Instituto Tecnológico de Las Américas

Departamento de Mecatrónica

Informe de Prácticas:

Título de Practica:

Circuitos secuencial

Estudiante:

Nelson J. Ramírez Mordan

Matricula:

2021-0360

Asignatura:

Sistemas Digitales

Docente:

Obed Hernández Castillo

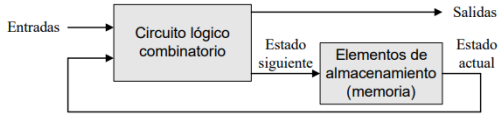
Fecha:

23/03/2023

Indice

Introducción

5.1 Introducción a los sistemas secuenciales. Hasta ahora, los circuitos lógicos que se han considerado han sido combinatorios. En estos las salidas en cualquier punto del tiempo dependen completamente de las entradas que se presenten en ese momento. Aunque los circuitos combinatorios son la base para un gran número de aplicaciones, en la práctica la mayoría de los sistemas también incluyen elementos de almacenamiento, por lo que su análisis y diseño se debe realizar en términos de circuitos secuenciales [4]. Un circuito lógico secuencial es aquel cuyas salidas no solo dependen de sus entradas actuales, sino también de su posición o estado actual, almacenada en elementos de memoria [3]. En la siguiente figura se presenta un diagrama a bloques de un circuito secuencial. Este consta de un circuito combinatorio y elementos de almacenamiento que juntos forman un sistema retroalimentado. Los elementos de almacenamiento son dispositivos que pueden almacenar información binaria en su interior (1's y 0's). La información binaria almacenada define el estado del circuito secuencial [4]. El circuito secuencial recibe información binaria de entradas externas, las cuales, junto con el estado presente almacenado en memoria, determinan el valor binario de las salidas, así como la condición para cambiar el estado del circuito.

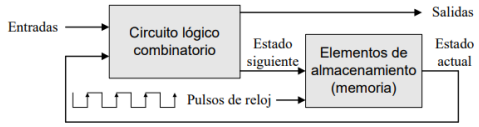


El diagrama a bloques muestra que las salidas de un circuito secuencial son función no solo de entradas externas sino también del estado actual o presente de los elementos de almacenamiento, cuyo estado siguiente o próximo, es así mismo, función de las entradas y del estado presente. Por lo tanto, un circuito secuencial se especifica por medio de una secuencia temporal de entradas, salidas y estados internos.

5.1.1 Tipos de circuitos secuenciales.

Existen dos tipos principales de circuitos secuenciales, los asíncronos y los síncronos. Circuito lógico combinatorio Entradas Salidas Elementos de almacenamiento (memoria) Estado siguiente Estado actual Un circuito secuencial asíncrono es aquel que su estado puede ser afectado en cualquier

instante al cambiar el valor de las entradas. Sus elementos de almacenamiento son dispositivos con retraso de tiempo, en los cuales la capacidad de almacenamiento se debe a que la señal tarda un tiempo finito en propagarse por el dispositivo. En los sistemas asíncronos del tipo compuertas, los elementos de almacenamiento constan de compuertas lógicas donde el tiempo de propagación de las señales proporciona el espacio de almacenamiento requerido. Por lo tanto, un circuito secuencial asíncrono puede considerarse como un circuito combinacional con retroalimentación. Debido a la retroalimentación entre compuertas lógicas, el sistema puede operar de manera impredecible y algunas veces incluso hacerse inestable, por lo que se utilizan en muy contadas ocasiones. Un circuito secuencial síncrono utiliza señales que modifican su estado solo en instantes discretos de tiempo. La sincronización se logra a través de un dispositivo de sincronización llamado generador de señales de reloj que produce una sucesión periódica de pulsos de reloj. Estos se distribuyen en todo el sistema de tal manera que los elementos de almacenamiento sólo sean afectados a la llegada de cada pulso:



Circuitos secuencial

Recrear el funcionamiento del Juego de google del dinosaurio, basado en sus estados.
Los display mostrarán el estado en el que se encuentra el dinosaurio. Debe tener un contador

Requiere por cada circuito.

1. Tabla de verdad
2. Diagrama de estados FSM
3. Diseño del circuito
4. Explicación del circuito y sus diferentes partes (combinacional y secuencial)
5. RTL
6. Verilog
7. Constraint file
8. Foto física de la implementación.

Desarrollo

Breve explicación

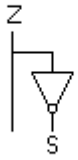
Para esta practica se utilizo un diagrama de estado costruido por mi, por el cual se hizo la tabla y se sacaron las ecuaciones para los Flip Flop.

Tabla de los estados:

PASADO		SALTAR	AGA	FUTURO		FLIP-FLOP		
A	B	C	D	A	B	DA	DB	
0	0	0	0	1	1	1	1	0,
0	0	0	1	0	1	0	1	1,
0	0	1	0	1	0	1	0	2,
0	1	0	0	1	1	1	1	4,
0	1	0	1	0	1	0	1	5,
0	1	1	0	1	0	1	0	6,
1	0	0	0	1	1	1	1	8,
1	0	0	1	0	1	0	1	9,
1	0	1	0	1	0	1	0	10,
1	1	0	1	0	0	0	0	13,

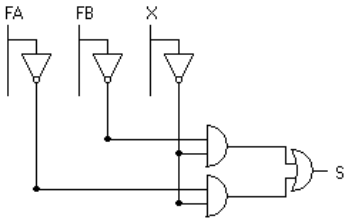
Ecuaciones dadas:

DA: (~Z)
DB: (~FB*~X)+(~FA*~X)



DA:

DB:



Ecuaciones display 7 segmentos

A	B	C	D		A	B	C	D	E	F	G	
0	0	0	0	S	0	1	0	0	1	0	0	0.
0	0	0	1	T	1	1	1	0	0	0	0	1.
0	0	1	0	P	0	0	1	1	0	0	0	2.
0	0	1	1	J	1	0	0	0	0	1	1	3.
0	1	0	0	U	1	1	0	0	0	1	1	4.
0	1	0	1	M	0	0	0	1	0	0	1	5.
0	1	1	0	L	1	1	1	0	0	0	1	6.
0	1	1	1	O	0	0	0	0	0	0	1	7.
1	0	0	0	W	0	1	0	0	0	1	1	8.
1	0	0	1	E	0	1	1	0	0	0	0	9.
1	0	1	0	N	1	1	0	1	0	1	0	10.
1	0	1	1	D	1	0	0	0	0	1	0	11.
1	1	0	0		1	1	1	1	1	1	1	12.
1	1	0	1		1	1	1	1	1	1	1	13.
1	1	1	0		1	1	1	1	1	1	1	14.
1	1	1	1		1	1	1	1	1	1	1	15.

A = (~A & ~B & D)|(B & ~D)|(A & C)|(A & B);

B = (~B & ~C)|(B & ~D)|(A & ~D)|(A & B);

C = (~B & ~C & D)|(~A & C & ~D)|(A & B);

D = (~B & C & ~D)|(B & ~C & D)|(A & B);

E = (~A & ~B & ~C & ~D)|(A & B);

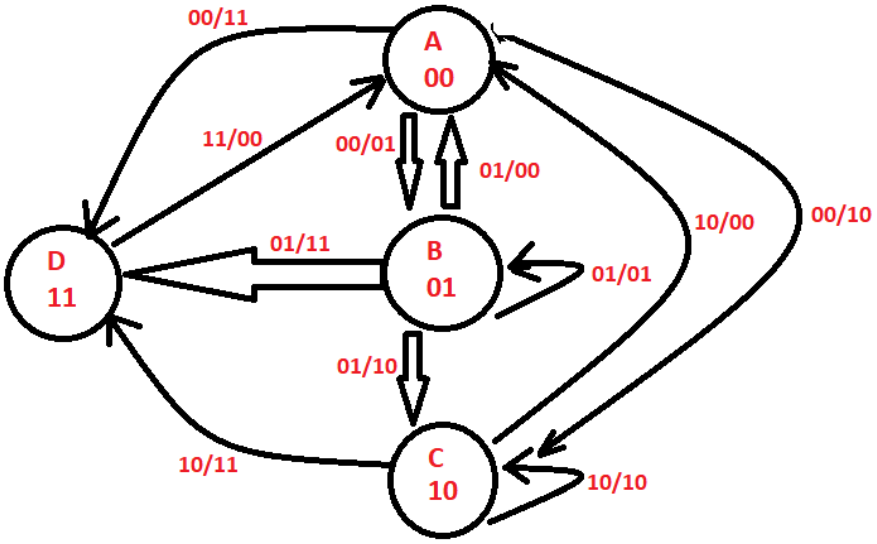
F = (B & ~C & ~D)|(~B & C & D)|(A & ~D)|(A & B);

G = (~A & C & D)|(B)|(A & ~C & ~D);

Anexos

Diagrama de estado

00 = STOP / COMENZAR
01 = SALTAR
10 = AGACHARSE
11 = MUERTO



Codigo:

```
1  `timescale 1ns / 1ps
2
3  module Dinosaurio(
4
5     // FLIP FLOP 1
6     input reset_n, // RESET
7     output reg FA, // SALIDA FLIP FLOP
8     output reg FA_n, //  SALIDA FLIP FLOP
9
10    // FLIP FLOP 2
11    output reg FB, // SALIDA FLIP FLOP
12    output reg FB_n, //  SALIDA FLIP FLOP
13
14    // VARIABLES DE LOS LED
15    output iGameOver,
16    output iComenzar,
17    output iSaltar,
18    output iAgacharse,
19    output iCorriendo,
20
21    // DEMAS VARIABLES
22    input clk_in, // ENTRADA CLK 100MHZ
23    output reg clk_out, // SALIDA CLK 5HZ
24    input Z, // AGACHARSE
25    input X, // SALTAR
26
27    // DISPLAY
28    output [6:0] out, // Salida
29    output [6:0] out2, // Salida
30    output [6:0] out3, // Salida
31    output reg [11:0] iSelect, // Salida de selección
32    output reg [2:0] iBase_Segmentos, // Salida de base
33    output reg [6:0] iSegmentos_Display // Salida de segmentos
34
35 );
```

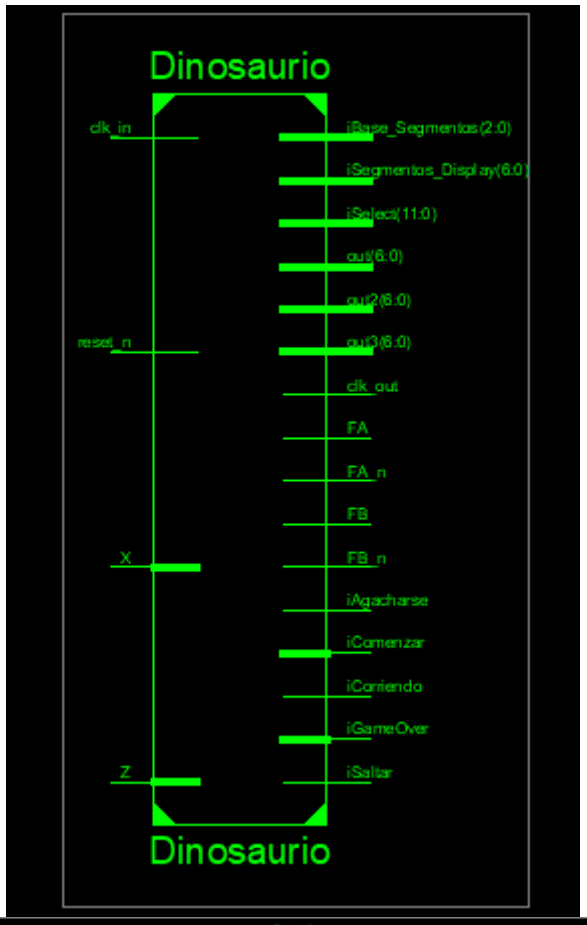
```
37  ////////////////////////////////////////////////////////////////////////
38
39  ///////////////     DIVISOR DE FRECUENCIA PARA EL CLK    ///////////////
40
41  ////////////////////////////////////////////////////////////////////////
42
43  reg [26:0] count; // Contador de 27 bits (2^27 = 134217728 > 1000000/5)
44
45  always @(posedge clk_in) begin
46    if (count == 100000) begin // Si se han contado suficientes ciclos
47      count <= 0; // Reiniciar el contador
48      clk_out <= ~clk_out; // Invertir el valor del clk de salida
49    end else begin
50      count <= count + 1; // Incrementar el contador
51    end
52  end
53
54  ////////////////////////////////////////////////////////////////////////
55
56  ///////////////     IMPLEMENTACION DE LOS 2 FLIP FLOP    ///////////////
57
58  ////////////////////////////////////////////////////////////////////////
59
60  always @(posedge clk_out or negedge reset_n) begin
61    if (~reset_n) begin
62      FA <= 1'b0;
63      FA_n <= 1'b1;
64      FB <= 1'b0;
65      FB_n <= 1'b1;
66    end else begin
67      FA <= (~Z);
68      FA_n <= Z;
69      FB <= (FB_n*~X)|(FA_n*~X);
70      FB_n <= ~(FA_n & ~X) & ~(FB_n & ~X);
71    end

74  ////////////////////////////////////////////////////////////////////////
75
76  ///////////////     LOGICA COMBINACIONAL PARA LOS LED    ///////////////
77
78  ////////////////////////////////////////////////////////////////////////
79
80  assign iGameOver = FA & FB;
81  assign iSaltar = FB & FA_n & Z;
82  assign iAgacharse = FB_n & FA & X;
83  assign iComenzar = ~(iGameOver | iSaltar | iAgacharse);
84  assign iCorriendo = ~(iGameOver ^ iComenzar ^ Z ^ X);
85
86  //assign ~( A ^ B ) = iComenzar; // Donde ^ significa que es una XOR
87
    ////////////////////////////////////////////////////////////////////////
```

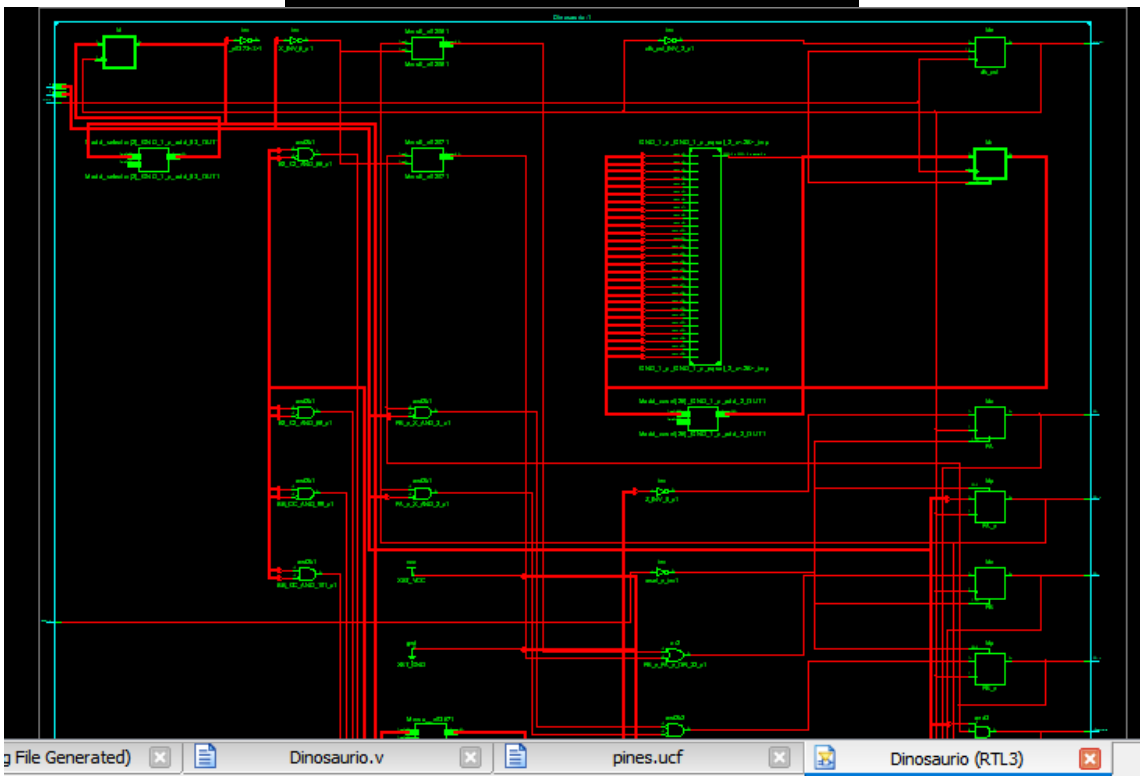
```
88  ////////////////////////////////////////////////////////////////////////
89
90  ///////////////        DECLARACIONES DE LOS DISPLAY      ///////////////
91
92  ////////////////////////////////////////////////////////////////////////
93
94  ////////////          DISPLAY 1          ////////////
95
96  wire A,B,C,D;
97  assign A = iSelect[3];
98  assign B = iSelect[2];
99  assign C = iSelect[1];
100 assign D = iSelect[0];
101
102 assign out[0] = (~A & ~B & D)|(B & ~D)|(A & C)|(A & B);
103 assign out[1] = (~B & ~C)|(B & ~D)|(A & ~D)|(A & B);
104 assign out[2] = (~B & ~C & D)|(~A & C & ~D)|(A & B);
105 assign out[3] = (~B & C & ~D)|(B & ~C & D)|(A & B);
106 assign out[4] = (~A & ~B & ~C & ~D)|(A & B);
107 assign out[5] = (B & ~C & ~D)|(~B & C & D)|(A & ~D)|(A & B);
108 assign out[6] = (~A & C & D)|(B)|(A & ~C & ~D);
109
110
111 ////////////          DISPLAY 2          ////////////
112
113 wire A2,B2,C2,D2;
114 assign A2 = iSelect[7];
115 assign B2 = iSelect[6];
116 assign C2 = iSelect[5];
117 assign D2 = iSelect[4];
118
119 assign out2[0] = (~A2 & ~B2 & D2 )+(B2 & ~D2 )+(A2 & C2)+(A2 & B2);
120 assign out2[1] = (~B2 & ~C2)+(B2 & ~D2)+(A2 & ~D2)+(A2 & B2);
121 assign out2[2] = (~B2 & ~C2 & D2)+(~A2 & C2 & ~D2)+( A2 & B2);
122 assign out2[3] = (~B2 & C2 & ~D2 )+(B2 & ~C2 & D2)+(A2 & B2);

122 assign out2[3] = (~B2 & C2 & ~D2 )+(B2 & ~C2 & D2)+(A2 & B2);
123 assign out2[4] = (~A2 & ~B2 & ~C2 & ~D2)+(A2 & B2);
124 assign out2[5] = (B2 & ~C2 & ~D2)+(~B2 & C2 & D2)+(A2 & ~D2)+(A2 & B2);
125 assign out2[6] = (~A2 & C2 & D2)+(B2)+(A2 & ~C2 & ~D2);
126
127 ////////////          DISPLAY 3          ////////////
128
129 wire AA,BB,CC,DD;
130 assign AA = iSelect[11];
131 assign BB = iSelect[10];
132 assign CC = iSelect[9];
133 assign DD = iSelect[8];
134
135 assign out3[0] = (~AA & ~BB & DD)+(BB & ~DD)+(AA & CC)+(AA & BB);
136 assign out3[1] = (~BB & ~CC)+(BB & ~DD)+(AA & ~DD)+(AA & BB);
137 assign out3[2] = (~BB & ~CC & DD)+(~AA & CC & ~DD)+(AA & BB);
138 assign out3[3] = (~BB & CC & ~DD)+(BB & ~CC & DD)+(AA & BB);
139 assign out3[4] = (~AA & ~BB & ~CC & ~DD)+(AA & BB);
140 assign out3[5] = (BB & ~CC & ~DD)+(~BB & CC & DD)+(AA & ~DD)+(AA & BB);
141 assign out3[6] = (~AA & CC & DD)+(BB)+(AA & ~CC & ~DD);
142
143 ////////////////////////////////////////////////////////////////////////
```

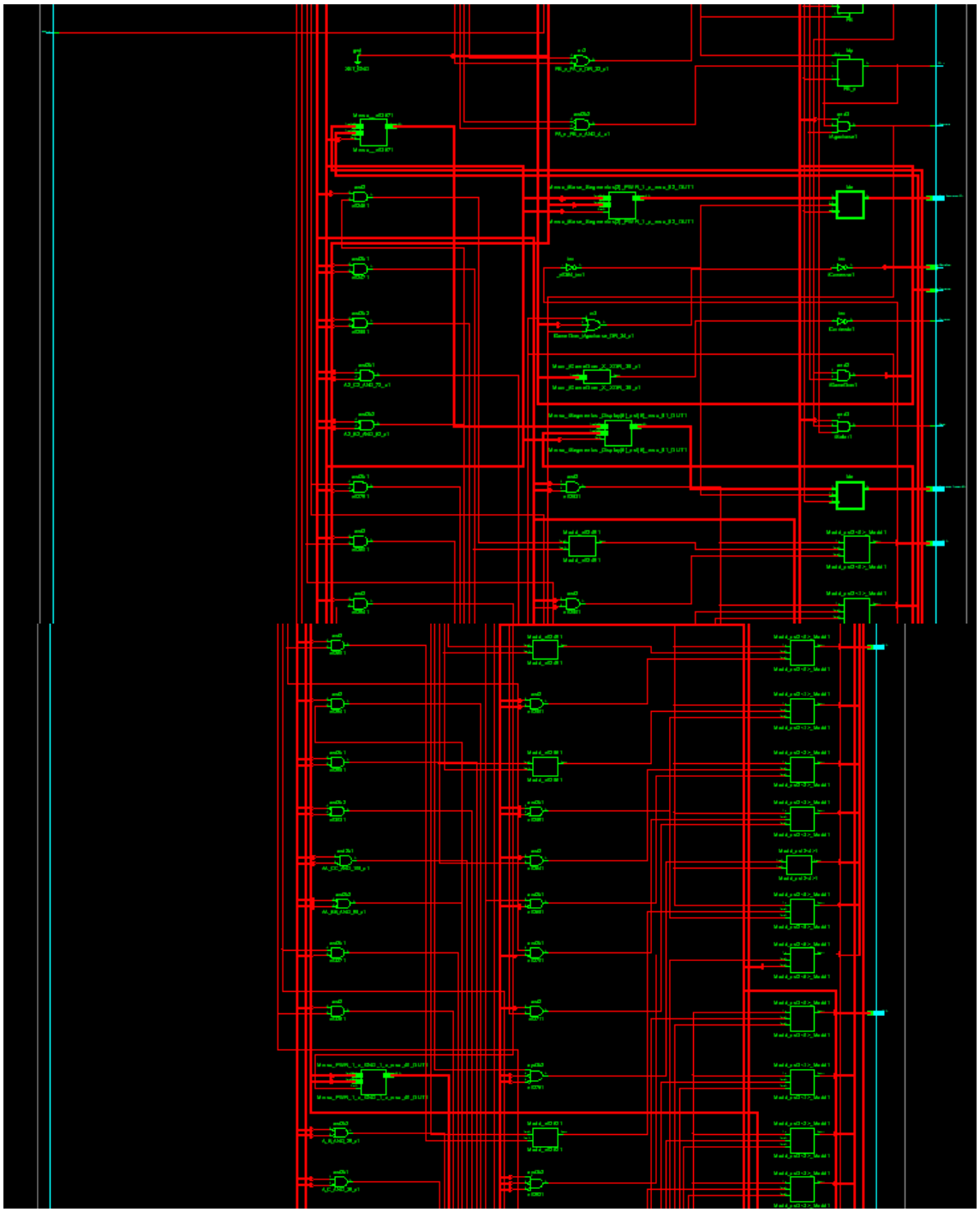
```
143 ////////////////////////////////////////////////////////////////////////
144
145 ///////////////          SALIDAS DE LOS DISPLAY          ///////////////
146
147 ////////////////////////////////////////////////////////////////////////
148
149 always @(*) begin
150
151
152       /*
153       0  0  0  0  S
154       0  0  0  1  T
155       0  0  1  0  P
156       0  0  1  1  J
157       0  1  0  0  U
158       0  1  0  1  M
159       0  1  1  0  L
160       0  1  1  1  O
161       1  0  0  0  W
162       1  0  0  1  E
163       1  0  1  0  N
164       1  0  1  1  D
165
166       */
167
168       // AQUI APARECERA STP = STOP
169       if (iComenzar)
170          iSelect = 12'b000000010010;
171
172       // AQUI APARECERA JMP = JUMP
173       else if (iSaltar)
174          iSelect = 12'b001101000101;
175
176       // AQUI APARECERA LOW = AGACHARSE
177       else if (iAgacharse)
178          iSelect = 12'b011001111000;
179
180       //AQUI APARECERA END = GAME OVER
181       else if (iGameOver)
182          iSelect = 12'b100110101011;
183       else
184          iSelect = 12'b111111111111; // Valor por defecto
185 end
186
```

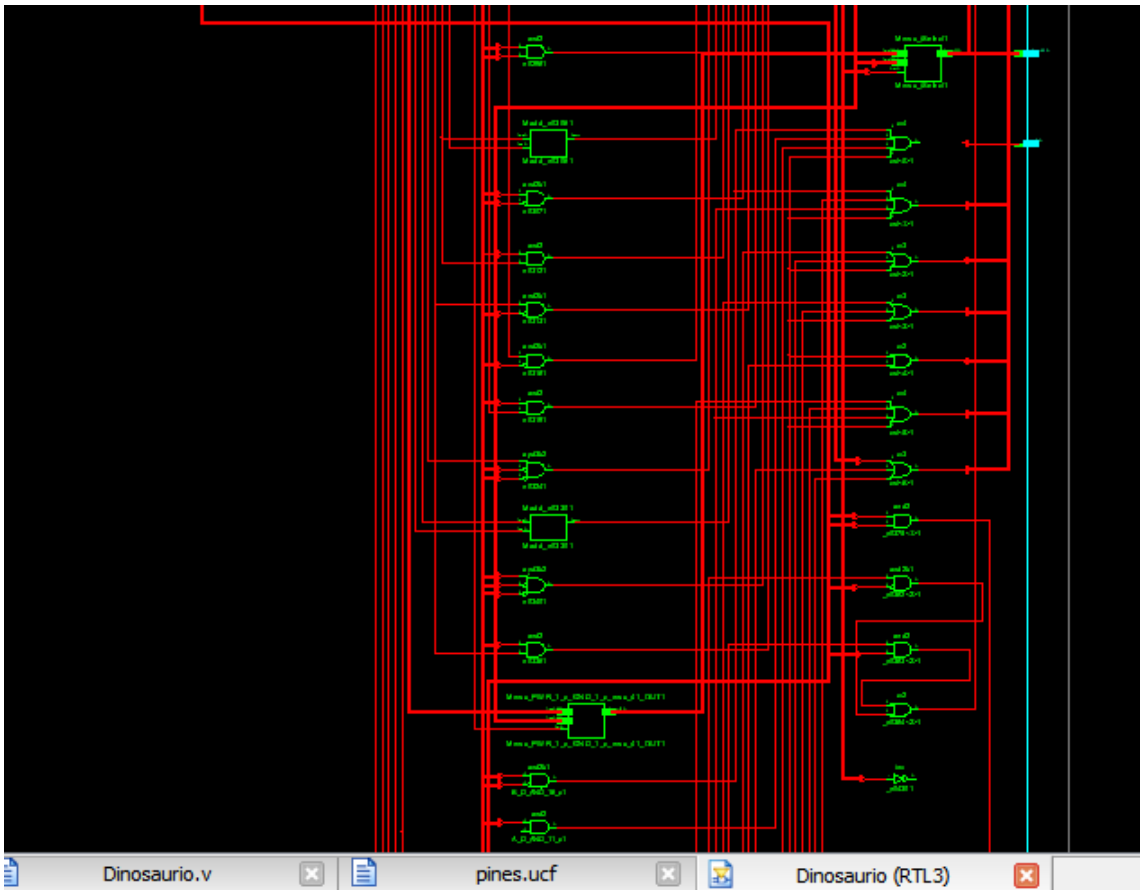
```
187 ////////////////////////////////////////////////////////////////////////
188
189 //////////////     SELECTOR PARA EL CAMBIO DE LAS BASES     //////////////
190
191 ////////////////////////////////////////////////////////////////////////
192
193 reg [2:0] selector;
194       always @ (posedge clk_out)
195
196    begin
197    if  (selector == 3'b000)
198    begin iSegmentos_Display = out; iBase_Segmentos = 3'b110; end
199
200    else if (selector == 3'b010)
201    begin iSegmentos_Display = out2; iBase_Segmentos = 3'b101; end
202
203    else if (selector == 3'b100)
204    begin iSegmentos_Display = out3; iBase_Segmentos = 3'b011; end
205
206    selector = selector + 1'b1; // Incrementa el selector
207
208 end
209
210 endmodule
211
```

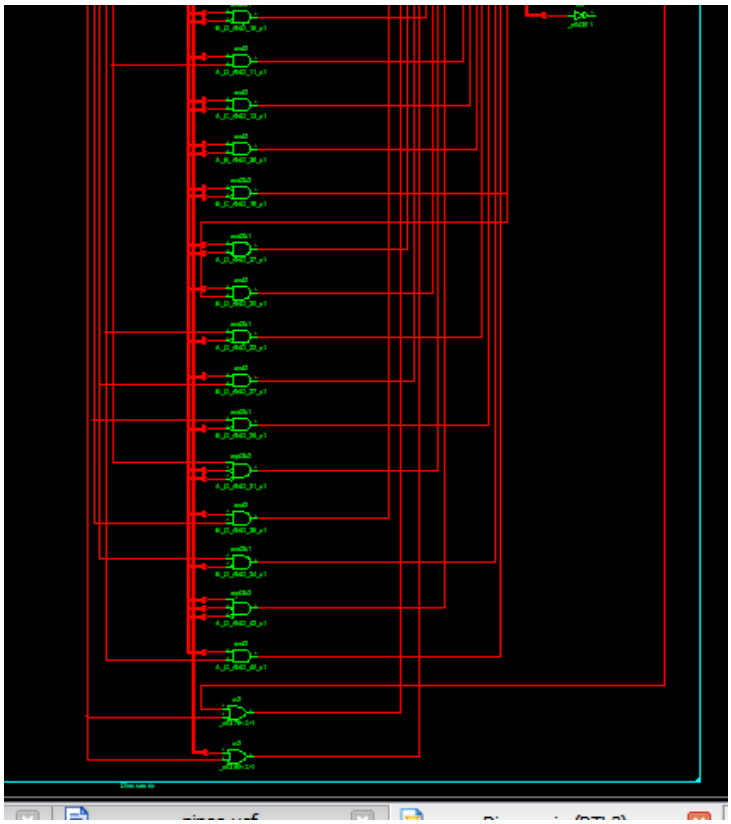
RTL


Dinosaurio
clk_in
reset_n
X
Z
iBase_Segmentos(2:0)
iSegmentos_Display(6:0)
iSelect(11:0)
out(6:0)
out2(6:0)
out3(6:0)
clk_out
FA
FA_n
FB
FB_n
iAgacharse
iComenzar
iCorriendo
iGameOver
iSaltar
Dinosaurio


File Generated)
Dinosaurio.v
pines.ucf
Dinosaurio (RTL3)




Dinosaurio.v
pines.ucf
Dinosaurio (RTL3)



UCR

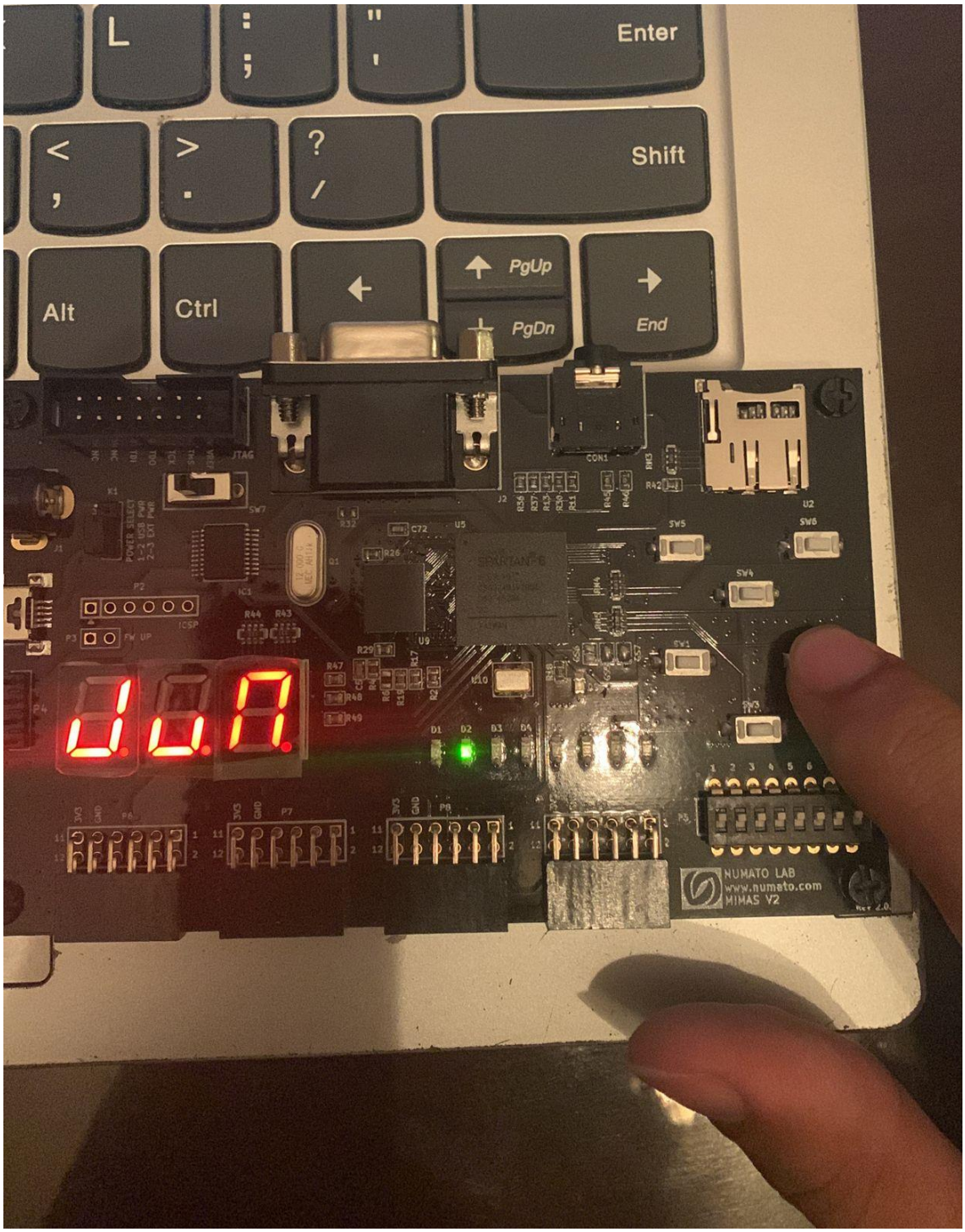
```
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#
# This file is a .ucf for Mimas V2 (http://www.numato.com)                                                                    #
# To use it in your project :                                                                                                 #
# * Remove or comment the lines corresponding to unused pins in the project.                                                  #
# * Rename the used signals according to the your project.                                                                    #
# * For more detail refer the User Guide for Mimas V2 available at http://numato.com/fpga-cpld                               #
#+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++#

#*****************************************************************************************************************************
#                                                 UCF for Mimas V2
#*****************************************************************************************************************************

CONFIG VCCAUX = "3.3" ;

   NET "clk_in"                 LOC = V10     | IOSTANDARD = LVCMOS33 | PERIOD = 100MHz ;
   #NET "CLK_12MHz"                 LOC = D9      | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz ;

#############################################################################################################################
#                                                 UART Interface
#############################################################################################################################
    #NET "UART_TX"                   LOC = A8      |  IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
    #NET "UART_RX"                   LOC = B8      |  IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

#############################################################################################################################
#                                                 SPI Flash
#############################################################################################################################
    #NET "SDI"                       LOC = T13     | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;  #MOSI
    #NET "SDO"                       LOC = R13     | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;  #MISO
    #NET "SCLK"                      LOC = R15     | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;  #SCK
    #NET "CS"                        LOC = V3      | IOSTANDARD = LVCMOS33 | SLEW = FAST | DRIVE = 8 ;  #CS

#############################################################################################################################
#                                                LPDDR MT46H32M16XXXX-5
#############################################################################################################################
    #NET  "calib_done"               LOC = P15     | IOSTANDARD = LVCMOS33;
    #NET  "error"                    LOC = P16     | IOSTANDARD = LVCMOS33;
    #NET  "c3_sys_rst_n"             LOC = L15     | IOSTANDARD = LVCMOS33 | PULLDOWN;    # Pin 7 of Header P9
```

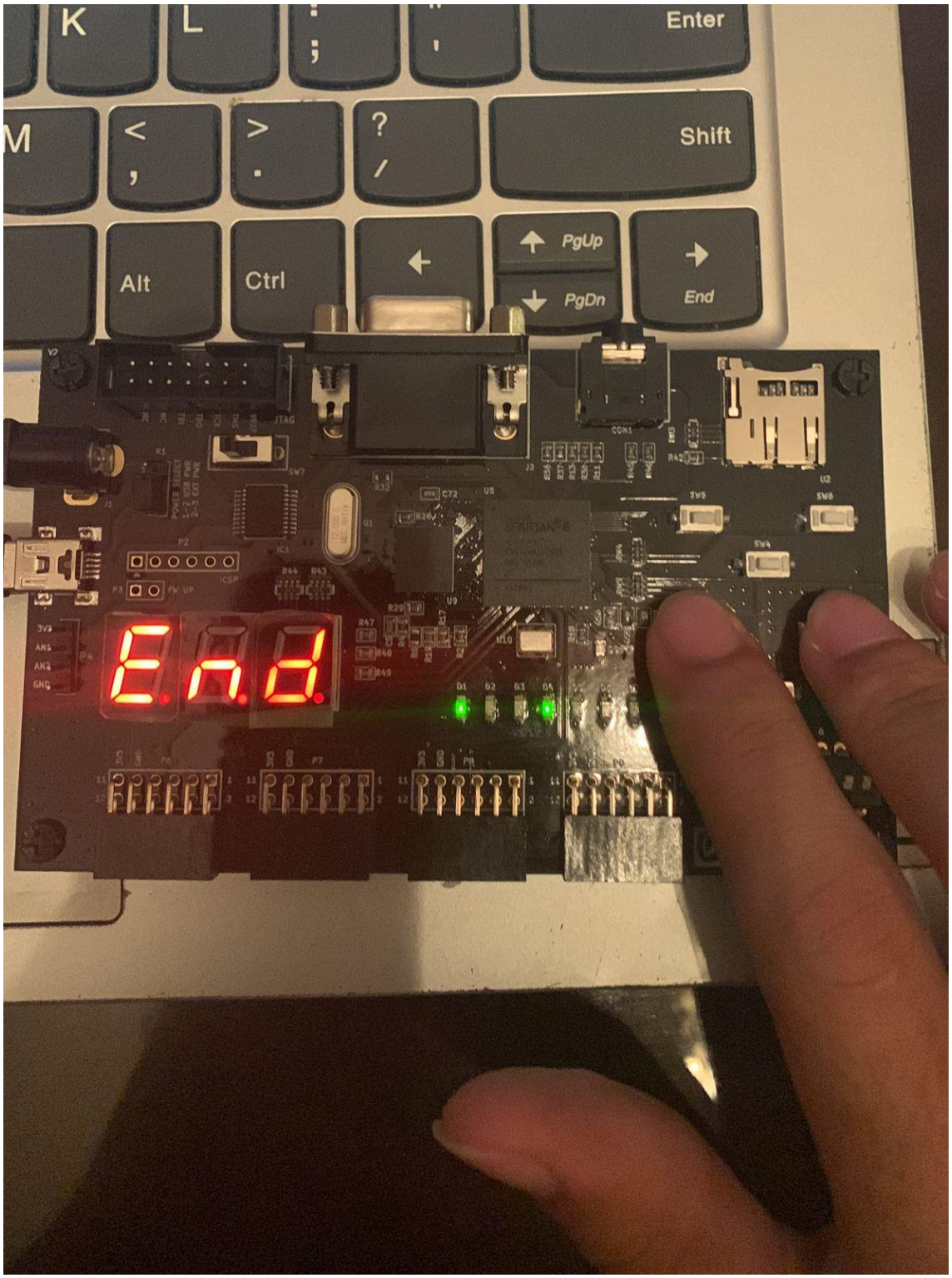
```
#############################################################################################################################
#                                          Push Buttons Switches
#############################################################################################################################
       //NET "iBoton_Saltar" CLOCK_DEDICATED_ROUTE = TRUE;

    NET "Z"                 LOC = M18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP; #SW1
    NET "X"                 LOC = L18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP; #SW2
    NET "reset_n"                LOC = M16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP; #SW3
    #NET "Switch[2]"                LOC = L17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP; #SW4
    #NET "Switch[1]"                LOC = K17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP; #SW5
    #NET "Switch[0]"                LOC = K18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST | PULLUP; #SW6

#############################################################################################################################
#                                                 LEDs
#############################################################################################################################

    NET "iGameOver"                   LOC = P15     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D1
    NET "iSaltar"                   LOC = P16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D2
    NET "iAgacharse"                  LOC = N15     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D3
    NET "iComenzar"                 LOC = N16     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D4
    NET "iCorriendo"                  LOC = U17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D5
    #NET "LED[2]"                   LOC = U18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D6
    #NET "LED[1]"                   LOC = T17     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D7
    #NET "LED[0]"                   LOC = T18     | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;  #D8

#############################################################################################################################
#                                                 Micro SD Card
```

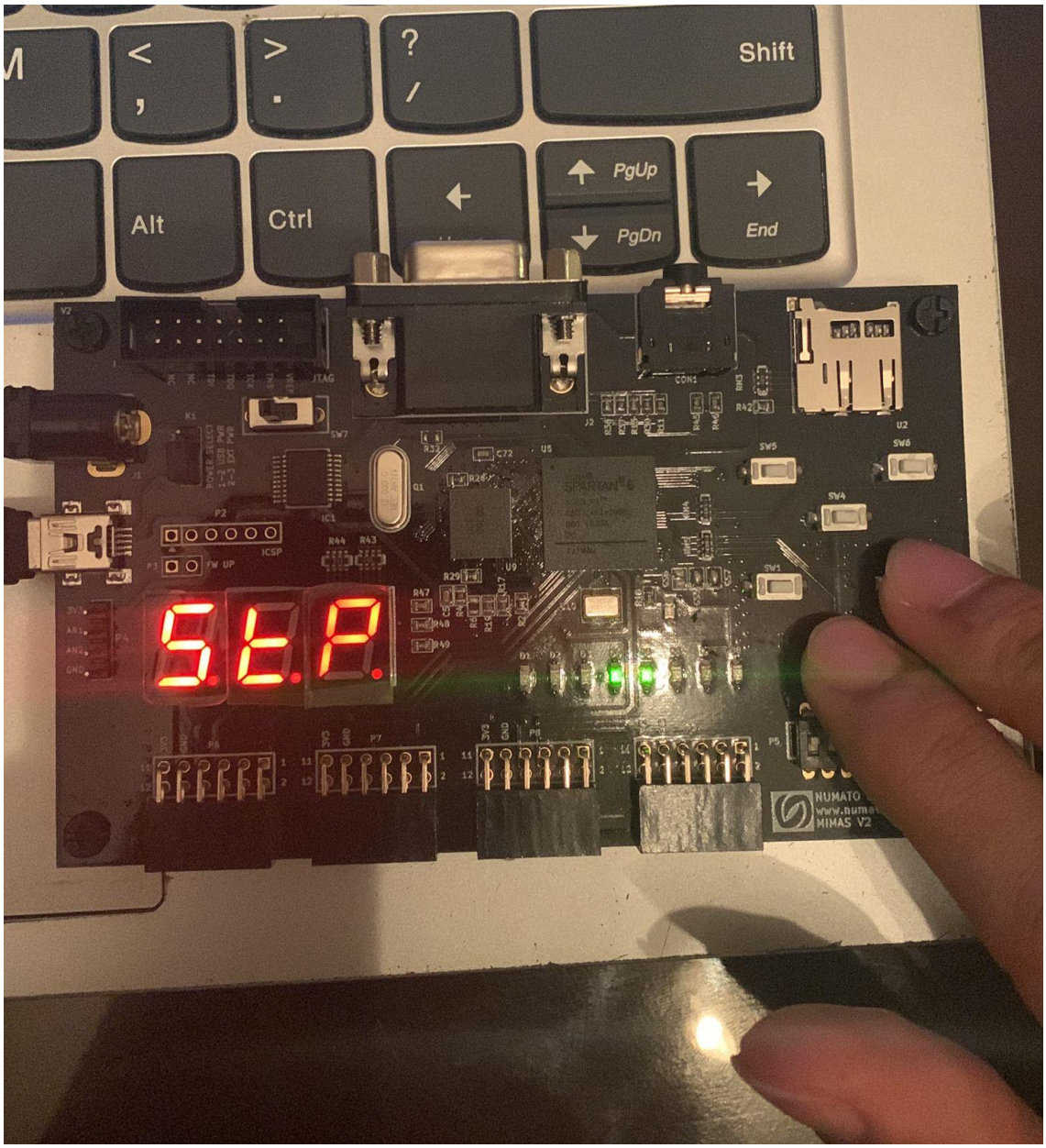
```
#############################################################################################################################
#                                          Seven Segment Display
#############################################################################################################################

    NET "iSegmentos_Display[0]"           LOC = A3      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #a
    NET "iSegmentos_Display[1]"           LOC = B4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #b
    NET "iSegmentos_Display[2]"           LOC = A4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #c
    NET "iSegmentos_Display[3]"           LOC = C4      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #d
    NET "iSegmentos_Display[4]"           LOC = C5      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #e
    NET "iSegmentos_Display[5]"           LOC = D6      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #f
    NET "iSegmentos_Display[6]"           LOC = C6      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #g
    //NET "iSegmentos_Display[0]"           LOC = A5      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #dot

    NET "iBase_Segmentos[2]"      LOC = B3      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;   #Enables for Seven Se
    NET "iBase_Segmentos[1]"      LOC = A2      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
    NET "iBase_Segmentos[0]"      LOC = B2      | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;

#############################################################################################################################
#                                                 Audio
#############################################################################################################################
```

Fotos en físico, estados:


F9
F10
F11
F12
PrtSc
Ins
Delete
Backspace
Enter
Shift
Alt
Ctrl
PgUp
PgDn
End
5t.P.
HUMATO LAB
www.humato.com
MIMAS V2


Enter
Shift
Alt
Ctrl
PgUp
PgDn
End
JTAG
CON1
SW4
SW5
SW6
NUMATO LAB
www.numato.com
MIMAS V2


Enter
Shift
PgUp
PgDn
End
Alt
Ctrl
End


Shift
Alt
Ctrl
PgUp
PgDn
End
StP.
JTAG
ICSP
FW UP
SPARTAN-6
SW4
SW5
SW6
SW1
NUMATO
MIMAS V2