Calculator

User Guide

Jorge Simões
jorgesimoes@tecnico.ulisboa.pt
José Carolino
jose.carolino@tecnico.ulisboa.pt

October 20, 2019

Contents

1 Introduction **7**

2 Block Diagram **7**

2.1 picoVersat 7

2.2 General Purpose Register File 7

2.3 Multiplier 8

2.4 VGA Controller 8

2.5 7-Segment Display Controller 8

2.6 Button Controller 8

2.7 Video Encoder 8

3 Interface Signals **9**

4 Program **10**

5 Memory Map **13**

6 Peripherals **14**

6.1 Multiplier 14

6.2 General purpose register 15

6.3 7-Segment display controller 15

6.4 Button Controller 16

6.5 VGA Controller 17

6.6 Video Encoder 19

7 Implementation Results **20**

8 Conclusions **20**

List of Tables

1 Interface signals. 9
2 Memory map base addresses 13
3 Multiplier interface signals. 14
4 Memory map of Multiplier 15
5 7-Segment diplay interface signals. 16
6 Memory map of 7-Segment display controller 16
7 Button controller interface signals. 17
8 Memory map of button controller controller 17
9 VGA controller interface signals. 19
10 Video decoder interface signals. 19
11 Memory map of video encoder 20

List of Figures

1 Calculator block diagram . . . 7
2 Calculator's symbol . . . 9
3 Flowchart of the operand's selection . . . 10
4 Flowchart of the operator's selection . . . 11
5 Flowchart of program . . . 12
6 Symbol of multiplier . . . 14
7 Block Diagram of multiplier . . . 14
8 Symbol of general purpose register . . . 15
9 Symbol of 7-Segment display controller . . . 15
10 Block Diagram of 7-Segment display controller . . . 16
11 Symbol of button controller . . . 16
12 Block Diagram of button controller . . . 17
13 Symbol of VGA controller . . . 18
14 Block Diagram of VGA controller . . . 18
15 Symbol of the video encoder . . . 19
16 Block Diagram of Video Encoder . . . 19

1 Introduction

Calculator is a simple 2 operands calculator with a small set of operations. It is capable of doing sums, subtractions, multiplications, exact divisions, powers, factorials and exact square roots. The numbers and the operations are both chosen using the buttons and displayed on the 7 segment display. The result of the operation and its operands are displayed on a VGA display.

This program uses picoVersat hardware controller and has peripherals to control the buttons, 7 segment display and VGA display. It has a multiplier as peripheral because it uses the built in multiplier of FPGA. Calculator is design to be implemented on a Basys2 board but can easily be ported to another board.

2 Block Diagram

The Calculator block diagram is show in Fig. 1. This system on chip named Calculator contains picoVersat hardware controller and several peripherals. The peripherals are a multiplier, a general purpose register file, a 7-Segment display controller, a button controller, a VGA controller and a Video Encoder.

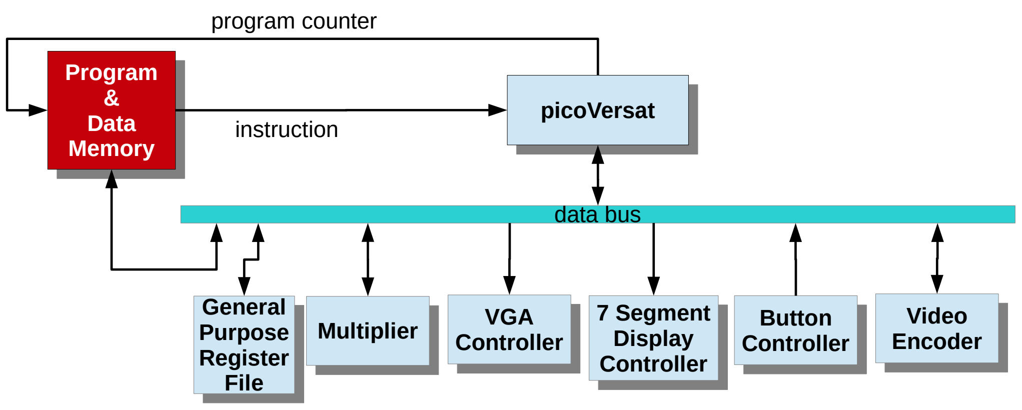

Figure 1: Calculator block diagram

2.1 picoVersat

The hardware controller used by this SoC is picoVersat. PicoVersat is designed to act like a hardware controller but in this case it is also used for calculations. All operations that are available are computed with picoVersat. The software that runs on picoVersat is capable of doing this operations on signed integers: sums, subtractions, multiplications, divisions. It can also do the following operations on unsigned integers: powers, factorials and square roots.

2.2 General Purpose Register File

The general purpose register file consists of 16 registers with 32 bits each. They are used to store intermediate values and the result of the last operation.

2.3 Multiplier

The multiplier is a peripheral designed to do all of the multiplications required by the operations, besides the multiplication operation.

2.4 VGA Controller

The VGA controller is used so the program can output to the display screen the numbers chosen by the user, the operation symbol and the result of said operation. The controller passes through all pixels of the screen and for each one of them displays the desired colour using an 8-bit pallet.

2.5 7-Segment Display Controller

The 7-Segment display controller displays, when choosing the input values, the represented number in hexadecimal form and, when choosing which operation to do, the operation abbreviation.

2.6 Button Controller

The button controller purpose is so the user can make use of the 3 buttons, using them to choose which operation to do, to use the last operation result as an input number for the current operation and to increase the value of the input number.

2.7 Video Encoder

The Video Encoder is a peripheral with a memory bank of the 10 digits and the operation symbols represented in pixels to be displayed on the screen. The encoder receives a signal corresponding to one of digits or symbols and sends its corresponding signal to then be displayed.

3 Interface Signals

The symbol of this system is shown in Fig- 2 and the interface signals of the Calculator are described in Table 1.

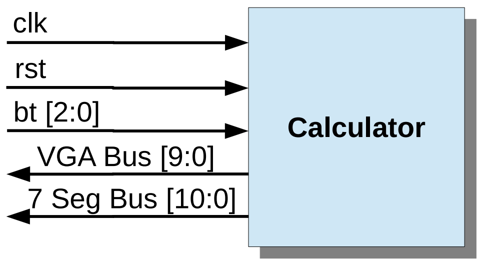

Figure 2: Calculator's symbol

Name	Direction	Description
clk	IN	Clock signal.
rst	IN	Reset signal.
Inputs Bus Interface		
bt[2:0]	IN	Buttons.
VGA Bus Interface		
HS	OUT	Horizontal sync for VGA.
VS	OUT	Vertical sync for VGA.
red[2:0]	OUT	VGA Red.
green[2:0]	OUT	VGA Green.
blue[1:0]	OUT	VGA Blue.
7 Seg Bus Interface		
enable[3:0]	OUT	enable the digit.
seg[6:0]	OUT	7 segments.

Table 1: Interface signals.

4 Program

This system is capable of doing two operands operations. Each operand has 16 bits and the result can have 32 bits. Each operand is chosen with the help of the buttons ”Enter” and ”Rotate”. For each operand 4 bits are chosen at a time. The ”Rotate” button rotates the number between 0 and 15, in hexadecimal 0 to F, and the ”Enter” button chooses these 4 bits. It starts with the 4 least significant bits and in four times one operand is picked. The operand is printed on the 7-Segment display while being chosen. The flowchart for the operation of choosing an operand is shown in Fig 3.

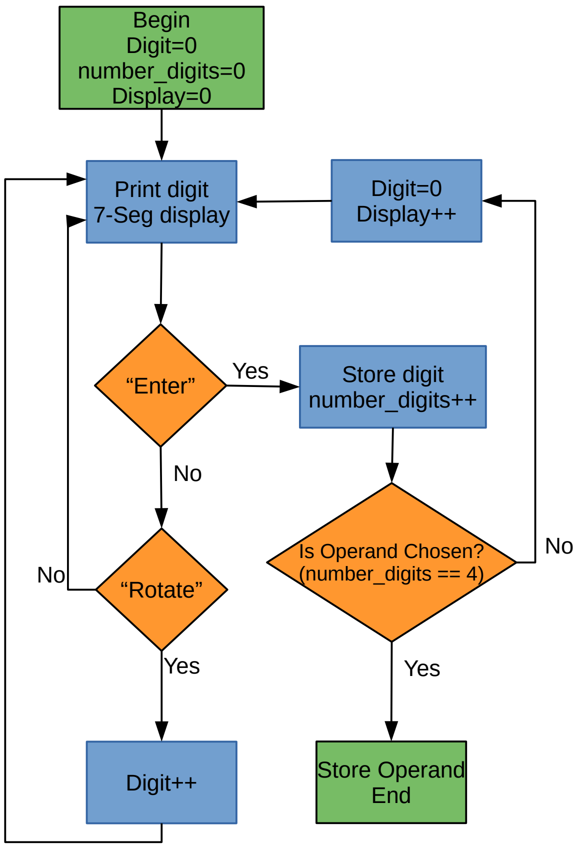

Figure 3: Flowchart of the operand’s selection

Choosing the operation is like choosing the operand. The ”Rotate” button rotates between the operations, while they are being printed in the 7-Segment display, and the ”Enter” button selects the operation. The flowchart for the operator’s selection is shown in Fig 4.

The order that the operands and operation are chosen is 1^{st} operand, operator and the last 2^{nd} operand. When the operand or operator is select they are printed on VGA display. When all 3 are selected,the result is computed and printed on the VGA display. At this point the result can be used as the 1^{st} operand in the next calculation using the ”Enter” button or can it be discarded using the ”Clear” button. The top-level routine is shown in Fig 5.

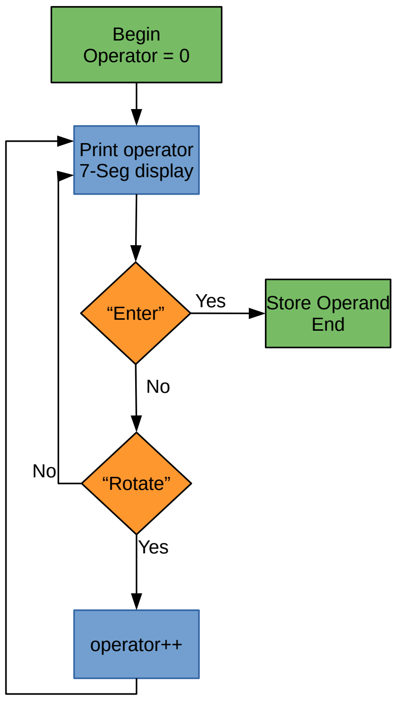

Figure 4: Flowchart of the operator's selection

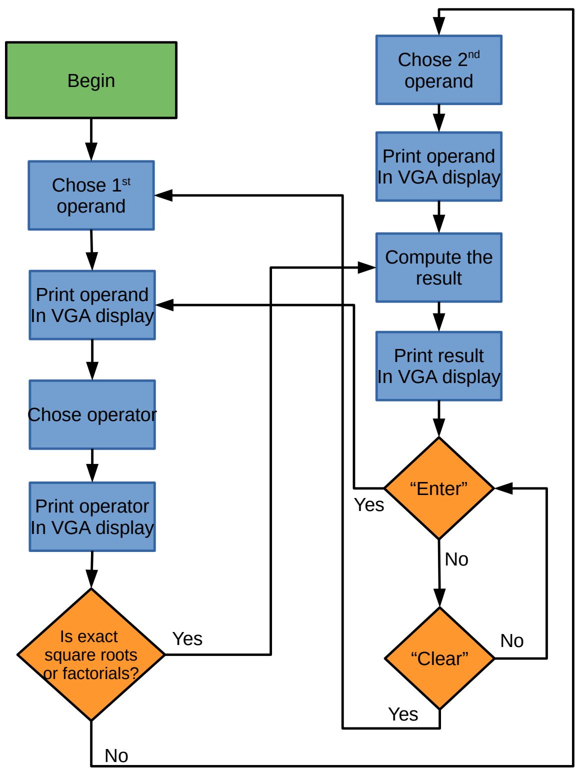
Begin
Chose 1st operand
Print operand In VGA display
Chose operator
Print operator In VGA display
Is exact square roots or factorials?
Yes
No
Chose 2nd operand
Print operand In VGA display
Compute the result
Print result In VGA display
"Enter"
Yes
No
"Clear"
No
Yes

Figure 5: Flowchart of program

The computation of the result depends on the operator. For sums and subtractions the method used is the picoVersat instruction for that purpose. In case of multiplications it is used the multiplier peripheral. In case of exact divisions the method used is an algorithm with shifts and subtractions. In case the operator is powers the algorithm used is described in Algorithm 1. For factorials and exact square roots the second operand is ignored. In case of factorial consist in multiplications and for exact square roots roots the algorithm is shown in Algorithm 2.

Algorithm 1: Powers

```
Powers(pow, exp)
accumPow:= 1
while exp $\geq$ 1 do
    if exp least significant bit is 1 then
        accumPow:= accumPow*pow
    end
    exp:= exp<<1
    pow:= pow*pow
end
return accumPow
```

Algorithm 2: Exact Square Root

```
exactsquareroots(number)
left:=0
right:=number
while right$\times$right $\geq$ number do
    if ((right+left)<<1)$\times$((right+left)<<1) $\geq$ number then
        right = (right+left)<<1
    else
        left = (right+left)<<1
    end
end
return right
```

5 Memory Map

The memory map of the system, as seen by picoVersat programs, is given in Table 2.

Mnemonic	Address	Read/Write	Read Latency	Description
REGF_BASE	512	Read+Write	0	Register file peripheral
MULT_BASE	700	Read+Write	0	Multiplier peripheral
7SEG_BASE	750	Write only	0	7-Seg display periheral
BUTT_BASE	800	Read only	0	Button controller periheral
VGA_BASE	850	Write only	0	VGA controller periheral
VIDENC_BASE	900	Read+Write	0	Video Enconder periheral

Table 2: Memory map base addresses

6 Peripherals

The SoC Calculator has picoVersat as processor/micro-controller and several peripherals as seen before. In this section for each peripheral there is a small description of how is works and information on inputs, outputs and registers. The picoVersat has a address decoder that enables each peripheral.

6.1 Multiplier

The purpose of this peripheral is to do multiplications of 16 bits. There are 2 registers for the two operators and a third register to store the result. The multiplications could be done in software but it is in hardware. This happens because the fpga's built-in multiplier is used. The symbol of this peripheral can be seen on Fig 15 and a table with descriptions of inputs and ouput in Table 3.

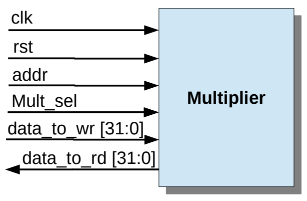

Figure 6: Symbol of multiplier

Name	Direction	Description
clk	IN	Clock signal.
rst	IN	Reset signal.
mult_sel	IN	select for the peripheral.
addr	IN	select for the register to write.
data_to_wr[31:0]	IN	data do write.
data_to_rd[31:0]	OUT	data to read from register.

Table 3: Multiplier interface signals.

The multipliers peripheral block diagram is shown in Fig 7. The register RegMultA and RegMultB are the operands and RegMultC stores the result of multiplication. In Table 4 is shown the memory map for this peripheral.

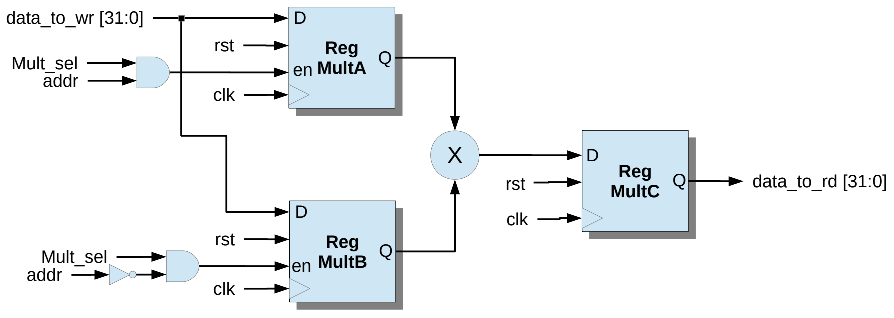

Figure 7: Block Diagram of multiplier

Mnemonic	Address	Read/Write	Read Latency	Description
RegMultA	0	Write only	NA	Register for operand
RegMultB	1	Write only	NA	Register for operand
RegMultC	2	Read	0	Register for result

Table 4: Memory map of Multiplier

6.2 General purpose register

General purpose register is a peripheral that picoVersat already contains. It has 16 registers that can be read and written for the software that runs on the processor.

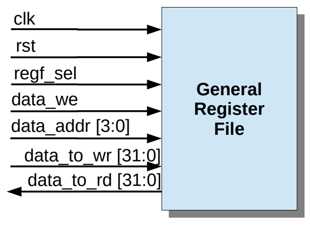

Figure 8: Symbol of general purpose register

6.3 7-Segment display controller

The purpose of this peripheral is to show the operands and the operator when they are being chosen. The operators are 16bit and they are displayed in hexadecimal so one operand will fill the whole 7-segment display. The fpga just has 7 outputs for this display, so in each clock cycle just one of the digits in the display will be illuminated. This will not be visible to human eye because clock is too high. The signal that chooses which digit will be activated is the enable, 1 bit for each digit. The symbol for this peripheral is shown in Fig 9. A table with descriptions of inputs and output in Table 5.

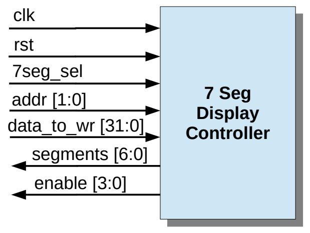

Figure 9: Symbol of 7-Segment display controller

This peripheral has 4 registers that correspond to the 4 digits.The enable is the signal that chooses the digit, so must only have one bit '1' and the remaining bits '0'. Every clock cycle enable does a left circular shift and the register used for output is the one corresponding to that digit. The controller has an encoder that translates a number, 0 to 15, to the bits for each segment in a digit. The block diagram is show in Fig 10 and the memory map for this peripheral in Table 6.

Name	Direction	Description
clk	IN	Clock signal.
rst	IN	Reset signal.
addr[1:0]	IN	select for register to write.
7seg_sel	IN	Peripheral select.
data_to_wr[31:0]	IN	data do write.
segments[6:0]	OUT	signal to the 7segments of 1 display.
enable[3:0]	OUT	enable of sgments.

Table 5: 7-Segment diplay interface signals.

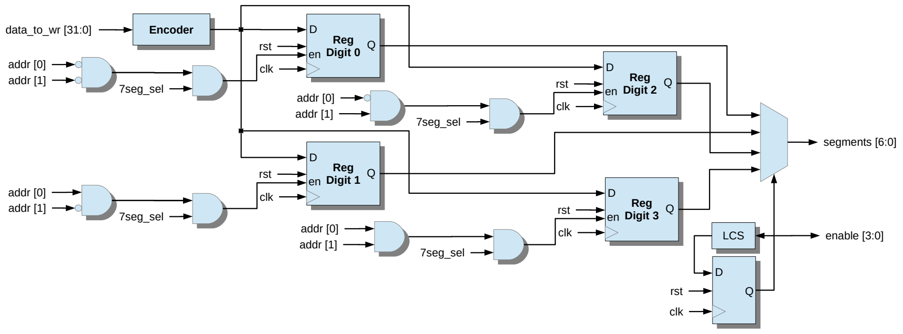

Figure 10: Block Diagram of 7-Segment display controller

Mnemonic	Address	Read/Write	Read Latency	Description
RegDigit0	0	Write only	NA	Register for digit
RegDigit1	1	Write only	NA	Register for digit
RegDigit2	2	Write only	NA	Register for digit
RegDigit3	3	Write only	NA	Register for digit

Table 6: Memory map of 7-Segment display controller

6.4 Button Controller

The buttons are used to select the operand's digits and the operations. The button controller symbol is shown in Fig 11 and a table with the descriptions of inputs and outputs in Table 7.

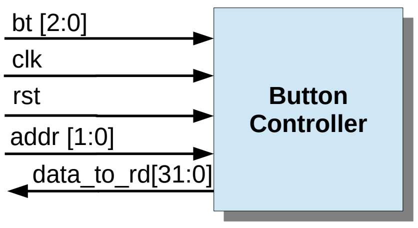

Figure 11: Symbol of button controller

This peripheral receives the signal of the buttons and stores the state on a register. Each button has a register so in total there are 3 registers. When the buttons are toggled they cause a unpredictable bounce in the signal. This is not desired because it affects how the system works. So the button signals are connected to a debouncer to suppress this. The block diagram of this peripheral is shown in Fig 12 and the memory map in Table 8.

Name	Direction	Description
clk	IN	Clock signal.
rst	IN	Reset signal.
addr[1:0]	IN	Select register to read.
bt[2:0]	IN	signal from 3 buttons.
data_to_rd[31:0]	OUT	data do read.

Table 7: Button controller interface signals.

The debouncer consists of a register, preseted with a number to be defined and the signal from the button that presets the register. In each clock cycle the register decreases it's stored value by one and when it reaches 0 the output signal of the debouncer is '1'.

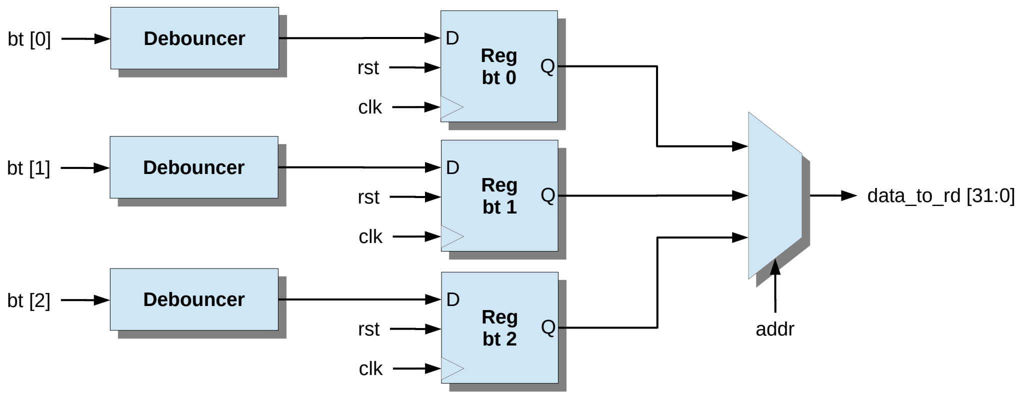

Figure 12: Block Diagram of button controller

Mnemonic	Address	Read/Write	Read Latency	Description
RegBT0	0	Read only	0	Register for button
RegBT1	1	Read only	0	Register for button
RegBT2	2	Red only	0	Register for button

Table 8: Memory map of button controller controller

6.5 VGA Controller

VGA controller is the peripheral that controls the signals entering VGA display. This peripheral generates a horizontal sync signal, vertical sync signal and the red, green and blue signal for each pixel. There is a video memory that contains in this case a '1' if the pixel is too be drawn white and '0' for it to be drawn black. Because this system only uses some space in the display just a little area of display is stored in the memory. The symbol for this peripheral is shown in Fig 13. The inputs and outputs description can be consulted in Table 9.

The block diagram is shown in Fig 14. The memory map for this peripheral is yet to be defined.

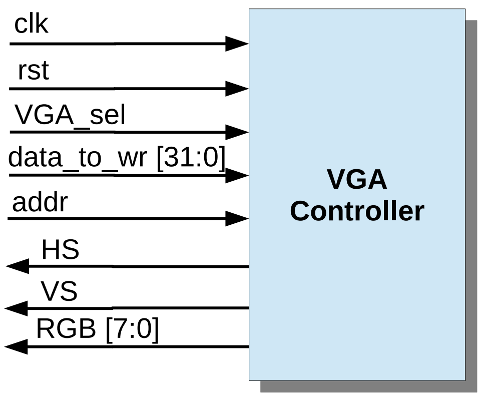

Figure 13: Symbol of VGA controller

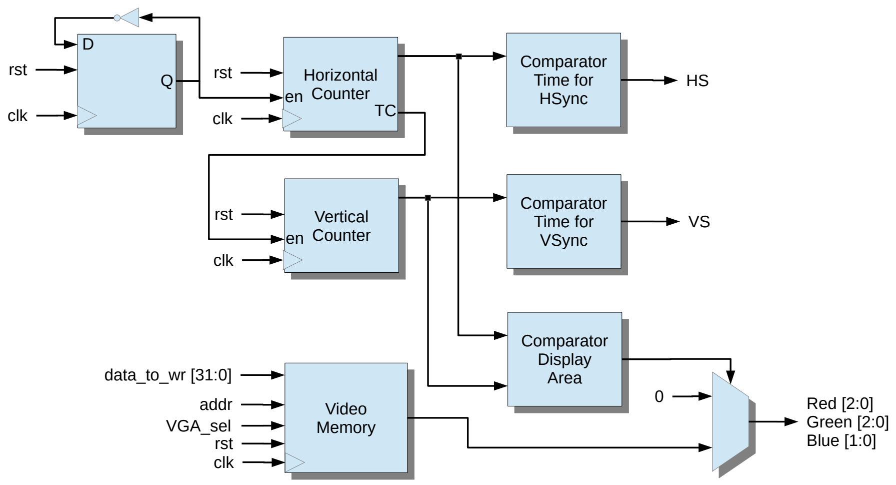

Figure 14: Block Diagram of VGA controller

Name	Direction	Description
clk	IN	Clock signal.
rst	IN	Reset signal.
enable_VGA	IN	enable for this peripheral.
addr	IN	address of register in this peripheral.
data_to_wr[31:0]	IN	data to write.
HS	OUT	horizontal sync.
VS	OUT	vertical sync.
red[2:0]	OUT	red for vga
green[2:0]	OUT	green for vga
blue[1:0]	OUT	blue for vga

Table 9: VGA controller interface signals.

6.6 Video Encoder

This peripheral translates a decimal number, or a operation to pixels. The VGA display is divided in blocks of 4x6 pixels and in each block a character will be drawn. The video encoder has a ROM with the pixels of all characters possible. The characters are the digits from 0 to 9 and the operations characters.

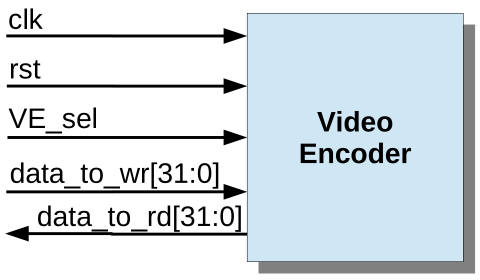

Figure 15: Symbol of the video encoder

Name	Direction	Description
clk	IN	Clock signal.
rst	IN	Reset signal.
data_to_rd[31:0]	IN	data to read.
data_to_wr[31:0]	OUT	data to write.

Table 10: Video decoder interface signals.

The program will write the character to translate to a register and in that address the ROM will have a word that correspond to the 24 pixels. In Fig 16 it can be seen the block diagram and memory map in Table 11.

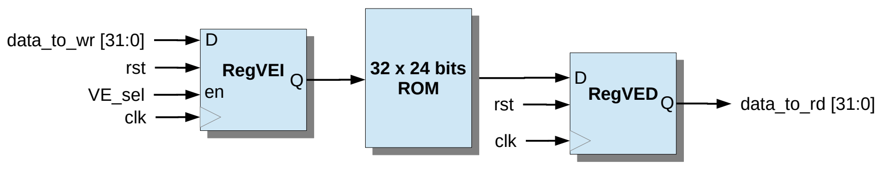

Figure 16: Block Diagram of Video Encoder

Mnemonic	Address	Read/Write	Read Latency	Description
RegVEI	0	Write only	0	Register for character
RegVEO	1	Read only	0	Register with pixels

Table 11: Memory map of video encoder

7 Implementation Results

8 Conclusions