

Universidad Nacional Autónoma de México
Facultad de Ingeniería
Ingeniería en Computación

Diseño Digital VLSI
Proyecto 04: ALU Automatizada

Alumnos:

- Kennedy Villa Carolina
- Murrieta Villegas Alfonso
- Reza Chavarria Sergio Gabriel
- Valdespino Mendieta Joaquin

Grupo: 5

Profesora: Elizabeth Fonseca Chávez

Fecha de entrega: 15 de noviembre del 2020

Proyecto 04: ALU Automatizada

Objetivos

- El alumno realizará la conexión de componentes, estos revisados durante el curso, para la creación de una Unidad Lógica Aritmética que muestre las operaciones y resultados, de manera automática, por medio del puerto VGA.

Introducción

Para este proyecto se necesitará el uso de diferentes componentes revisadas durante el curso. Los componentes serán una memoria ROM, unidad para solo almacenamiento de datos y su consulta, ALU, circuito utilizado para el cálculo de operaciones aritméticas y lógicas, de la unidad de VGA y estos componentes se unirán a partir de un sistema secuencial, ya que necesitamos que realice las operaciones de manera automática.

Desarrollo

Para la realización de este proyecto, se necesitó de la visualización de las operaciones y los resultados de la ALU. Estas se tienen que contemplar tanto en la parte de la salida (diseño de cada operación) y de la automatización (contador para realizar los cambios). Al tener el conjunto de esto, se realizará la conexión de las memorias ROM para guardar los datos de entrada y el contador para el cambio de operaciones con la ALU. Una vez teniendo las conexiones necesarias, este conjunto se unió con la salida proporcionada con el puerto VGA.

Conclusiones

Al tener el conjunto de componentes básicos, como lo son la ALU o los componentes para el uso del puerto VGA, se pudo realizar un proyecto complejo. Esto nos implica que los siguientes proyectos de la materia serán desafiantes para su conceptualización, análisis y resolución.

Bibliografía

- Fonseca E. (2020) VGA 3. Encontrado el 12 de noviembre de 2020, en <https://youtu.be/dDPRDm7rY7Y>
- Fonseca E. (2020) ULA. Encontrado el 12 de noviembre de 2020 en <https://youtu.be/8vszGhGa1RI>

Anexo

Para este proyecto se retomaron las operaciones realizadas que ejecuta la ALU.

Cin	Sel(2)	Sel(1)	Sel(0)	MUX B	Unidad	Operación	Función
0	0	0	0	'0'	UA	$\text{suma}(A + '0')$	$F = A$
0	0	0	1	\bar{B}	UA	$\text{suma}(A + \bar{B})$	$F = A + \bar{B}$
0	0	1	0	B	UA	$\text{suma}(A + B)$	$F = A + B$
0	0	1	1	'1'	UA	$\text{suma}(A + '1')$	$F = A - 1 = A - -$
0	1	0	0	'0'	UL	AND	$F = A AND B$
0	1	0	1	\bar{B}	UL	OR	$F = A OR B$
0	1	1	0	B	UL	XOR	$F = A XOR B$
0	1	1	1	'1'	UL	NOT	$F = NOT A$
1	0	0	0	'0'	UA	$\text{resta}(A - '0')$	$F = A + 1$
1	0	0	1	\bar{B}	UA	$\text{resta}(A - \bar{B})$	$F = A + \bar{B} + 1 = A - B$
1	0	1	0	B	UA	$\text{resta}(A - B)$	$F = A + B + 1$
1	0	1	1	'1'	UA	$\text{resta}(A - '1')$	$F = A$
1	1	0	0	'0'	UL	AND	$F = A AND B$
1	1	0	1	\bar{B}	UL	OR	$F = A OR B$
1	1	1	0	B	UL	XOR	$F = A XOR B$
1	1	1	1	'1'	UL	NOT	$F = NOT A$

Para la representación de las operaciones de la ALU con el puerto VGA se tomará como base la columna de operaciones. Se modificará la salida para cada caso dependiendo de las entradas correspondientes de Cin y Sel.

Para el proceso automático se realizará un contador que contiene 16 estados. Este componente secuencial se usará la máquina de tipo Moore para utilizar directo el estado como salida. Este contador se conectará y representará a las entradas de la ALU (cin y sel).

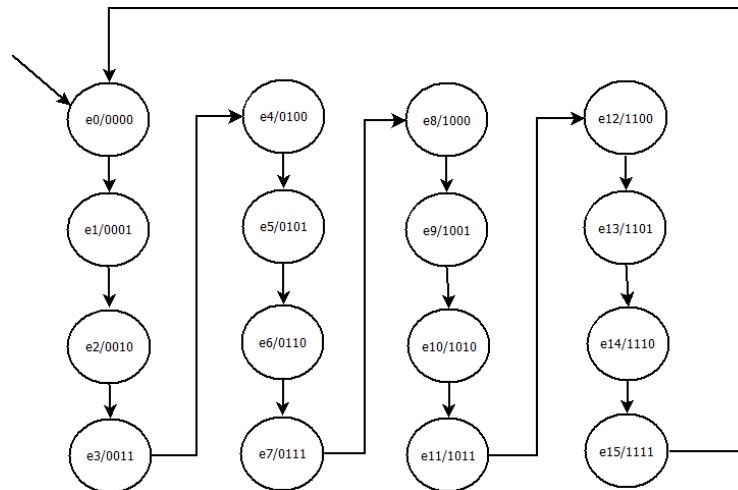


Ilustración 1: Contador utilizado

Para la entrada de la ALU se utilizó una ROM la cual contiene el código binario de los números de 3 bits (0 al 7). Además, se dio uso de otra ROM, la cual contiene la estructura en 7 segmentos de los números de 3 bits para la salida por el VGA, para las entradas al sistema.

Una vez teniendo la ejecución de este componente en general se conectará con los componentes de VGA. Tomará la estructura de los números de 7 segmentos para las operaciones aritméticas y la representación de la entrada activada o desactivada para las operaciones lógicas.

Se tomará en cuenta el estado que nos obtiene el contador. Estos serán los siguientes casos para cambiar la representación de las operaciones.

- Cin: Cambio entre suma o resta
- SEL (2): Cambio entre operación aritmética y lógica
- SEL (0 to 1): Cambio de operaciones
 - Caso UA
 - 00: Operación con solo ceros.
 - 01: B negada
 - 10: B
 - 11: Operación con solo unos.
 - Caso UL
 - 00: AND (X)
 - 01: OR (+)
 - 10: XOR (+ con una línea)
 - 11: NOT (Línea arriba del a, no se mostrará b)

Código

ALU

```
Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA - [...]
```

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3
4 entity mux4x1 is port(
5     b: in std_logic_vector(2 downto 0);
6     s: in std_logic_vector(1 downto 0);
7     sa1: out std_logic_vector(2 downto 0)
8 );
9 end mux4x1;
10
11 architecture arqmux of mux4x1 is
12 begin
13     with s select
14         sa1 <= (others => '0') when "00",
15                not b         when "01",
16                b             when "10",
17                (others => '1') when "11",
18                (others => '0') when others;
19
20 end arqmux;
```

0% 00:00:00

Código 1: Multiplexor de estado de B

```
Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA - [...]
```

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 use ieee.std_logic_unsigned.all;
5
6 entity sum is port(
7     a,b: in std_logic_vector(2 downto 0);
8     cin: in std_logic;
9     sa1sum: out std_logic_vector(2 downto 0);
10    cout: out std_logic
11 );
12 end sum;
13
14 architecture arqsum of sum is
15 signal mid: std_logic_vector(3 downto 0);
16
17 begin
18     mid<=('0'&a)+('0'&b)+cin;
19     cout<=mid(3);
20     sa1sum<=mid(2 downto 0);
21
22 end architecture arqsum;
```

0% 00:00:00

Código 2: Suma AU

```

1  |library ieee;
2  |use ieee.std_logic_1164.all;
3  |use ieee.std_logic_unsigned.all;
4  entity UA is port(
5  |    a,b: in std_logic_vector(2 downto 0);
6  |    s0: in std_logic_vector(1 downto 0);
7  |    cin: in std_logic;
8  |    salsum: out std_logic_vector(2 downto 0);
9  |    cout: out std_logic
10 |);
11 |end entity;
12 |
13 |architecture arq_UA of UA is
14 |    signal sal: std_logic_vector(2 downto 0);
15 |    begin
16 |        u1: entity work.mux4x1(arqmux) port map(b,s0,sal);
17 |        u2: entity work.sum(arqsum) port map(a, sal, cin,salsum, cout);
18 |    end arq_UA;
19 |

```

Código 3: Unidad Aritmética

```

1  |library ieee;
2  |use ieee.std_logic_1164.all;
3  entity UL is port(
4  |    a,b: in std_logic_vector(2 downto 0);
5  |    sel: in std_logic_vector(1 downto 0);
6  |    sallog: out std_logic_vector(2 downto 0)
7  |);
8  |end entity;
9  |
10 |architecture arq_UL of UL is
11 |    signal cand,cor,cxor,cnot: std_logic_vector(2 downto 0);
12 |    begin
13 |        cand<=a and b;
14 |        cor <=a or b;
15 |        cxor<=a xor b;
16 |        cnot<=not a;
17 |
18 |        with sel select
19 |            sallog <=
20 |                cand when "00",
21 |                cor  when "01",
22 |                cxor when "10",
23 |                cnot when "11";
24 |    end arq_UL;
25 |

```

Código 4: Unidad Lógica

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 entity TOP_ALU is port (
4     a, b: in std_logic_vector (2 downto 0);
5     sel: in std_logic_vector (2 downto 0);
6     cin: in std_logic;
7     disp0, disp1, disp2: out std_logic_vector(6 downto 0);
8     salfinal: out std_logic_vector(3 downto 0)
9 );
10 end entity;
11
12 architecture arq_alu of TOP_ALU is
13     signal SAL_UA, SAL_UL: std_logic_vector(2 downto 0);
14     signal cout: std_logic;
15 begin
16
17     arit: entity work.UA (arq_UA) port map (a,b,sel(1 downto 0),cin,SAL_UA,cout);
18     log: entity work.UL (arq_UL) port map (a,b,sel(1 downto 0),SAL_UL);
19     disp_sal: entity work.display(arq_disp)
20         port map (SAL_UA,SAL_UL,sel,cout,disp0,disp1,disp2,salfinal);
21
22 end architecture;
23

```

1% 00:00:08

Código 5: ALU

```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 entity display is port(
4     UA, UL: in std_logic_vector(2 downto 0);
5     sel: in std_logic_vector (2 downto 0);
6     cout: in std_logic;
7     disp0, disp1, disp2: out std_logic_vector(6 downto 0);
8     salfinal: out std_logic_vector(3 downto 0)
9 );
10 end entity;
11
12 architecture arq_disp of display is
13     signal uaparcial, uaparcial2, ulparcial: std_logic_vector( 3 downto 0);
14 begin
15
16     process(sel,cout,UA,UL)
17     begin
18         --UA
19         if(sel(2)='0') then
20             --resta
21             if (sel(1 downto 0) ="01" OR sel(1 downto 0) ="11") then
22                 uaparcial <=cout&UA;
23
24             case uaparcial is
25                 when "0000" => disp0 <="1000000";--0
26                 when "0001" => disp0 <="1111001";--1
27                 when "0010" => disp0 <="0100100";--2
28                 when "0011" => disp0 <="0110000";--3
29                 when "0100" => disp0 <="0011001";--4
30                 when "0101" => disp0 <="0010010";--5
31                 when "0110" => disp0 <="0000010";--6
32                 when "0111" => disp0 <="1111000";--7
33                 when "1000" => disp0 <="1000000";--0
34                 when "1001" => disp0 <="1111001";--1
35                 when "1010" => disp0 <="0100100";--2
36                 when "1011" => disp0 <="0110000";--3
37                 when "1100" => disp0 <="0011001";--4
38                 when "1101" => disp0 <="0010010";--5
39                 when "1110" => disp0 <="0000010";--6
40                 when "1111" => disp0 <="1111000";--7
41                 when others => disp0 <="1000000";
42             end case;
43
44             disp1 <= "1000000";
45             disp2 <= "1111111";
46             salfinal <= uaparcial;
47             --suma y casos extra
48             else
49                 uaparcial2 <= cout&UA;
50

```

2% 00:00:48

```

49      else
50      uaparcial2 <= cout&UA;
51
52      case uaparcial2 is
53      when "0000" => disp0 <= "1000000";--0
54      when "0001" => disp0 <= "1111001";--1
55      when "0010" => disp0 <= "0100100";--2
56      when "0011" => disp0 <= "0110000";--3
57      when "0100" => disp0 <= "0011001";--4
58      when "0101" => disp0 <= "0010010";--5
59      when "0110" => disp0 <= "0000010";--6
60      when "0111" => disp0 <= "1111000";--7
61      when "1000" => disp0 <= "0000000";--8
62      when "1001" => disp0 <= "0011000";--9
63      when "1010" => disp0 <= "1000000";--10
64      when "1011" => disp0 <= "1111001";--11
65      when "1100" => disp0 <= "0100100";--12
66      when "1101" => disp0 <= "0110000";--13
67      when "1110" => disp0 <= "0011001";--14
68      when "1111" => disp0 <= "0010010";--15
69      when others => disp0 <= "1000000";
70      end case;
71
72      case uaparcial2 is
73      when "0000" => disp1 <= "1000000";--0
74      when "0001" => disp1 <= "1000000";--1
75      when "0010" => disp1 <= "1000000";--2
76      when "0011" => disp1 <= "1000000";--3
77      when "0100" => disp1 <= "1000000";--4
78      when "0101" => disp1 <= "1000000";--5
79      when "0110" => disp1 <= "1000000";--6
80      when "0111" => disp1 <= "1000000";--7
81      when "1000" => disp1 <= "1000000";--8
82      when "1001" => disp1 <= "1000000";--9
83      when "1010" => disp1 <= "1111001";--10
84      when "1011" => disp1 <= "1111001";--11
85      when "1100" => disp1 <= "1111001";--12
86      when "1101" => disp1 <= "1111001";--13
87      when "1110" => disp1 <= "1111001";--14
88      when "1111" => disp1 <= "1111001";--15
89      when others => disp1 <= "1000000";
90      end case;
91
92      disp2 <= "1111111";
93
94      salfinal <= uaparcial2;
95
96      end if;
97
98      --!!!

```

```

73      when "0000" => disp1 <= "1000000";--0
74      when "0001" => disp1 <= "1000000";--1
75      when "0010" => disp1 <= "1000000";--2
76      when "0011" => disp1 <= "1000000";--3
77      when "0100" => disp1 <= "1000000";--4
78      when "0101" => disp1 <= "1000000";--5
79      when "0110" => disp1 <= "1000000";--6
80      when "0111" => disp1 <= "1000000";--7
81      when "1000" => disp1 <= "1000000";--8
82      when "1001" => disp1 <= "1000000";--9
83      when "1010" => disp1 <= "1111001";--10
84      when "1011" => disp1 <= "1111001";--11
85      when "1100" => disp1 <= "1111001";--12
86      when "1101" => disp1 <= "1111001";--13
87      when "1110" => disp1 <= "1111001";--14
88      when "1111" => disp1 <= "1111001";--15
89      when others => disp1 <= "1000000";
90      end case;
91
92      disp2 <= "1111111";
93
94      salfinal <= uaparcial2;
95
96      end if;
97
98      --UL
99      else
100      uaparcial <= '0'&UL;
101      case uaparcial (0) is
102      when '1' => disp0 <= "1111001";
103      when '0' => disp0 <= "1000000";
104      when others => disp0 <= "1111111";
105      end Case;
106      case uaparcial (1) is
107      when '1' => disp1 <= "1111001";
108      when '0' => disp1 <= "1000000";
109      when others => disp1 <= "1111111";
110      end Case;
111      case uaparcial (2) is
112      when '1' => disp2 <= "1111001";
113      when '0' => disp2 <= "1000000";
114      when others => disp2 <= "1111111";
115      end Case;
116
117      salfinal <= '0'&UL;
118
119      end if;
120      end process;
121      end architecture arq_disp;

```

Código 6: Display ALU, usado para la estructura de las salidas de la ALU

Contador

```
Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA - [... - □ X
File Edit View Project Processing Tools Window Help Search altera.com
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_arith.all;
4 entity reloj_lento is port (
5     clk1: in std_logic;
6     led: buffer std_logic:= '0'
7 );
8 end reloj_lento;
9
10 architecture arqreloj_lento of reloj_lento is
11     signal conteo: integer range 0 to 30000000;
12 begin
13     process (clk1)
14     begin
15         if (clk1' event and clk1='1') then
16             conteo <= conteo+1;
17             if (conteo=30000000) then
18                 conteo <= 0;
19                 led <= not(led);
20             end if;
21         end if;
22     end process;
23 end arqreloj_lento;
```

Código 7: Reloj lento

```
Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA - [contad... - □ X
File Edit View Project Processing Tools Window Help Search altera.com
1 library ieee;
2 use ieee.std_logic_1164.all;
3 entity contador is port(
4     clk, reset: in std_logic;
5     sal: out std_logic_vector(3 downto 0)
6 );
7 end entity;
8
9 architecture arqcont of contador is
10     subtype state is std_logic_vector(3 downto 0);
11     signal present_state, next_state: state;
12     constant e0: state := "0000";
13     constant e1: state := "0001";
14     constant e2: state := "0010";
15     constant e3: state := "0011";
16     constant e4: state := "0100";
17     constant e5: state := "0101";
18     constant e6: state := "0110";
19     constant e7: state := "0111";
20     constant e8: state := "1000";
21     constant e9: state := "1001";
22     constant e10: state := "1010";
23     constant e11: state := "1011";
24     constant e12: state := "1100";
25     constant e13: state := "1101";
26     constant e14: state := "1110";
27     constant e15: state := "1111";
28
29 begin
30     process (clk)
31     begin
32         if rising_edge(clk) then
33             if (reset='0') then
34                 present_state <= e0;
35             else
36                 present_state <= next_state;
37             end if;
38         end if;
39     end process;
40     process (present_state)
41     begin
42         case present_state is
43             when e0 => next_state <= e1;
44             when e1 => next_state <= e2;
45             when e2 => next_state <= e3;
46             when e3 => next_state <= e4;
47             when e4 => next_state <= e5;
48             when e5 => next_state <= e6;
49             when e6 => next_state <= e7;
50             when e7 => next_state <= e8;
51             when e8 => next_state <= e9;
52             when e9 => next_state <= e10;
53             when e10 => next_state <= e11;
54             when e11 => next_state <= e12;
55             when e12 => next_state <= e13;
56             when e13 => next_state <= e14;
57             when e14 => next_state <= e15;
58             when e15 => next_state <= e0;
59             when others => next_state <= e0;
60         end case;
61         sal <= present_state;
62     end process;
63 end arqcont;
```

Código 8: Estados del contador

Memorias ROM

```
Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA - [...]
```

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 entity romDatos is port
5 (
6     bus_dir: in std_logic_vector(2 downto 0);
7     cs: in std_logic;
8     bus_datos: out std_logic_vector(2 downto 0)
9 );
10 end romDatos;
11 architecture arqromDatos of romDatos is
12     constant L0: std_logic_vector(2 downto 0):="000";
13     constant L1: std_logic_vector(2 downto 0):="001";
14     constant L2: std_logic_vector(2 downto 0):="010";
15     constant L3: std_logic_vector(2 downto 0):="011";
16     constant L4: std_logic_vector(2 downto 0):="100";
17     constant L5: std_logic_vector(2 downto 0):="101";
18     constant L6: std_logic_vector(2 downto 0):="110";
19     constant L7: std_logic_vector(2 downto 0):="111";
20     type memoria is array (7 downto 0) of std_logic_vector(2 downto 0);
21     constant mem_rom:memoria:=(L7,L6,L5,L4,L3,L2,L1,L0);
22     signal dato: std_logic_vector(2 downto 0);
23 begin
24     prom: process(bus_dir)
25     begin
26         dato<=mem_rom(conv_integer(bus_dir));
27     end process prom;
28     pbuf: process(dato,cs)
29     begin
30         if(cs='1') then
31             bus_datos<=dato;
32         else
33             bus_datos<=(others=>'Z');
34         end if;
35     end process pbuf;
36 end arqromDatos;
```

2% 00:00:48

Código 9: ROM con datos de entrada

```
Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA - [...]
```

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 entity rom is port
5 (
6     bus_dir: in std_logic_vector(2 downto 0);
7     cs: in std_logic;
8     bus_datos: out std_logic_vector(6 downto 0)
9 );
10 end rom;
11 architecture arqrom of rom is
12     constant L0: std_logic_vector(6 downto 0):="1000000";
13     constant L1: std_logic_vector(6 downto 0):="1111001";
14     constant L2: std_logic_vector(6 downto 0):="0100100";
15     constant L3: std_logic_vector(6 downto 0):="0110000";
16     constant L4: std_logic_vector(6 downto 0):="0011001";
17     constant L5: std_logic_vector(6 downto 0):="0010010";
18     constant L6: std_logic_vector(6 downto 0):="0000010";
19     constant L7: std_logic_vector(6 downto 0):="1111000";
20     type memoria is array (7 downto 0) of std_logic_vector(6 downto 0);
21     constant mem_rom:memoria:=(L7,L6,L5,L4,L3,L2,L1,L0);
22     signal dato: std_logic_vector(6 downto 0);
23 begin
24     prom: process(bus_dir)
25     begin
26         dato<=mem_rom(conv_integer(bus_dir));
27     end process prom;
28     pbuf: process(dato,cs)
29     begin
30         if(cs='1') then
31             bus_datos<=dato;
32         else
33             bus_datos<=(others=>'Z');
34         end if;
35     end process pbuf;
36 end arqrom;
```

2% 00:00:48

Código 10: ROM con estructuras de 7 segmentos

VGA

```

1  library ieee;
2  use ieee.std_logic_arith.all;
3  use ieee.std_logic_1164.all;
4  entity genMhz is port(
5      clk50mhz: in std_logic;
6      clk25mhz: buffer std_logic:= '0'
7  );
8  end genMhz;
9
10 architecture arggenMhz of genMhz is
11 begin
12 process(clk50mhz)
13 begin
14 if(clk50mhz' event and clk50mhz='1') then
15     clk25mhz<= not clk25mhz;
16 end if;
17 end process;
18 end arggenMhz;

```

Código 11: Divisor de frecuencia

```

1  --
2  --
3  -- FileName:          vga_controller.vhd
4  -- Dependencies:      none
5  -- Design Software:   Quartus II 64-bit Version 12.1 Build 177 SJ Full Version
6  --
7  -- HDL CODE IS PROVIDED "AS IS." DIGI-KEY EXPRESSLY DISCLAIMS ANY
8  -- WARRANTY OF ANY KIND, WHETHER EXPRESS OR IMPLIED, INCLUDING BUT NOT
9  -- LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
10 -- PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL DIGI-KEY
11 -- BE LIABLE FOR ANY INCIDENTAL, SPECIAL, INDIRECT OR CONSEQUENTIAL
12 -- DAMAGES, LOST PROFITS OR LOST DATA, HARM TO YOUR EQUIPMENT, COST OF
13 -- PROCUREMENT OF SUBSTITUTE GOODS, TECHNOLOGY OR SERVICES, ANY CLAIMS
14 -- BY THIRD PARTIES (INCLUDING BUT NOT LIMITED TO ANY DEFENSE THEREOF),
15 -- ANY CLAIMS FOR INDEMNITY OR CONTRIBUTION, OR OTHER SIMILAR COSTS.
16 --
17 -- Version History
18 -- Version 1.0 05/10/2013 Scott Larson
19 -- Initial Public Release
20 --
21 --
22
23 library ieee;
24 use ieee.std_logic_1164.all;
25
26 entity vga_controller is
27 generic(
28     h_pulse : integer := 96; --horizontal sync pulse width in pixels
29     h_bp    : integer := 48; --horizontal back porch width in pixels
30     h_pixels: integer := 640; --horizontal display width in pixels
31     h_fp    : integer := 16; --horizontal front porch width in pixels
32     h_pol   : std_logic := '0'; --horizontal sync pulse polarity (1 = positive, 0
33     v_pulse : integer := 2;  --vertical sync pulse width in rows
34     v_bp    : integer := 33; --vertical back porch width in rows
35     v_pixels: integer := 480; --vertical display width in rows
36     v_fp    : integer := 10; --vertical front porch width in rows
37     v_pol   : std_logic := '0'); --vertical sync pulse polarity (1 = positive, 0 =
38
39 port(
40     pixel_clk : in std_logic; --pixel clock at frequency of VGA mode being used
41     reset_n   : in std_logic; --active low asynchronous reset
42     h_sync    : out std_logic; --horizontal sync pulse
43     v_sync    : out std_logic; --vertical sync pulse
44     disp_ena  : out std_logic; --display enable ('1' = display time, '0' = blanking
45     column   : out integer;    --horizontal pixel coordinate
46     row      : out integer;    --vertical pixel coordinate
47     n_blank  : out std_logic;  --direct blanking output to DAC
48     n_sync   : out std_logic;  --sync-on-green output to DAC
49 end vga_controller;

```

```

Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - [vga_controller.vhd]
File Edit View Project Processing Tools Window Help
Search altera.com

51 ARCHITECTURE behavior OF vga_controller IS
52   CONSTANT h_period : INTEGER := h_pulse + h_bp + h_pixels + h_fp; --total number of pixel clocks in a row
53   CONSTANT v_period : INTEGER := v_pulse + v_bp + v_pixels + v_fp; --total number of rows in column
54 BEGIN
55   --n_blank <= '1'; --no direct blanking
56   --n_sync <= '0'; --no sync on green
57
58   PROCESS(pixel_clk, reset_n)
59   VARIABLE h_count : INTEGER RANGE 0 TO h_period - 1 := 0; --horizontal counter (counts the columns)
60   VARIABLE v_count : INTEGER RANGE 0 TO v_period - 1 := 0; --vertical counter (counts the rows)
61 BEGIN
62   IF(reset_n = '0') THEN --reset asserted
63     h_count := 0; --reset horizontal counter
64     v_count := 0; --reset vertical counter
65     h_sync <= h_pol; --deassert horizontal sync
66     v_sync <= v_pol; --deassert vertical sync
67     disp_ena <= '0'; --disable display
68     column <= 0; --reset column pixel coordinate
69     row <= 0; --reset row pixel coordinate
70   ELSE
71     ELIF(pixel_clk'EVENT AND pixel_clk = '1') THEN
72       --counters
73       IF(h_count < h_period - 1) THEN --horizontal counter (pixels)
74         h_count := h_count + 1;
75       ELSE
76         h_count := 0;
77         IF(v_count < v_period - 1) THEN --vertical counter (rows)
78           v_count := v_count + 1;
79         ELSE
80           v_count := 0;
81         END IF;
82       END IF;
83       --horizontal sync signal
84       IF(h_count < h_pixels + h_fp OR h_count > h_pixels + h_fp + h_pulse) THEN
85         h_sync <= NOT h_pol; --deassert horizontal sync pulse
86       ELSE
87         h_sync <= h_pol; --assert horizontal sync pulse
88       END IF;
89       --vertical sync signal
90       IF(v_count < v_pixels + v_fp OR v_count > v_pixels + v_fp + v_pulse) THEN
91         v_sync <= NOT v_pol; --deassert vertical sync pulse
92       ELSE
93         v_sync <= v_pol; --assert vertical sync pulse
94       END IF;
95     END IF;
96   END IF;
97 END PROCESS;
98 END behavior;
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

```

```

Text Editor - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - [vga_controller.vhd]
File Edit View Project Processing Tools Window Help
Search altera.com

71 row <= 0; --reset row pixel coordinate
72
73 ELIF(pixel_clk'EVENT AND pixel_clk = '1') THEN
74   --counters
75   IF(h_count < h_period - 1) THEN --horizontal counter (pixels)
76     h_count := h_count + 1;
77   ELSE
78     h_count := 0;
79     IF(v_count < v_period - 1) THEN --vertical counter (rows)
80       v_count := v_count + 1;
81     ELSE
82       v_count := 0;
83     END IF;
84   END IF;
85   --horizontal sync signal
86   IF(h_count < h_pixels + h_fp OR h_count > h_pixels + h_fp + h_pulse) THEN
87     h_sync <= NOT h_pol; --deassert horizontal sync pulse
88   ELSE
89     h_sync <= h_pol; --assert horizontal sync pulse
90   END IF;
91   --vertical sync signal
92   IF(v_count < v_pixels + v_fp OR v_count > v_pixels + v_fp + v_pulse) THEN
93     v_sync <= NOT v_pol; --deassert vertical sync pulse
94   ELSE
95     v_sync <= v_pol; --assert vertical sync pulse
96   END IF;
97   --set pixel coordinates
98   IF(h_count < h_pixels) THEN --horizontal display time
99     column <= h_count; --set horizontal pixel coordinate
100   END IF;
101   IF(v_count < v_pixels) THEN --vertical display time
102     row <= v_count; --set vertical pixel coordinate
103   END IF;
104   --set display enable output
105   IF(h_count < h_pixels AND v_count < v_pixels) THEN --display time
106     disp_ena <= '1'; --enable display
107   ELSE
108     disp_ena <= '0'; --blanking time
109   END IF;
110 END IF;
111 END PROCESS;
112 END behavior;
113
114
115
116
117
118
119
120

```

Código 12: Controlador de VGA

```

22 LIBRARY ieee;
23 USE ieee.std_logic_1164.all;
24
25 ENTITY hw_image_generator IS
26
27   GENERIC(
28     pixels_v: INTEGER := 640; --row that first color will persist until
29     pixels_h: INTEGER := 480; --column that first color will persist until
30   );
31   PORT(
32     disp_ena: row IN INTEGER; --display enable ('1' = display time, '0' = blanking time)
33     row: row OUT STD_LOGIC_VECTOR(3 DOWNTO 0) := (OTHERS => '0'); --row pixel coordinate
34     column: column OUT STD_LOGIC_VECTOR(3 DOWNTO 0) := (OTHERS => '0'); --column pixel coordinate
35     green: row OUT STD_LOGIC_VECTOR(3 DOWNTO 0) := (OTHERS => '0'); --red magnitude output to DAC
36     blue: row OUT STD_LOGIC_VECTOR(3 DOWNTO 0) := (OTHERS => '0'); --blue magnitude output to DAC
37
38   );
39   --Datos obtenidos de ALU
40   sat: in std_logic_vector(3 downto 0)
41   a, b in std_logic_vector(2 downto 0)
42   bus_A, bus_B, disp0, disp1, disp2: in std_logic_vector(6 downto 0)
43
44   -- unidades del contador en 7 segmentos
45   );
46 END hw_image_generator;
47
48 ARCHITECTURE behavior OF hw_image_generator IS
49 BEGIN
50
51   PROCESS(disp_ena, row, column, sat, a, bus_A, bus_B, disp0, disp1, disp2)
52   BEGIN
53     if (disp_ena = '1') then
54       if (sat(3) = '0') then
55
56         if ((row > 120 and row <= 140) AND (column = 70 and column <= 100)) then
57           if (bus_A(0) = '0') then
58             red <= (Others => '1');
59             green <= (Others => '0');
60             blue <= (Others => '0');
61           else
62             red <= (Others => '0');
63             green <= (Others => '0');
64             blue <= (Others => '0');
65           end if;
66         else
67           --if ((row > 140 and row <= 220) AND (column = 100 and column <= 115)) then
68           if (bus_A(1) = '0') then
69             red <= (Others => '1');
70             green <= (Others => '0');
71             blue <= (Others => '0');
72           else
73             red <= (Others => '0');
74             green <= (Others => '0');
75             blue <= (Others => '0');
76           end if;
77         else if ((row > 240 and row <= 320) AND (column = 100 and column <= 115)) then
78           if (bus_A(2) = '0') then
79             red <= (Others => '1');
80             green <= (Others => '0');
81             blue <= (Others => '0');
82           else
83             red <= (Others => '0');
84             green <= (Others => '0');
85             blue <= (Others => '0');
86           end if;
87         else if ((row > 320 and row <= 340) AND (column = 70 and column <= 100)) then
88           if (bus_A(3) = '0') then
89             red <= (Others => '1');
90             green <= (Others => '0');
91             blue <= (Others => '0');
92           else
93             red <= (Others => '0');
94             green <= (Others => '0');
95             blue <= (Others => '0');
96           end if;
97         else
98           red <= (Others => '0');
99           green <= (Others => '0');
100          blue <= (Others => '0');
101        end if;
102      end if;
103    end if;
104  end PROCESS;
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```

```
File Editor - C:/Users/Sergio/Desktop/ALU_Verilog/ALU_Verilog - [http://www.altera.com]
File Edit View Project Processing Tools Window Help Search altera.com

94      blue <= (Others == "0");
95    else
96      red <= (Others == "0");
97      green <= (Others == "0");
98      blue <= (Others == "0");
99    end if;
100  --c
101  elsif ((row>240 and row<320) AND (column==55 and column <70)) then
102    if(bus_A[4]=="0") then
103      red <= (Others == "1");
104      green <= (Others == "0");
105      blue <= (Others == "0");
106    else
107      red <= (Others == "0");
108      green <= (Others == "0");
109      blue <= (Others == "0");
110    end if;
111  --s
112  if((bus_A[3]=="0") then
113    red <= (Others == "1");
114    green <= (Others == "0");
115    blue <= (Others == "0");
116  else
117    red <= (Others == "0");
118    green <= (Others == "0");
119    blue <= (Others == "0");
120  end if;
121  --e
122  elsif ((row>220 and row<240) AND (column==70 and column <100)) then
123    if(bus_A[6]=="0") then
124      red <= (Others == "1");
125      green <= (Others == "0");
126      blue <= (Others == "0");
127    else
128      red <= (Others == "0");
129      green <= (Others == "0");
130      blue <= (Others == "0");
131    end if;
132  --t
133  --Gama B
134  --l
135  elsif ((l downto 0)-1 or sal(l downto 0)-"00" and sal(i)-"0") then
136    red <= (Others == "1");
137    green <= (Others == "0");
138    blue <= (Others == "0");
139  else
140    red <= (Others == "0");
141    green <= (Others == "0");
142    blue <= (Others == "0");
143  end if;
144  --f
145  elsif ((row>100 and row<110) AND (column==265 and column <270)) then
146    if((sal(i downto 0)-1 or sal(l downto 0)-"00" and sal(i)-"0") then
147      red <= (Others == "1");
148      green <= (Others == "0");
149      blue <= (Others == "0");
150    else
151      red <= (Others == "0");
152      green <= (Others == "0");
153      blue <= (Others == "0");
154    end if;
155  --lines negative B
156  elsif ((row>80 and row<100) AND (column==200 and column <250)) then
157    if(sal(i downto 0)-"0") then
158      red <= (Others == "1");
159      green <= (Others == "0");
160      blue <= (Others == "0");
161  else
162    red <= (Others == "0");
163    green <= (Others == "0");
164    blue <= (Others == "0");
165  end if;
166
167
```



```

118 and if:
119 --disp2
120 if (disp2[0]="0") then
121   red <= (Others <="1");
122   green <= (Others <="0");
123   blue <= (Others <="0");
124 else
125   red <= (Others <="0");
126   green <= (Others <="1");
127   blue <= (Others <="0");
128 end if;
129 --if
130 if (row <= 140 and row <= 220 ) AND (column <= 350 and column <= 395) then
131   if (disp2[1]="0") then
132     red <= (Others <="1");
133     green <= (Others <="0");
134     blue <= (Others <="0");
135   else
136     red <= (Others <="0");
137     green <= (Others <="1");
138     blue <= (Others <="0");
139   end if;
140 --if
141 if (row <= 240 and row <= 320 ) AND (column <= 380 and column <= 395) then
142   if (disp2[1]="0") then
143     red <= (Others <="1");
144     green <= (Others <="0");
145     blue <= (Others <="0");
146   else
147     red <= (Others <="0");
148     green <= (Others <="1");
149     blue <= (Others <="0");
150   end if;
151 --if
152 if (disp2[1]="1") then
153   red <= (Others <="1");
154   green <= (Others <="0");
155   blue <= (Others <="0");
156 else
157   red <= (Others <="0");
158   green <= (Others <="1");
159   blue <= (Others <="0");
160 end if;
161 --if
162 if (row <= 240 and row <= 320 ) AND (column <= 355 and column <= 355) then
163   if (disp2[1]="0") then
164     red <= (Others <="1");
165     green <= (Others <="0");
166     blue <= (Others <="0");
167   else
168     red <= (Others <="0");
169     green <= (Others <="1");
170     blue <= (Others <="0");
171   end if;
172 --if
173 if (disp2[1]="1") then
174   red <= (Others <="1");
175   green <= (Others <="0");
176   blue <= (Others <="0");
177 else
178   red <= (Others <="0");
179   green <= (Others <="1");
180   blue <= (Others <="0");
181 end if;
182 --if
183 if (row <= 220 and row <= 400 ) AND (column <= 350 and column <= 385) then
184   if (disp2[1]="0") then
185     red <= (Others <="1");
186     green <= (Others <="0");
187     blue <= (Others <="0");
188   else
189     red <= (Others <="0");
190     green <= (Others <="1");
191     blue <= (Others <="0");
192   end if;
193 --if
194 if (disp2[1]="1") then
195   red <= (Others <="1");
196   green <= (Others <="0");
197   blue <= (Others <="0");
198 else
199   red <= (Others <="0");
200   green <= (Others <="1");
201   blue <= (Others <="0");
202 end if;

```

```

199 --if
200 if (row <= 200 and row <= 220 ) AND (column <= 280 and column <= 315) then
201   red <= (Others <="1");
202   green <= (Others <="0");
203   blue <= (Others <="0");
204 else
205   red <= (Others <="0");
206   green <= (Others <="1");
207   blue <= (Others <="0");
208 end if;
209 --if
210 if (row <= 250 and row <= 400 ) AND (column <= 280 and column <= 315) then
211   red <= (Others <="1");
212   green <= (Others <="0");
213   blue <= (Others <="0");
214 else
215   red <= (Others <="0");
216   green <= (Others <="1");
217   blue <= (Others <="0");
218 end if;
219 --if
220 if (row <= 225 and row <= 315 ) AND (column <= 330 and column <= 365) then
221   red <= (Others <="1");
222   green <= (Others <="0");
223   blue <= (Others <="0");
224 else
225   red <= (Others <="0");
226   green <= (Others <="1");
227   blue <= (Others <="0");
228 end if;
229 --if
230 if (disp2[1]="0") then
231   red <= (Others <="1");
232   green <= (Others <="0");
233   blue <= (Others <="0");
234 else
235   red <= (Others <="0");
236   green <= (Others <="1");
237   blue <= (Others <="0");
238 end if;
239 --if
240 if (disp2[1]="1") then
241   red <= (Others <="1");
242   green <= (Others <="0");
243   blue <= (Others <="0");
244 else
245   red <= (Others <="0");
246   green <= (Others <="1");
247   blue <= (Others <="0");
248 end if;
249 --if
250 if (row <= 120 and row <= 400 ) AND (column <= 75 and column <= 95) then
251   if (disp2[1]="0") then
252     red <= (Others <="1");
253     green <= (Others <="0");
254     blue <= (Others <="0");
255   else
256     red <= (Others <="0");
257     green <= (Others <="1");
258     blue <= (Others <="0");
259   end if;
260 --if
261 if (row <= 120 and row <= 400 ) AND (column <= 100 and column <= 115) then
262   if (disp2[1]="0") then
263     red <= (Others <="1");
264     green <= (Others <="0");
265     blue <= (Others <="0");
266   else
267     red <= (Others <="0");
268     green <= (Others <="1");
269     blue <= (Others <="0");
270   end if;
271 --if
272 if (disp2[1]="1") then
273   red <= (Others <="1");
274   green <= (Others <="0");
275   blue <= (Others <="0");
276 else
277   red <= (Others <="0");
278   green <= (Others <="1");
279   blue <= (Others <="0");
280 end if;
281 --if
282 if (row <= 400 and row <= 400 ) AND (column <= 50 and column <= 115) then
283   if (disp2[1]="0") then
284     red <= (Others <="1");
285     green <= (Others <="0");
286     blue <= (Others <="0");
287   else
288     red <= (Others <="0");
289     green <= (Others <="1");
290     blue <= (Others <="0");
291   end if;
292 --if
293 if (disp2[1]="1") then
294   red <= (Others <="1");
295   green <= (Others <="0");
296   blue <= (Others <="0");
297 else
298   red <= (Others <="0");
299   green <= (Others <="1");
300   blue <= (Others <="0");
301 end if;

```



```

--2
elsif ((row > 240 and row < 320) AND (column < 530 and column < 545)) then
  if (disp2(2) = '1') then
    red <= (Others <= '1');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  else
    red <= (Others <= '0');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  end if;
--3
elsif ((row > 320 and row < 400) AND (column < 500 and column < 530)) then
  if (disp3(3) = '0') then
    red <= (Others <= '1');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  else
    red <= (Others <= '0');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  end if;
--4
elsif ((row > 400 and row < 480) AND (column < 485 and column < 500)) then
  if (disp4(4) = '0') then
    red <= (Others <= '1');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  else
    red <= (Others <= '0');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  end if;
--5
elsif ((row > 480 and row < 560) AND (column < 485 and column < 500)) then
  if (disp5(5) = '0') then
    red <= (Others <= '1');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  else
    red <= (Others <= '0');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  end if;
--6
elsif ((row > 560 and row < 640) AND (column < 50 and column < 530)) then
  if (disp6(6) = '0') then
    red <= (Others <= '1');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  else
    red <= (Others <= '0');
    green <= (Others <= '1');
    blue <= (Others <= '0');
  end if;
else
  red <= (Others <= '0');
  green <= (Others <= '0');
  blue <= (Others <= '0');
end if;
end if;
END PROCESS;
END behavior;

```

Código 13: Generador de imágenes

```

library ieee;
use ieee.std_logic_1164.all;

ENTITY TOP IS
  PORT (
    --a, b : in std_logic_vector(2 downto 0);
    reset : in std_logic;
    input_clk : in std_logic;
    red : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) := (OTHERS => '0'); --red magnitude output to DAC
    green : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) := (OTHERS => '0'); --green magnitude output to DAC
    blue : OUT STD_LOGIC_VECTOR(3 DOWNTO 0) := (OTHERS => '0');
    h_sync : OUT STD_LOGIC; --horizontal sync pulse
    v_sync : OUT STD_LOGIC; --vertical sync pulse
    salFinal : OUT std_logic_vector(3 downto 0)
  );
END TOP;

ARCHITECTURE arqTOP OF TOP IS
  signal pix_clock : std_logic;
  signal relop : std_logic;
  signal disp_ena : std_logic;
  signal column : INTEGER;
  signal row : INTEGER;
  signal a, b : std_logic_vector(2 downto 0);
  signal bus_A, bus_B : std_logic_vector(6 downto 0);
  signal sal : std_logic_vector(3 downto 0);
  signal disp0, disp1, disp2 : std_logic_vector(6 downto 0);
BEGIN
  u1 : entity work.genMhz(arqgenMhz) port map(input_clk, pix_clock);
  u2 : entity work.reloj(arqreloj) port map(input_clk, relop);
  cont : entity work.contador(arqcont) port map(relop, reset, sal);
  romData : entity work.romData(arqromData) port map("111", '1', a);
  romDatb : entity work.romData(arqromData) port map("011", '1', b);
  ALU : entity work.TOP_ALU(arq_alu) port map(a, b, sal(2 downto 0), sal(3), disp0, disp1, disp2, salFinal);
  valorA : entity work.rom(arqrom) port map(a, '1', bus_A);
  valorB : entity work.rom(arqrom) port map(b, '1', bus_B);
  --DISPLAY
  u3 : entity work.vga_controller(behavior) port map(pix_clock, '1', h_sync, v_sync, disp_ena, column, row);
  u4 : entity work.hw_image_generator(behavior) port map(disp_ena, row, column, red, green, blue, sal, a, b, bus_A, bus_B, disp0, disp1, disp2);
END ARCHITECTURE arqTOP;

```

Código 14: TOP proyecto

Pin Planner

Pin Planner - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA

File Edit View Processing Tools Window Help

Groups

Named: *

Node Name Direction

blue[3..0] Output Group

green[3..0] Output Group

red[3..0] Output Group

Groups Report

Tasks

Early Pin Planning

Run I/O Assignment Analysis

Top View - Wire Bond

MAX10 - 10M500AF48C7G

Pin Legend

Symbol Pin Type

User I/O

User assigned...

Filter assigne...

Unbonded pad

Reserved pin

Other configu...

DEV_OE

DEV_CLR

DIFF_n

DIFF_p

Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	Filter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Tri-Buffer	Preservation
blue[3]	Output	PIN_N2	2	B2_NO	PIN_N2	2.5 V		12mA (default)	2 (default)			
blue[2]	Output	PIN_P4	2	B2_NO	PIN_P4	2.5 V		12mA (default)	2 (default)			
blue[1]	Output	PIN_T1	2	B2_NO	PIN_T1	2.5 V		12mA (default)	2 (default)			
blue[0]	Output	PIN_P1	2	B2_NO	PIN_P1	2.5 V		12mA (default)	2 (default)			
green[3]	Output	PIN_R1	2	B2_NO	PIN_R1	2.5 V		12mA (default)	2 (default)			
green[2]	Output	PIN_R2	2	B2_NO	PIN_R2	2.5 V		12mA (default)	2 (default)			
green[1]	Output	PIN_T2	2	B2_NO	PIN_T2	2.5 V		12mA (default)	2 (default)			
green[0]	Output	PIN_W1	2	B2_NO	PIN_W1	2.5 V		12mA (default)	2 (default)			
h_sync	Output	PIN_N3	2	B2_NO	PIN_N3	2.5 V		12mA (default)	2 (default)			
input_clk	Input	PIN_P11	3	B3_NO	PIN_P11	2.5 V		12mA (default)	2 (default)			
red[3]	Output	PIN_Y1	3	B3_NO	PIN_Y1	2.5 V		12mA (default)	2 (default)			
red[2]	Output	PIN_Y2	3	B3_NO	PIN_Y2	2.5 V		12mA (default)	2 (default)			
red[1]	Output	PIN_V1	2	B2_NO	PIN_V1	2.5 V		12mA (default)	2 (default)			
red[0]	Output	PIN_AA1	3	B3_NO	PIN_AA1	2.5 V		12mA (default)	2 (default)			

0% 00:00:00

Pin Planner - C:/Users/Sergio/Desktop/VLSI/Práctica5/ALU_VGA/ALU_VGA - ALU_VGA

File Edit View Processing Tools Window Help

Groups

Named: *

Node Name Direction

blue[3..0] Output Group

green[3..0] Output Group

red[3..0] Output Group

Groups Report

Tasks

Early Pin Planning

Run I/O Assignment Analysis

Top View - Wire Bond

MAX10 - 10M500AF48C7G

Pin Legend

Symbol Pin Type

User I/O

User assigned...

Filter assigne...

Unbonded pad

Reserved pin

Other configu...

DEV_OE

DEV_CLR

DIFF_n

DIFF_p

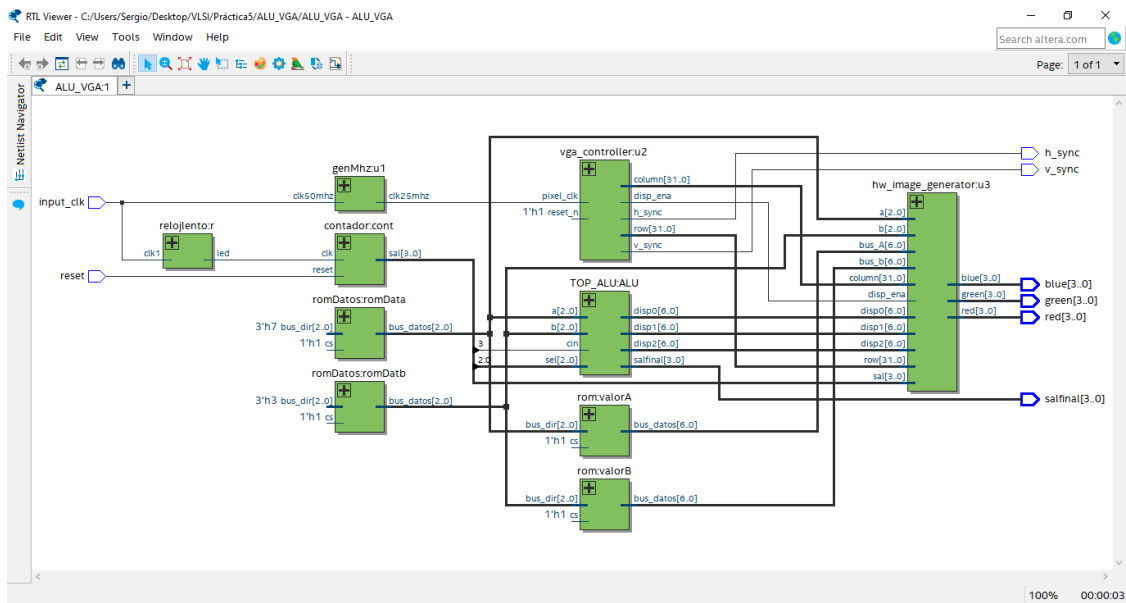
Filter: Pins: all

Node Name	Direction	Location	I/O Bank	VREF Group	Filter Location	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Tri-Buffer	Preservation
h_sync	Output	PIN_N3	2	B2_NO	PIN_N3	2.5 V		12mA (default)	2 (default)			
input_clk	Input	PIN_P11	3	B3_NO	PIN_P11	2.5 V		12mA (default)	2 (default)			
red[3]	Output	PIN_Y1	3	B3_NO	PIN_Y1	2.5 V		12mA (default)	2 (default)			
red[2]	Output	PIN_Y2	3	B3_NO	PIN_Y2	2.5 V		12mA (default)	2 (default)			
red[1]	Output	PIN_V1	2	B2_NO	PIN_V1	2.5 V		12mA (default)	2 (default)			
red[0]	Output	PIN_AA1	3	B3_NO	PIN_AA1	2.5 V		12mA (default)	2 (default)			
reset	Input	PIN_F15	7	B7_NO	PIN_F15	2.5 V		12mA (default)	2 (default)			
salfinal[3]	Output	PIN_B10	7	B7_NO	PIN_B10	2.5 V		12mA (default)	2 (default)			
salfinal[2]	Output	PIN_A10	7	B7_NO	PIN_A10	2.5 V		12mA (default)	2 (default)			
salfinal[1]	Output	PIN_A9	7	B7_NO	PIN_A9	2.5 V		12mA (default)	2 (default)			
salfinal[0]	Output	PIN_A8	7	B7_NO	PIN_A8	2.5 V		12mA (default)	2 (default)			
v_sync	Output	PIN_N1	2	B2_NO	PIN_N1	2.5 V		12mA (default)	2 (default)			
<-new node-->												

0% 00:00:00

Pin Planner 1: ALU Automatizada




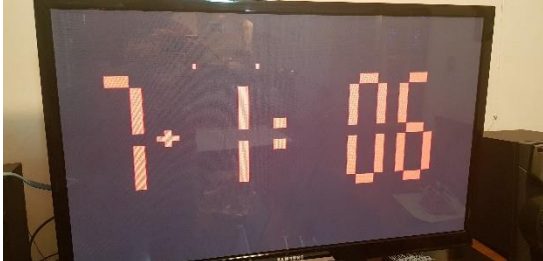

RTL

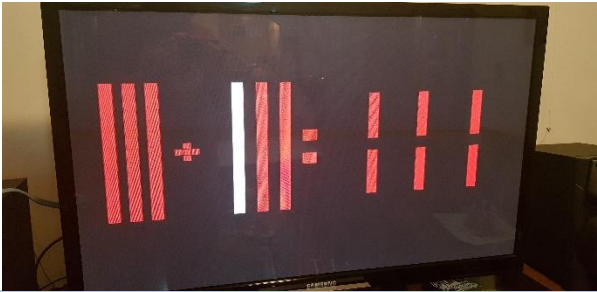
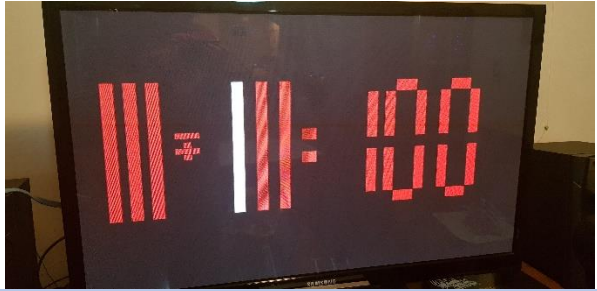

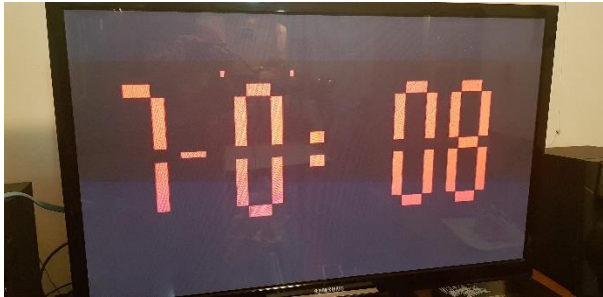
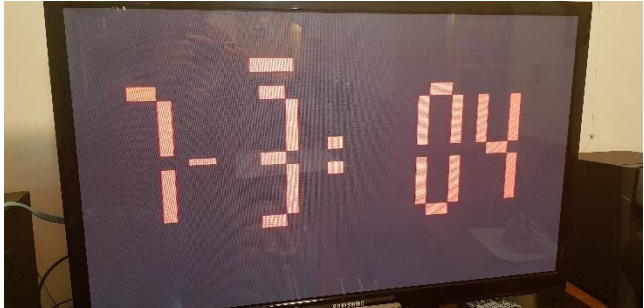


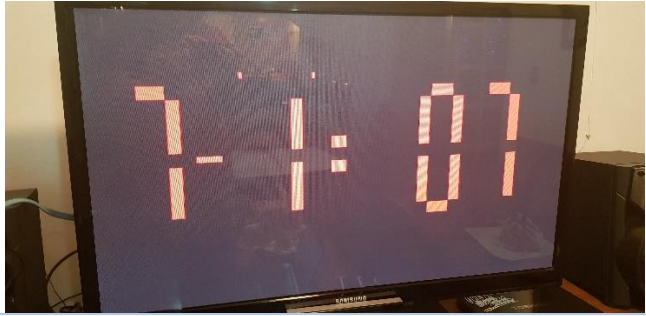
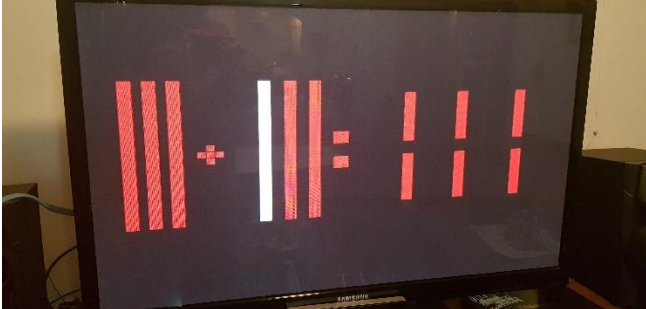
RTL 1: ALU automatizada

Prueba

Para el manejo de estas pruebas se utilizaron los números guardados $A=7$ (111) y $B=3$ (011).

Estado	Operación	Prueba
0000	$\text{suma}(A + '0')$	
0001	$\text{suma}(A + \bar{B})$	
0010	$\text{suma}(A + B)$	
0011	$\text{suma}(A + '1')$	
0100	AND	

0101	<i>OR</i>	
0110	<i>XOR</i>	
0111	<i>NOT</i>	
1000	<i>resta(A - 0')</i>	
1001	<i>resta(A - B̄)</i>	

1010	$resta(A - B)$	
1011	$resta(A - '1')$	
1100	AND	
1101	OR	

1110	<i>XOR</i>	
1111	<i>NOT</i>	

Video de prueba: