GUSY-V6

Jesús Arias Alvarez

1 The GUSY-v6 computer

The Gusy computer is an FPGA implementation of the GUS16 CPU along with several peripherals and memories. In this version the CPU is the V6 revision with its new instruction set and the main idea was to reduce the internal memory to a minimum boot ROM that allows the uploading of code to the external RAM and no more. In fact the size of the boot ROM is only 32 16-bit words, enough for a simple program that reads program images into RAM via a serial port and jumps to the uploaded code.

A descriptive diagram of the Gusy computer is shown in figure 1. Here we should remark:

- A GUS16-v6 CPU with an stoppable clock that allows the stealing of cycles, mainly from the video controller.
- A Video controller with a graphic mode of 640 x 480 pixels and 2 bits per pixel. The framebuffer memory is also located in the external RAM, so, the CPU has to be stopped when the video controller is reading the memory. This happens 1 cycle every 8 during the visible part of video lines. On average, the video controller steals 38400 cycles every frame of 800×525 pixels, resulting in a 9.14% memory bandwidth (or an effective CPU clock of 22.7MHz instead of 25MHz)
- A simple UART with a fixed baud rate (default 115200bps). This peripheral is also a bit peculiar because it can also stop the CPU clock. This was done in order to simplify the bootloader code: If the receiver register is read when there is no data available the program execution stalls until a new character arrives at the receiver. In this way there is no need to keep polling an status register until a data available flag is set and the code is more compact. The same happens with the transmitter: if the transmitter register gets written while an old data is still being transmitted the CPU stalls until the new character can be written. Of course, the related flags can also be read in an status register and the usual polling routines do not result in stalls (this can be desirable if there are interrupts enabled).
- A 16-bit timer counter is also included, mainly for profiling and interrupt generation. It includes a fixed prescaler divider that results in a 1 MHz counter frequency (it counts microseconds), or in a close value if there isn't an integer divider.
- Five interrupt sources with independent vectors. These sources can be enabled by writing the Interrupt Enable register with ones (all its bits are zeroes after reset). The interrupts have a fixed priority order.

1.1 CPU clock

In this computer there is a memory contention between the video controller and the CPU, with the video controller having the highest priority. Thus, when a video read is done the CPU has to wait until the next clock cycle. This is achieved by forcing the CPU clock high when the video read signal, vrd, is active (OR logic).

The same happens when the UART data register is read or written and the corresponding flag is inactive. The clock logic and its timing is shown in figure 2.

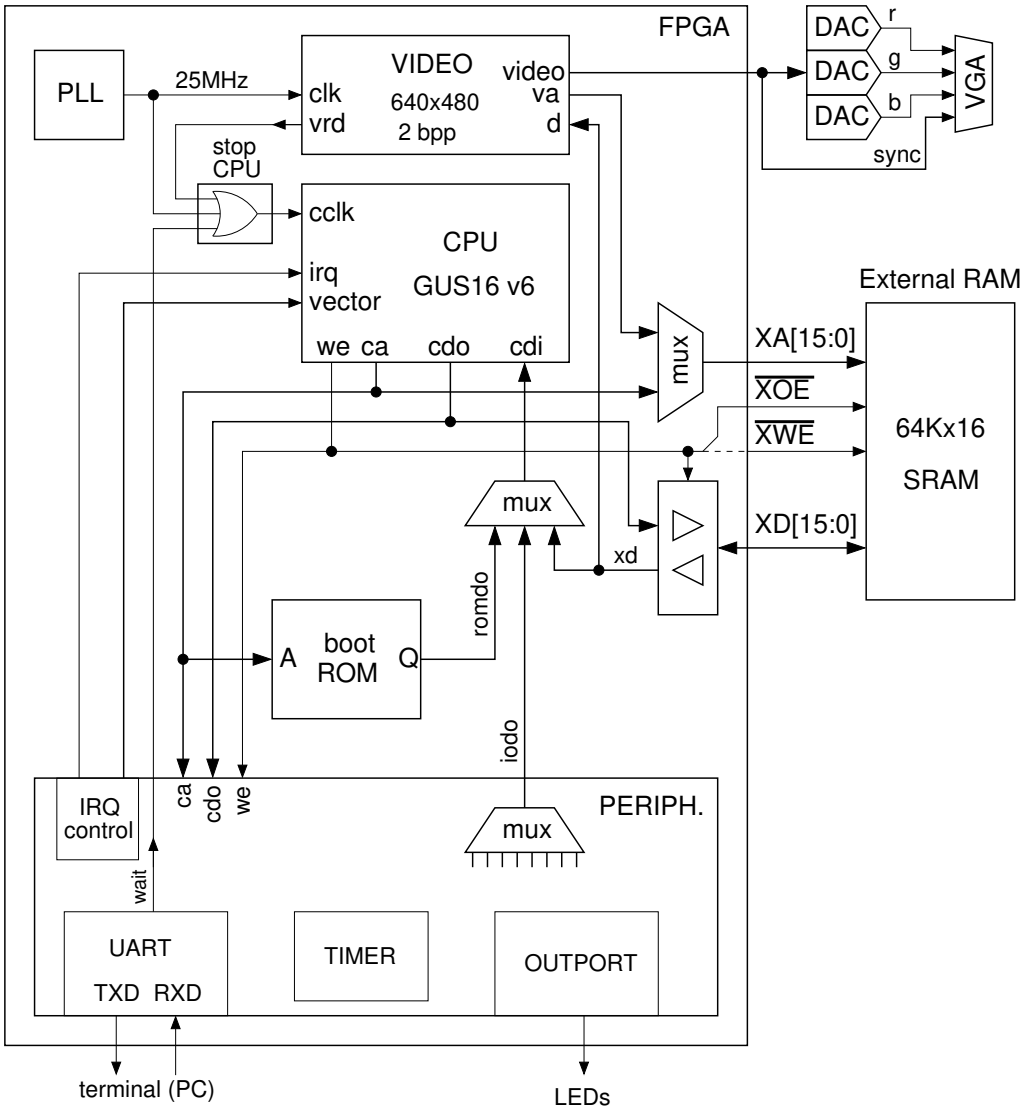


Figure 1: Block diagram of the GUSY computer.

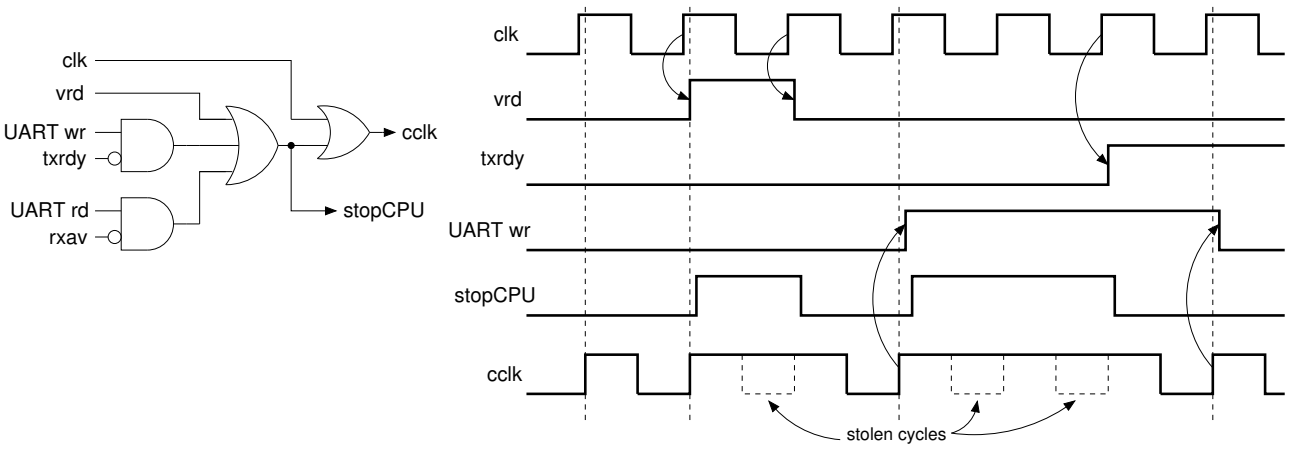


Figure 2: CPU clock logic and timing example. Clocks are active on the rising edges.

```
0000  -                ORG 0           ; RESET
0000  -  5722  INIT:   LDI     R7,0x22  ; UARTDAT
0001  -  60E0  BUC1:   LD      R0,(R7)  ; Blocking read
0002  -  484C          CMPI    R0,'L'
0003  -  9FFD          JNZ     BUC1
0004  -  68E0          ST      (R7),R0
0005  -  700B          JAL     GETW
0006  -  0A01          OR      R2,R0,R0 ; ADDRESS
0007  -  7009          JAL     GETW
0008  -  0B01          OR      R3,R0,R0 ; COUNT
0009  -  7007          JAL     GETW
000A  -  0C01          OR      R4,R0,R0 ; ENTRY POINT
000B  -
000B  -  7005  BUC2:   JAL     GETW
000C  -  6840          ST      (R2),R0
000D  -  1201          ADDI    R2,1
000E  -  1B01          SUBI    R3,1
000F  -  9FFB          JNZ     BUC2
0010  -  58F2          JIND    R4
0011  -
0011  -  60E0  GETW:   LD      R0,(R7) ; LSB, Blocking read
0012  -  61E0          LD      R1,(R7) ; MSB, Blocking read
0013  -  5944          RORI    R1,R1,8
0014  -  0805          OR      R0,R0,R1
0015  -  58FA          JIND    R6
```

Figure 3: Listing of the bootloader code

Most peripherals use the same clock as the CPU (cclk), but the UART and the timer are running on the main clock because they require an accurate frequency. In these peripherals the writing and read strobes have to be validated with the inverted 'stopCPU' signal in order to avoid multiple read or writes if the CPU clock is stopped during the peripheral access.

1.2 Boot ROM

The boot ROM is just a simple combinational block with 5 address inputs and 16 data outputs. The synthesis tool translates it into a digital circuit that depends on the ROM contents, and for the actual code it requires 40 logic cells. The bootloader program includes 22 instructions and is listed in figure 3. It first waits until an 'L' is received, then it reads 6 bytes with the program image information (destination address, size, and start address), reads the whole program, and jumps to its starting address.

The program images that are loaded through the serial port must be preceded by an 8-byte header, LSB bytes first:

LSB	MSB	Comments
0xFF	0x4C ('L')	0xFF is a byte-sync character for UART RX
Loading Address		
Size (words)		not including this header
Starting Address		0x0000 means enter bootloader again

The first 0xFF byte isn't strictly needed, but it is included in order to have an even number of bytes in the header, and to ease the correct reception of the following data as none of its data bits could be misunderstood as an start bit. Also, the 'Start Address' word can be made zero and the bootloader jumps to itself after loading a data block. In this way more than a single data block can be loaded before jumping to the program code.

1.3 External RAM

The external memory is an static RAM with 16-bit data and address buses. Reads and writes are always 16-bit wide, so, the byte select signals, XBHE, and XBLE, are always active (low). The chip select signal is also

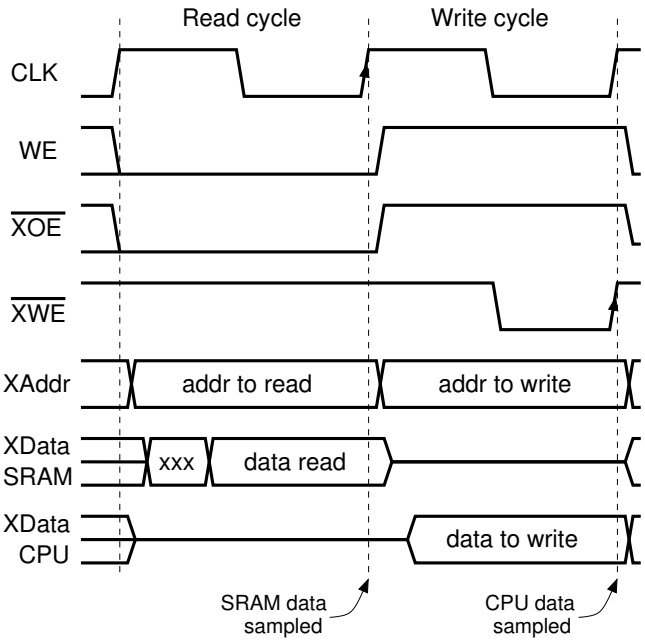


Figure 4: Chronograph of the external RAM read and write cycles.

always active, leaving only two control signals, output-enable, and write-enable, that have to be generated by the computer. The timing of the external RAM interface is presented in figure 4, where the internal write-enable signal, WE, is used directly as the active-low output enable, XOE, and selects the direction of the data bus. Also, on write cycles, a low pulse is generated on the SRAM write-enable input at the end of the cycle, when both the address and data buses have stable values.

Also, notice that when the video controller is performing a memory read, the CPU clock, CCLK, is forced high and WE is forced LOW. This results in a read cycle for the memory and one cycle delay for the CPU.

1.4 Memory map

The GUS16 CPU can address up to 64Kword. In this address space the boot ROM, the external RAM, and the I/O registers have to be mapped. In figure 5 the relevant areas of the memory map are shown. The first 32 words are mapped to the boot ROM, followed by another 32 words reserved for I/O registers. The external RAM occupies all the memory space and gets read and written along with the ROM and the peripherals, but its data is ignored by the CPU for these low addresses. 38400 words are reserved at the top of the map for video memory, with each word containing 8 pixels.

In figure 5 the registers of the included peripherals are also shown. These peripherals are:

1.4.1 UART

The UART is a simple one with a fixed speed of 115200 bps and a fixed data format of 8 bits, 1 stop bit and no parity. There are two data registers for receiver and transmitter and several flags. The data registers have 8 bits and therefore the bits 8 to 15 are read as zero and ignored on writes. The receiver include a holding register (1-byte FIFO) in addition to its corresponding shifting register. The flags can be read in the PFLAGS register, where all the peripheral flags are located, and are:

- RXAV: Receiver data available. Set to one when a character is received and cleared by reading the UART data register. This flag can request an interrupt when set.
- TXRDY: Transmitter ready to accept data. Set to one when the transmitter is idle. This flag can request an interrupt when set.

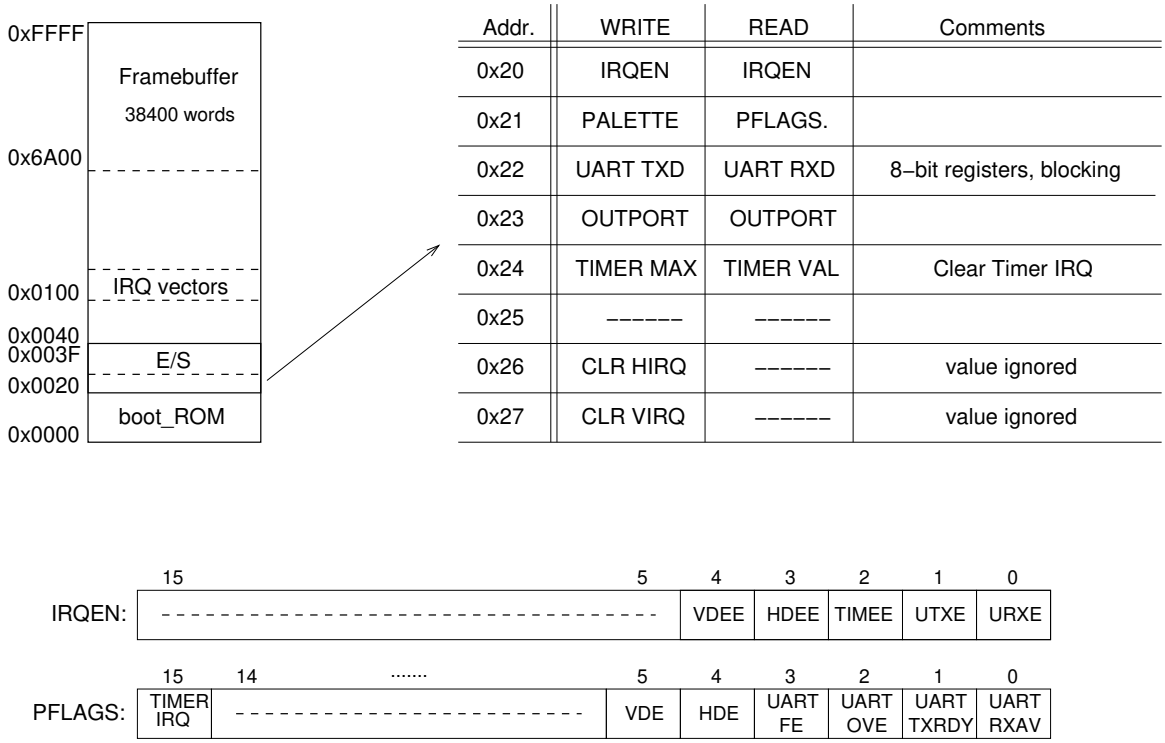


Figure 5: Memory map, including the I/O block, and bit fields of I/O registers

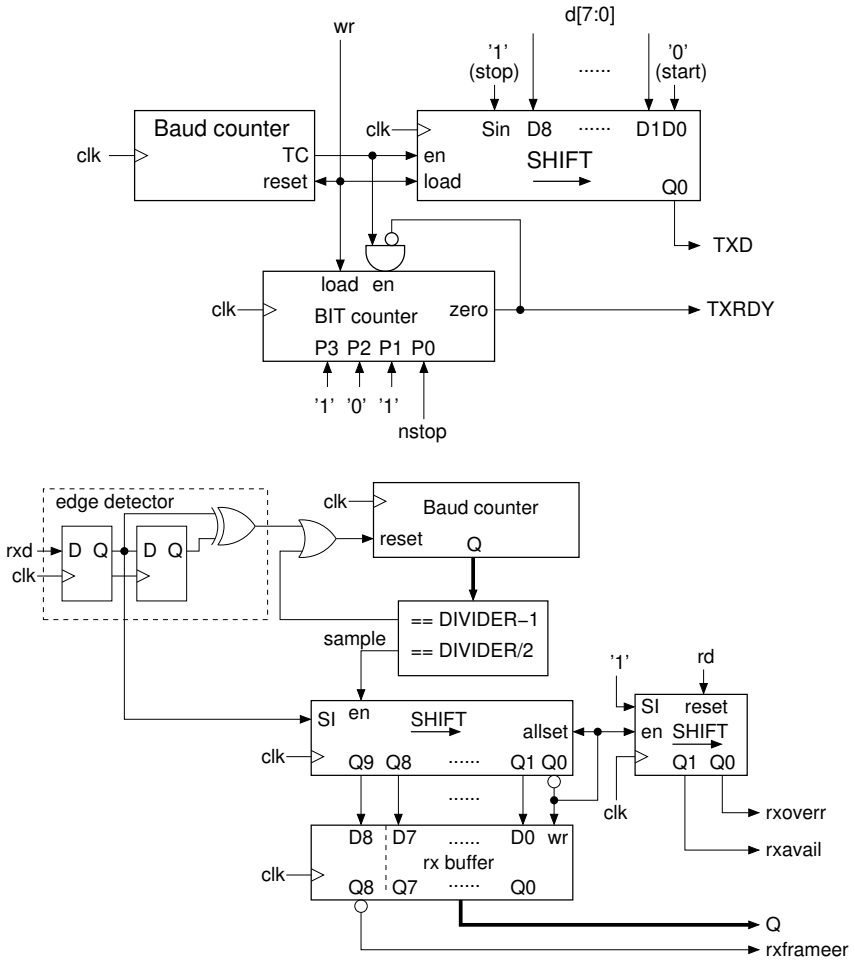


Figure 6: Block diagrams for the UART transmitter and receiver subsystems

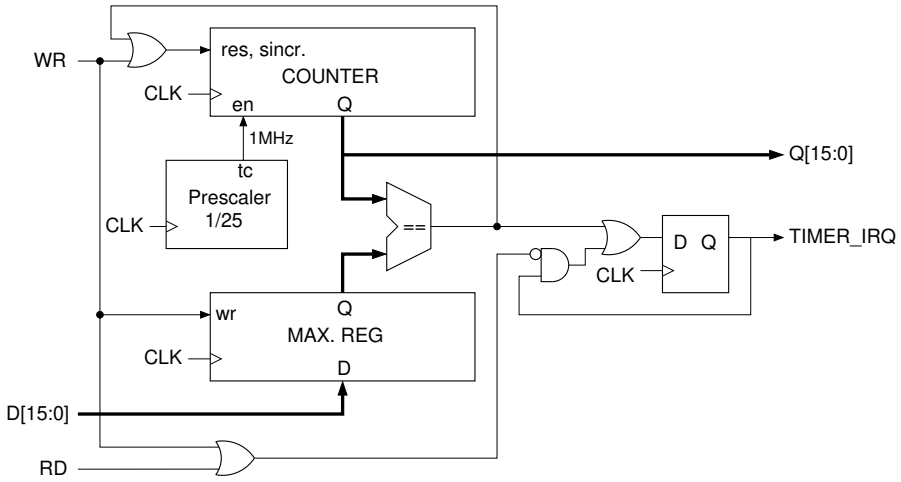


Figure 7: Block diagram of the Timer peripheral

- OVE: Overrun. Set to one if a character is received while RXAV was active.
- FE: Frame error. Set to one when a character is received with its stop bit as zero.

The block diagrams for the transmitter and receiver are presented in figure 6. The transmitter includes a clock divider, “baud counter”, that gets reset on writes, a 9-bit shift register, and a bit counter. ’nstop’ is hardwired to ’0’, so, only one stop bit is transmitted. The receiver is a bit more complex. It also includes a clock divider that get reset every time an edge is present in the input data waveform. A ’sample’ pulse is also generated when the clock divider reaches its middle value, meaning the input bits are sampled at the center of their bit time. These bits are shifted into a 10-bit shift register, and when the start bit reaches the LSB position the remaining bits are transferred to the RX buffer, the RX available and RX overrun flags are updated, and the whole shift register is preset synchronously to ’all-ones’.

1.4.2 Timer

A simple 16-bit timer allows the generation of periodic interrupts. It includes two registers, one of them is the actual counter while the other holds the maximum count (see figure 7). When the counter reaches the value of maximum count on the next cycle it gets loaded with zero and an interrupt flag is set. This flag can be read in the PFLAGS register (bit 15).

A write to the timer register loads the maximum count and also resets the counter while a read returns the current count. A read or write clears the interrupt flag.

A prescaler divider provides a 1MHz effective clock to the timer, so, it counts microseconds.

1.5 Interrupts

In the system there are 5 interrupt sources that are handled with an interrupt-enable register and a 3-bit priority encoder. The GUS16-v6 core has a design parameter, VECTORBASE, that allows the selection of a convenient base address for the interrupt jump table. In our case this address is 0x100 because it is a good idea to reserve the addresses below 0x100 for program variables as these addresses can be loaded into a pointer register with a single LDI instruction (the I/O registers are also mapped into this area for the same reason). The interrupt sources are detailed in the next table:

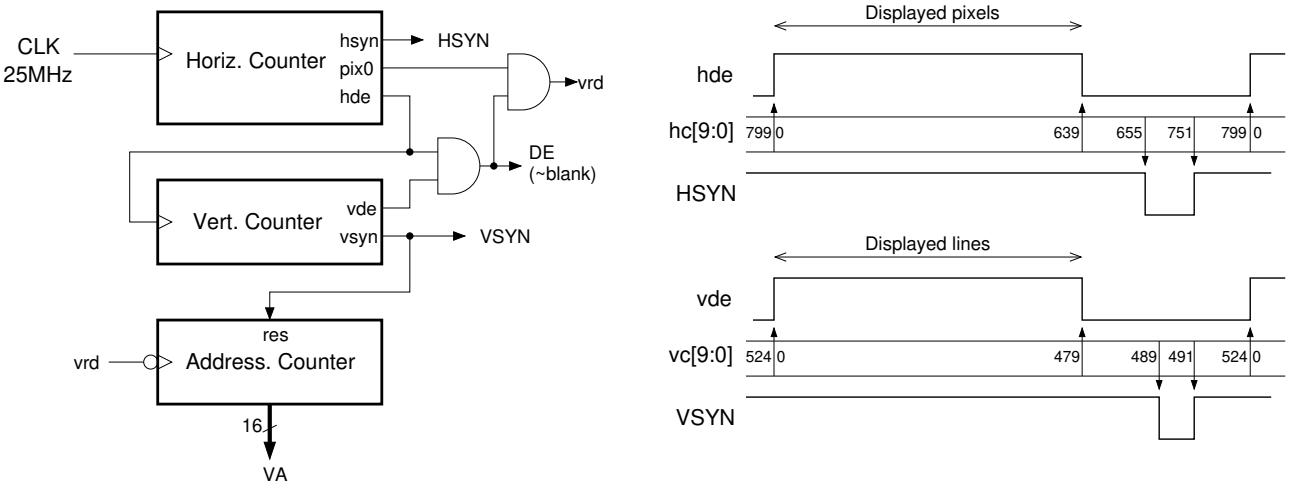


Figure 8: Timing block of the video controller and its waveforms.

Source	INTEN bit	Priority	Jump address	Clear with
UART RX	#0	Highest	0x0100	UART data read
UART TX	#1		0x0104	UART data write
Timer Overflow	#2		0x0108	Read or write Timer
HDE falling edge	#3		0x010c	write to 0x0026
VDE falling edge	#4	Lowest	0x0110	write to 0x0027

HDE is the horizontal display enable signal, meaning an interrupt can be posted as soon as the horizontal retrace interval begins, and the vertical display enable interrupt is handled in a similar way. These two interrupt signals are asserted until a write is done to addresses 0x26 or 0x27. Only the write operation matters, the data written is ignored.

1.6 Video generation

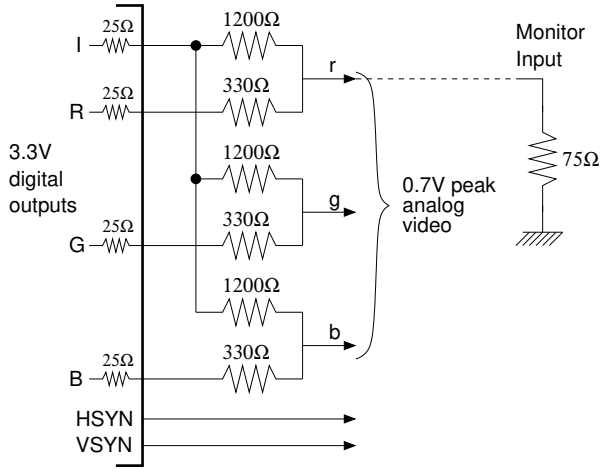
In the standard VGA mode a line is 32μs long, and by using a 25MHz clock, this results in a total of 800 pixels. Only 640 pixels are actually visible, the rest of the line is a blanking interval. During this blanking interval 'hde' is low and the 'HSYN' pulse is generated. The same happens in the vertical dimension: There are a total of 525 lines, with 480 lines being displayed. 'vde' is low and the 'VSYN' pulse is generated during the vertical blanking interval. A complete frames takes 16.8ms. A simplified diagram of the video controller timing is shown in figure 8 (the actual counter block is fully synchronous).

When both 'hde' and 'vde' are high the controller reads a 16-bit word from the external memory every 8 clock cycles, in fact when the three lower bits of the horizontal counter are 000. This word is stored into a double 8-bit shift register and a 2-bit pixel index is output every cycle. The content of a video word is:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
H[0]	H[1]	H[2]	H[3]	H[4]	H[5]	H[6]	H[7]	L[0]	L[1]	L[2]	L[3]	L[4]	L[5]	L[6]	L[7]

The MSB bits of these words are displayed first ({H[0],L[0]} is the pixel on the left), and the actual bit arrangement was chosen in order to ease the rendering of bitmapped text.

There are 4 possible values for pixels, meaning we can display up to four simultaneous colors on the screen. But these colors can be chosen from a 16-color palette. The palette RAM has a total of 16 bits, meaning all the required storage is just a single 16-bit, write-only, peripheral register with the following bits:



R,G,B	I	analog	%
0	0	0 V	0
0	1	0.16 V	23
1	0	0.55 V	78
1	1	0.7 V	100

Figure 9: IRGB to VGA interface

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
pixel 11				pixel 10				pixel 01				pixel 00			
I	R	G	B	I	R	G	B	I	R	G	B	I	R	G	B

This IRGB output is similar to that of the IBM CGA, with I being an additional level of white added to each color. In the SIMRETRO port the I bit accounts for 4/15 of every color component, while the R, G, and B, outputs accounts for the remaining 11/15 (the SIMRETRO card has a 4-bit DAC for each color component). For other simpler DACs these levels will depend on the actual resistor values used (an example is presented in figure 9).