '1' and TLT = '0' and TEQ = '0')  then
                    NextState <= WAIT_NEG;      -- above trigger and - slope
                else
                    NextState <= IDLE;          -- trigger not possible yet
                end if;

            when  WAIT_POS =>           -- waiting for positive slope trigger
                if  (TS = '0' and TLT = '1')  then
                    NextState <= WAIT_POS;      -- no trigger yet
                elsif  (TS = '0' and TLT = '0')  then
                    NextState <= TRIGGER;       -- got a trigger
                else
                    NextState <= IDLE;          -- trigger slope changed
                end if;

            when  WAIT_NEG =>           -- waiting for negative slope trigger
                if  (TS = '1' and TLT = '0' and TEQ = '0')  then
                    NextState <= WAIT_NEG;      -- no trigger yet
                elsif  (TS = '1' and (TLT = '1' or TEQ = '1'))  then
                    NextState <= TRIGGER;       -- got a trigger
                else
                    NextState <= IDLE;          -- trigger slope changed
                end if;

            when  TRIGGER =>            -- in the trigger state
                NextState <= IDLE;      -- always go back to idle

                        when others =>
                                NextState <= IDLE;

        end case;

        if  Reset = '1'  then           -- reset overrides everything
            NextState <= IDLE;          --   go to idle on reset
        end if;

    end process transition;


    -- storage of current state (loads the next state on the clock)
    process (clk)
    begin

        if  clk = '1'  then              -- only change on rising edge of clock
            CurrentState <= NextState;  -- save the new state information
        end if;

    end process;


end  assign_statebits;
```

B Timing Diagrams

B.1 ADC

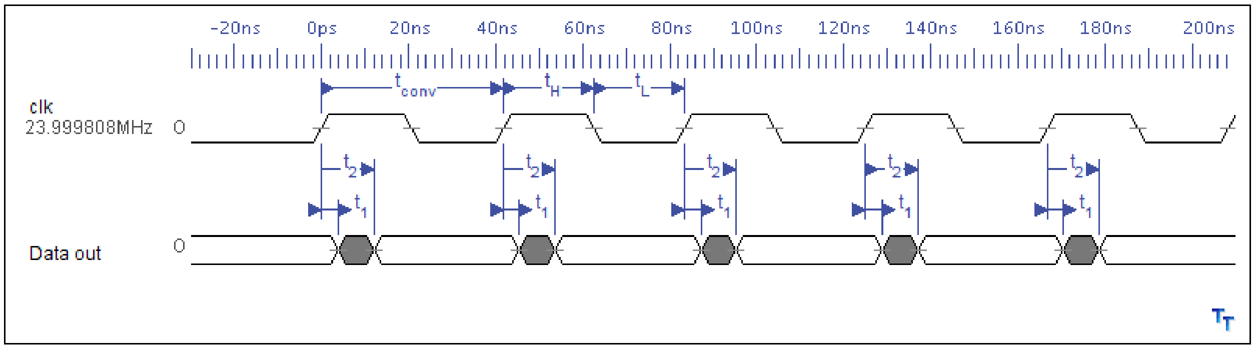

Figure 10: ADC Timing Diagram

B.2 LCD

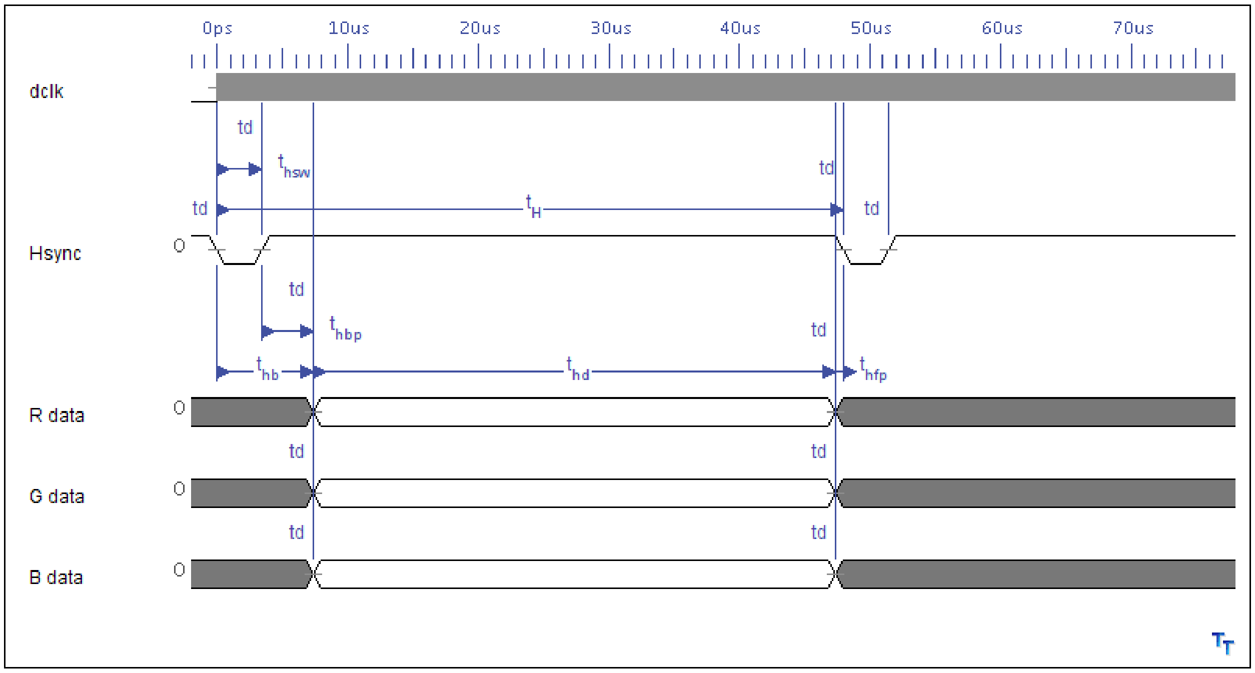

Figure 11: LCD Horizontal Timing Diagram

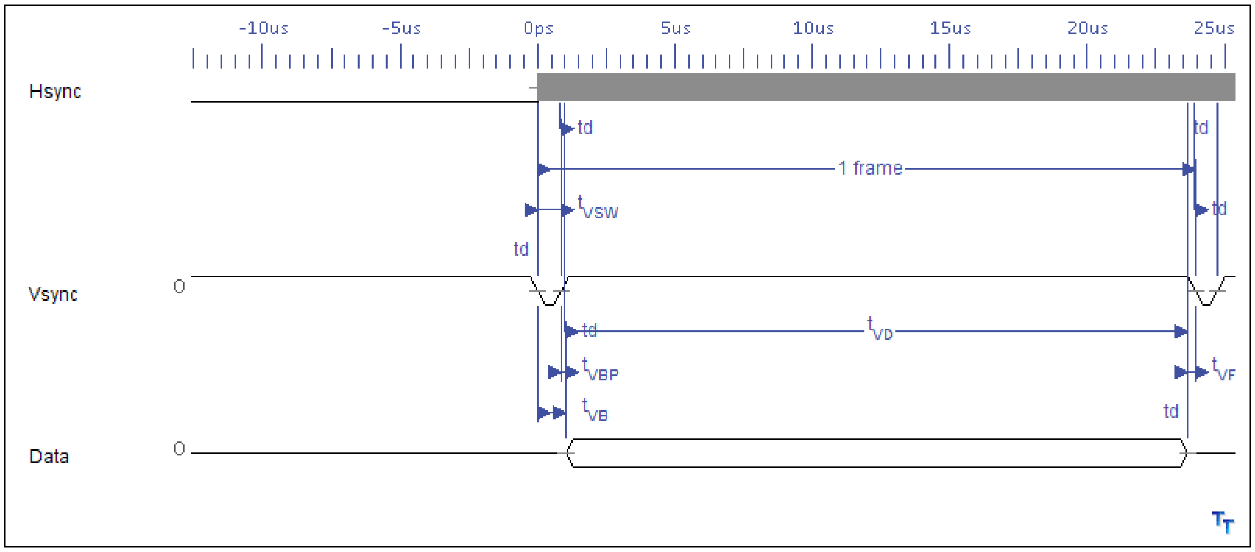

Figure 12: LCD Vertical Timing Diagram

B.3 ROM and RAM

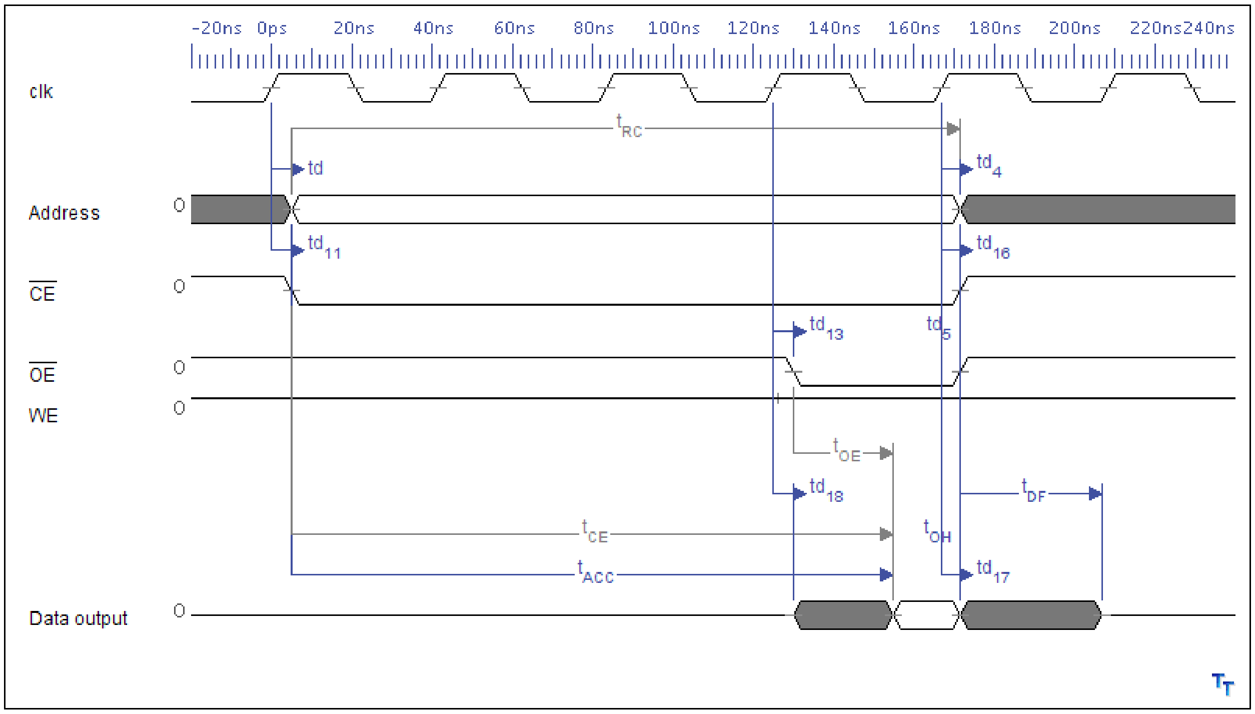

Figure 13: ROM Read Cycle Diagram

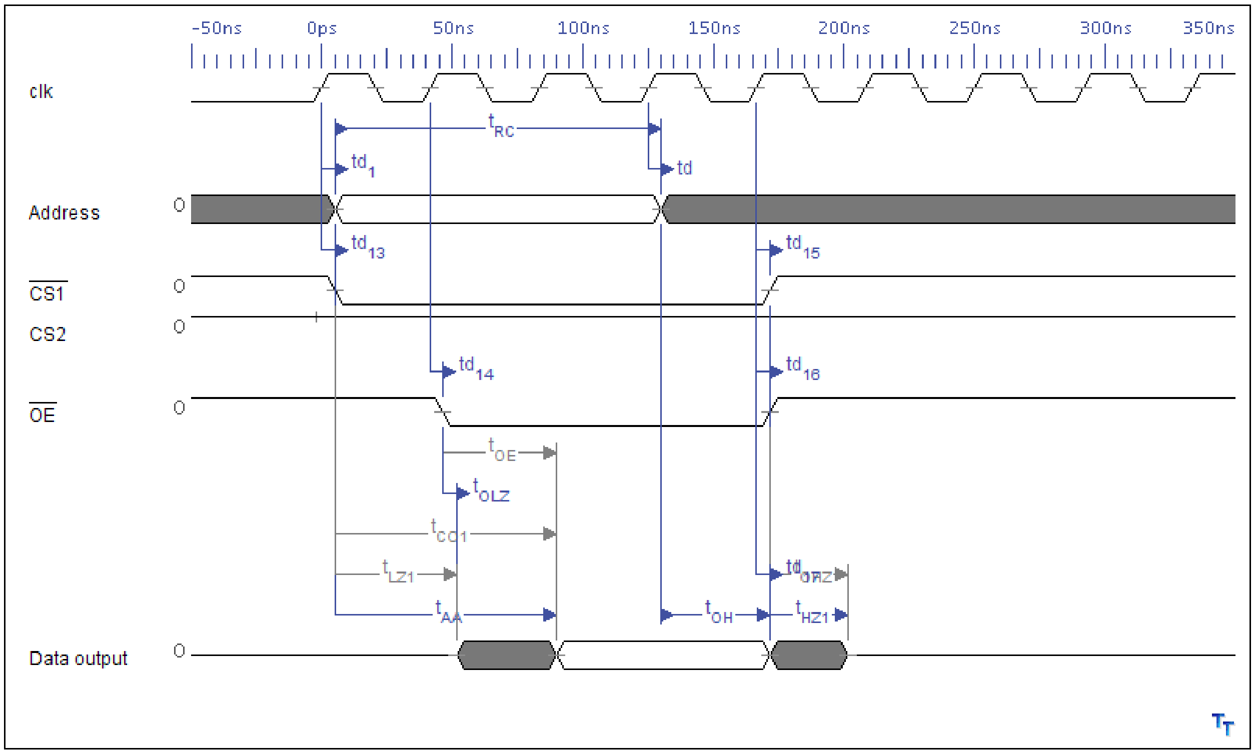

Figure 14: RAM Read Cycle Diagram

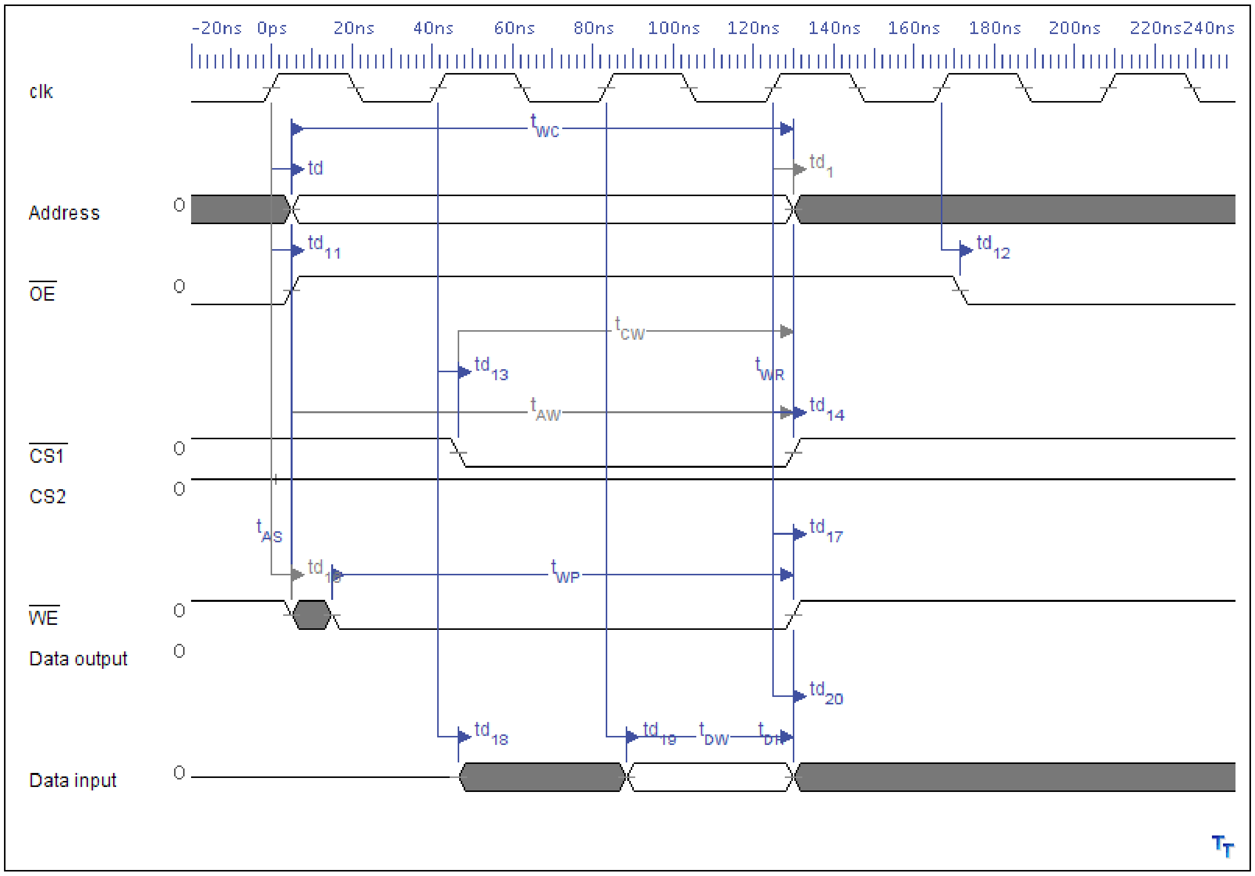

Figure 15: RAM Write Cycle Diagram

B.4 VRAM

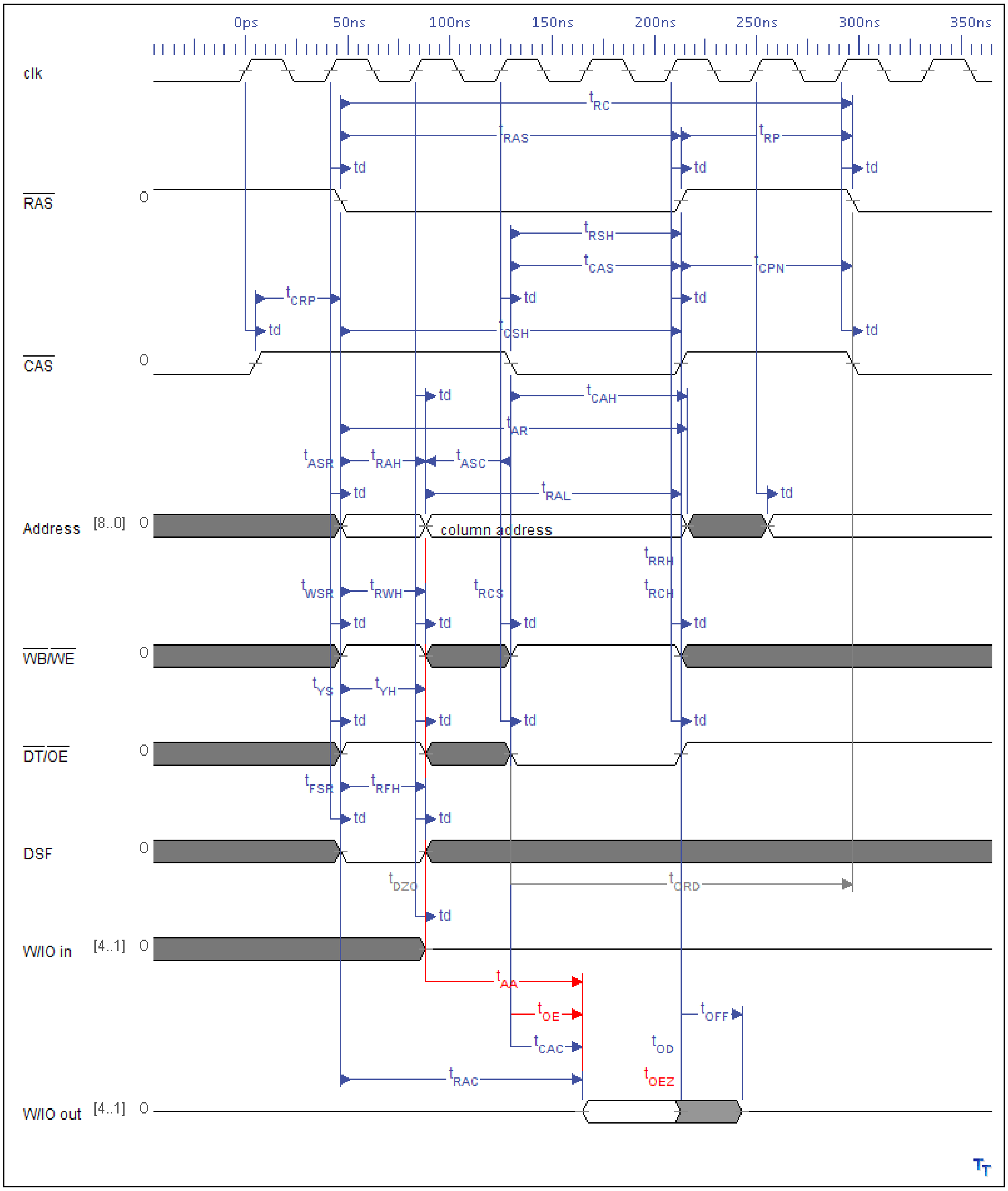

Figure 16: VRAM Read Cycle Diagram

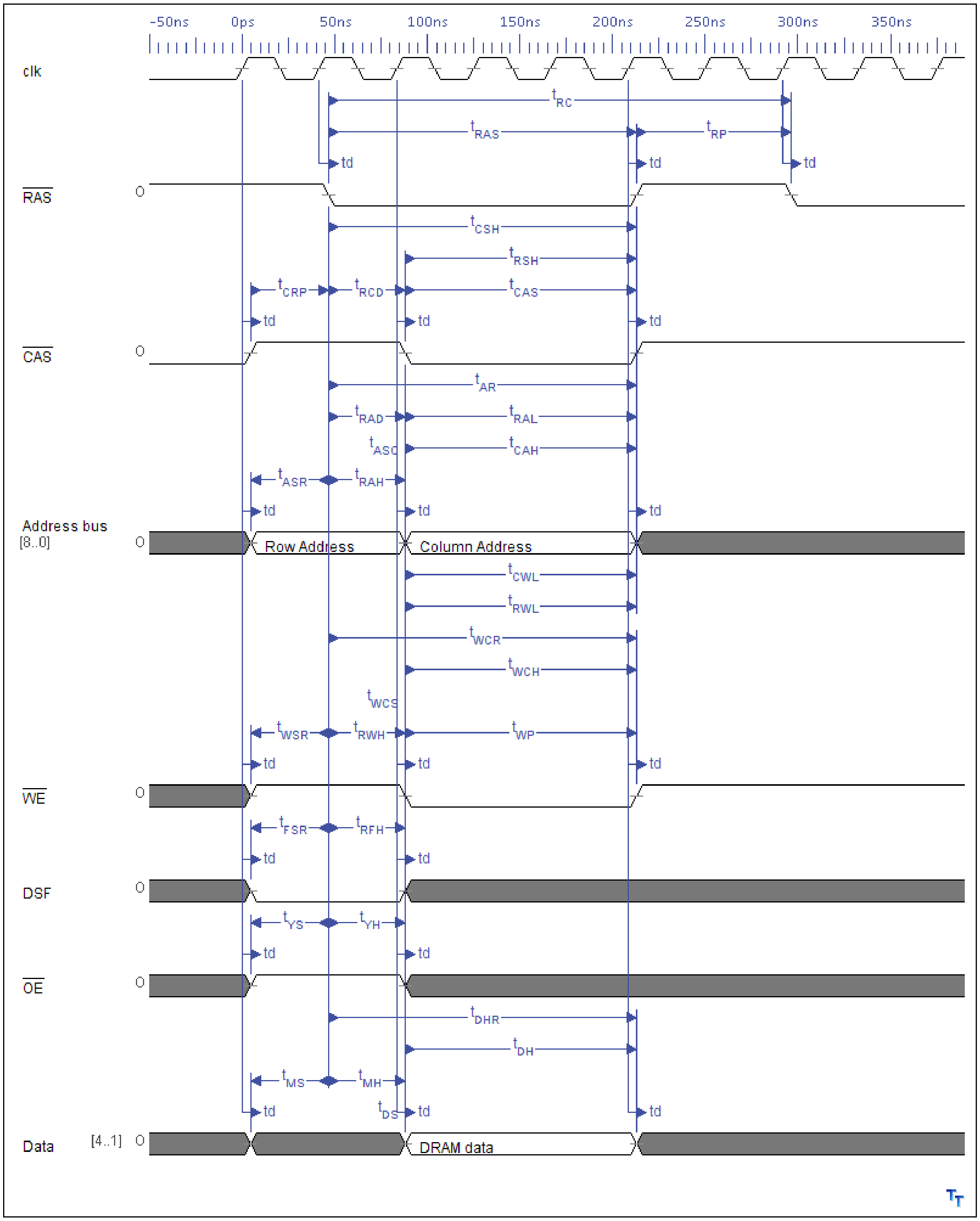

Figure 17: VRAM Write Cycle Diagram

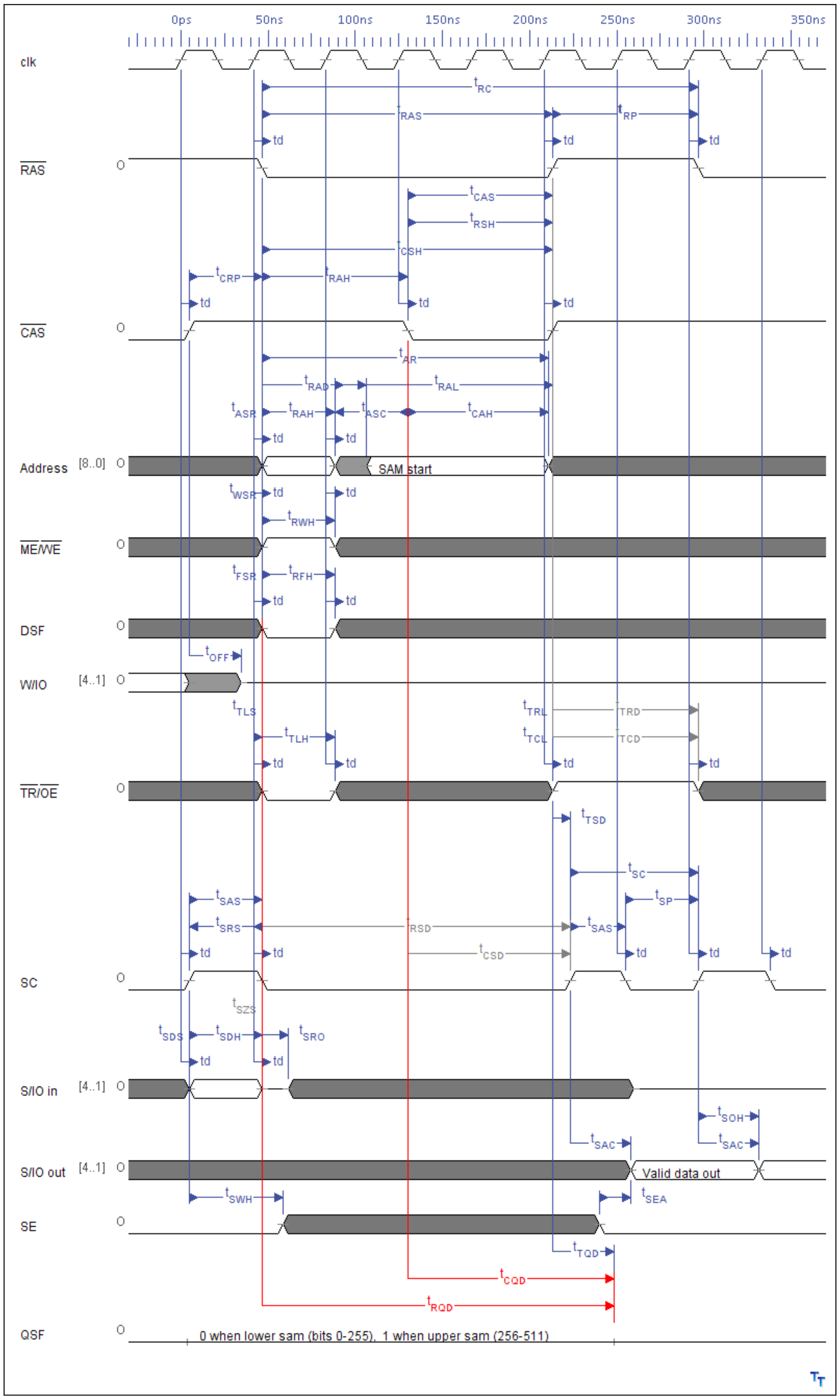

Figure 18: VRAM Row Transfer Cycle Diagram

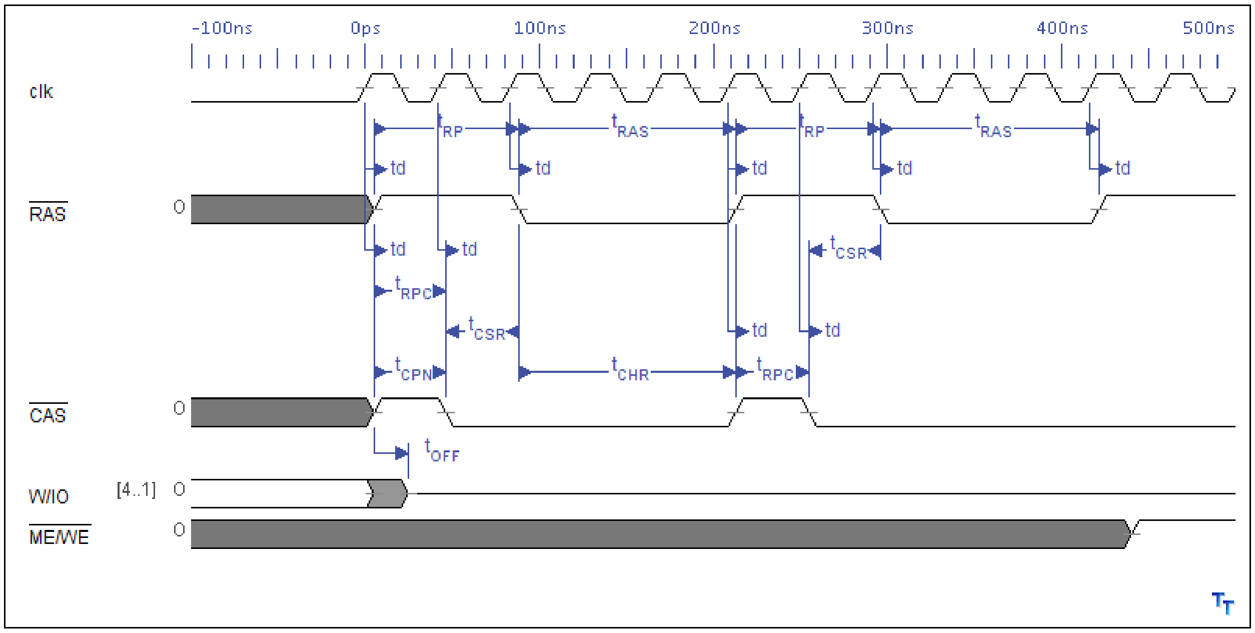

Figure 19: VRAM Refresh Cycle Diagram (CAS before RAS)

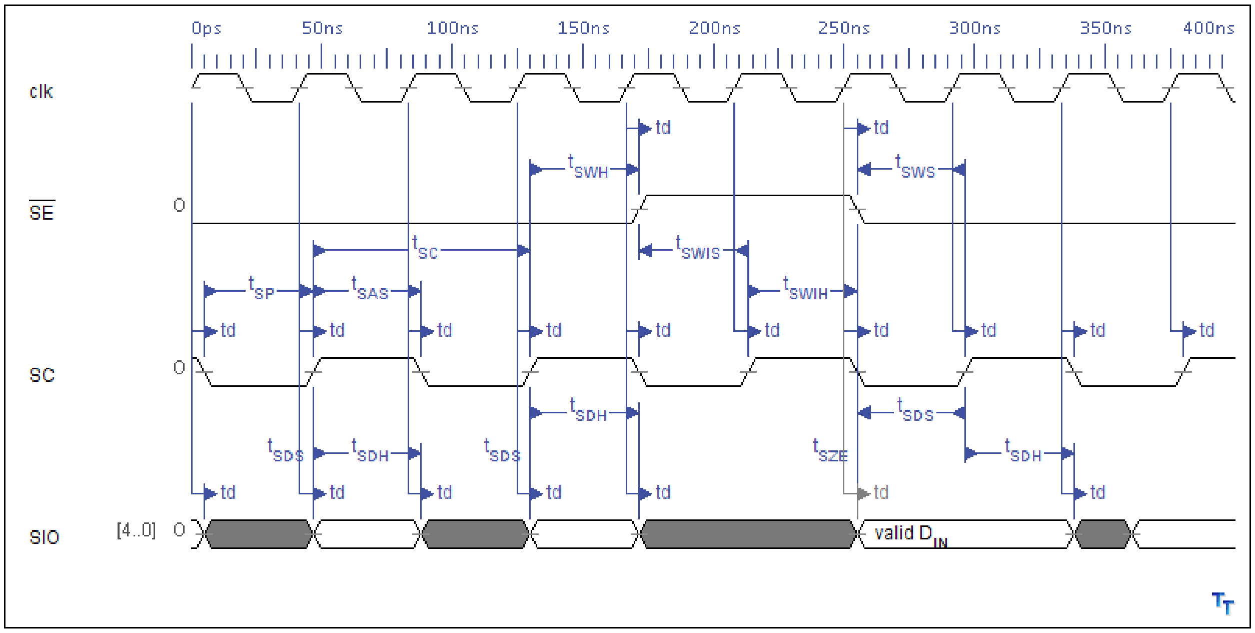

Figure 20: VRAM Serial Write Cycle Diagram

Index