

**UNIVERSIDAD NACIONAL AUTÓNOMA  
DE MÉXICO**

---

---

**FACULTAD DE INGENIERIA**

**PROYECTO DISEÑO DIGITAL MODERNO**

**RELOJ DE AJEDREZ DISEÑADO EN VHDL**

**HECHO POR LOS ALUMNOS:**

**Martínez García Isaac  
Overa Martínez Rodrigo Iván**

**P R E S E N T A :**



**Semestre 2021-2**

## **Tabla de Contenido**

- 0.Introducción
- 1. Descripción del Proyecto
- 2. Diagrama de entradas
- 3. Objetivo
- 4. Código
- 5. Simulación
- 6. Conclusiones
- 7. Bibliografía

### *Introducción*

Para poder entender la realización de este proyecto es necesario dejar en claro que la composición de éste está basada en contadores escritos en vhdl.

Un contador digital es un circuito que realiza una cuenta ascendente o descendente entre un valor inicial y otro final.

Un contador es un circuito secuencial, es decir contiene elementos de memoria (FLIP-FLOPS) . Estos Flips-Flops almacenan el valor de la cuenta. Cada ciclo de reloj leemos el valor y lo incrementamos en uno, volviéndolo a guardar en el mismo registro. Podemos ver el contador como una maquina de estados cuyos estados son el 1,2,3,4,5,.. y claro podemos ir en decremento, pero eso ya entra en los tipos de coontadores.

Como mencionamos con anterioridad existen diferentes tipos de contadores y en cada aplicación debemos utilizar el que mas nos convenga, esto depende del modo en que se aplique la señal de reloj, los contadores se clasifican en dos amplias categorías: asíncronos y síncronos. En los contadores asíncronos, normalmente denominados contadores con propagación (ripple counters), se aplica una señal de reloj externa a la entrada de reloj del primer flip-flop y luego a los siguientes flip-flops se les aplica la señal de reloj mediante la salida del flipflop anterior. En los contadores síncronos, la entrada de reloj se conecta a todos los flip-flops, de forma que se les aplica la señal de reloj simultáneamente.Dentro de cada una de estas dos categorías, los contadores se clasifican por el tipo de secuencia, el número de estados o el número de flip-flops del contador, teniendo esto en cuenta utilizaremos los descendentes básicos para nuestro reloj de ajedrez.

*Descripción del Proyecto:*

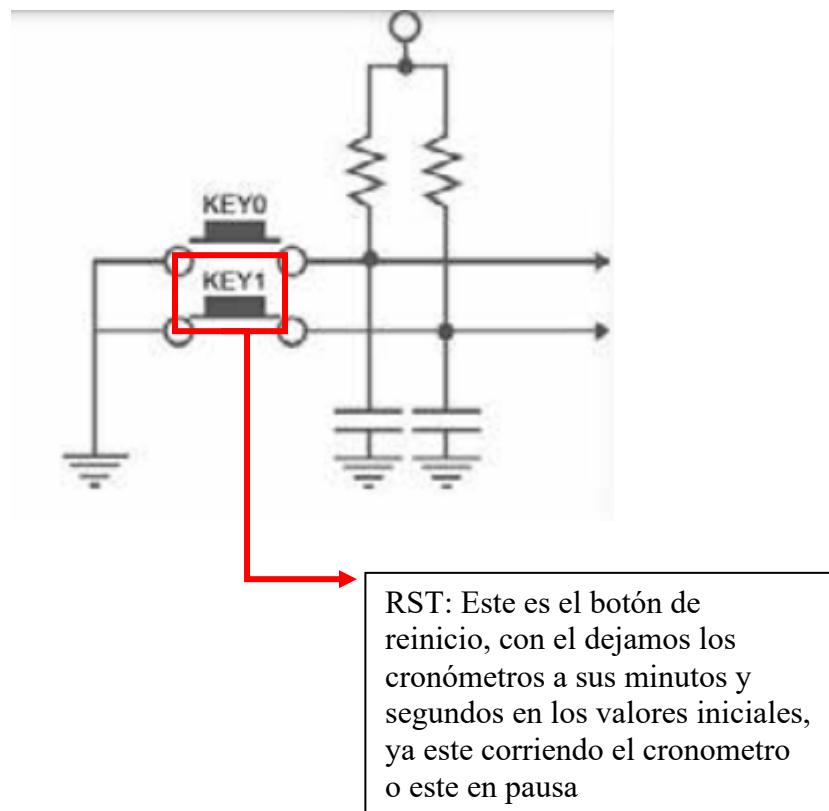
El sistema controlara una partida de Ajedrez para gestionar los tiempos utilizados en cada partida. Las premisas son las siguientes:

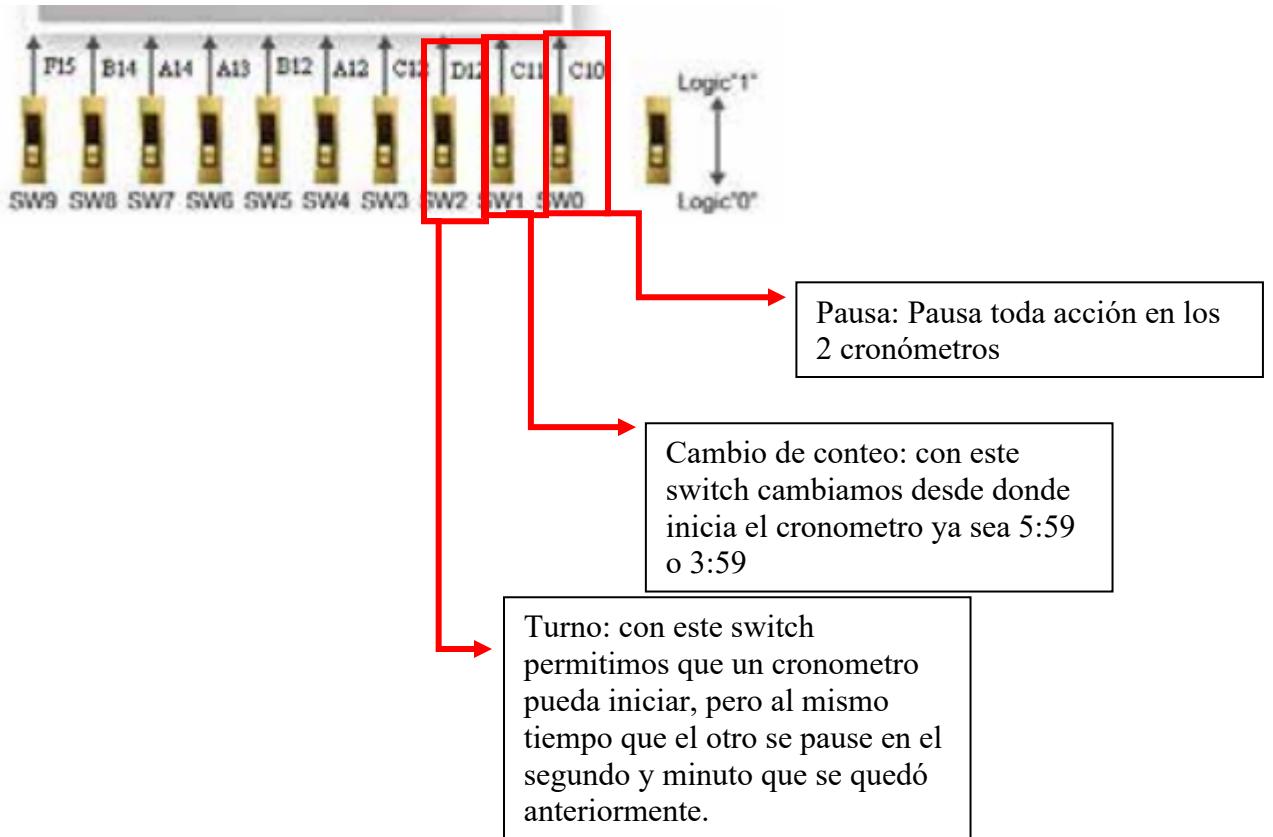
El reloj correrá de manera descendente, al llegar el tiempo de cualquier jugador a "00:00:00" la partida habrá terminado.

El tiempo entre movimientos será contabilizado y mostrado a los jugadores, podrán tener 2 formas de juegos.

Las entradas del sistema serán: botón de reinicio, switch pausa, switch cambio de conteo, switch turno.

*Diagrama de entradas:*





### Objetivo:

Se requiere la elaboración de un código para programar un reloj de ajedrez que realizará un conteo, manejando el formato MM: SS, el conteo iniciará en 00:00 y para finalizar estará se podrán seleccionar alguna de las dos formas posibles, la primera hasta 03:59 y la otra hasta 5:59. Otra característica es que cuando cualquiera de los jugadores llegue al tiempo límite, entonces el reloj del segundo jugador también se detendrá y, por lo tanto, se tendrán que reiniciar los temporizadores.

Para cumplir con el objetivo se podrán manejar los siguientes módulos en VHDL:

- Divisor de frecuencia
- Un módulo o componente para las unidades de segundo
- Un módulo o componente para las decenas de segundo
- Un módulo o componente para las unidades de minuto
- Un módulo o componente para las decenas de minutos
- Un módulo o componente para detener los temporizadores
- Un módulo o componente para el reinicio
- Un módulo o componente para el manejo de los displays de siete segmentos

Las acciones que se manejarán en el sistema podrán ser realizadas mediante los switchs que se requieran, la señal de reinicio es recomendable que sea mediante un botón.

*Código:*

#### Archivo Top:

```
library ieee;
use ieee.std_logic_1164.all;

entity TOP is
port(clk: in std_logic;
      ssd1:out std_logic_vector(6 downto 0);
      ssu1:out std_logic_vector(6 downto 0);
      ssu2:out std_logic_vector(6 downto 0);
      ssd2:out std_logic_vector(6 downto 0);
      ssu3:out std_logic_vector(6 downto 0);
      ssu4:out std_logic_vector(6 downto 0);
      pausa: in std_logic;
      rst: in std_logic;
      sel: in std_logic;
      turno: in std_logic);

end entity;

architecture arqTOP of TOP is

signal clk1: std_logic;
signal f, vuelta, f1: std_logic := '0';
signal g, g1: std_logic ;
signal T2:integer range 0 to 60;
signal T3:integer range 0 to 60;
signal salida:integer range 0 to 60;
signal segundos,segundos1 :integer range 0 to 60;
signal minutos,minutos1:integer range 0 to 60;
signal conteo1:integer range 0 to 60;

begin
    u1: entity work.divf(arqdivf) port map(clk, clk1);
    u2: entity work.cont2(arqcont2) port map(f, minutos, g,
rst,sel);
    u3: entity work.cont1(arqcont1) port map(clk1,
segundos, f, pausa, rst, g, turno);
    u4: entity work.cont2(arqcont2) port map(f1, minutos1,
g1, rst,sel);
    u5: entity work.cont1b(arqcont1b) port map(clk1,
segundos1, f1, pausa, rst, g1,turno);
    u6: entity work.ssd(arqssd) port map(segundos, ssd1);
    u7: entity work.ssu(arqssu) port map(segundos, ssu1);
    u8: entity work.ssu(arqssu) port map(minutos, ssu2);
    u9: entity work.ssd(arqssd) port map(segundos1, ssd2);
    u10: entity work.ssu(arqssu) port map(segundos1,
ssu3);
    u11: entity work.ssu(arqssu) port map(minutos1, ssu4);
end architecture;
```

#### Archivo ssu:

```
library ieee;
use ieee.std_logic_1164.all;
entity ssu is
port(a: in integer;
      f: out std_logic_vector(6 downto 0));
end entity;
architecture arqssu of ssu is
begin
with a select
f<="1000000" when 0,
          "1111001" when 1,
          "0100100" when 2,
          "0110000" when 3,
          "0011001" when 4,
          "0010010" when 5,
          "0000010" when 6,
          "1111000" when 7,
          "0000000" when 8,
          "0011000" when 9,
          "1000000" when 10,
          "1111001" when 11,
          "0100100" when 12,
          "0110000" when 13,
          "0011001" when 14,
          "0010010" when 15,
          "0000010" when 16,
          "1111000" when 17,
          "0000000" when 18,
          "0011000" when 19,
          "1000000" when 20,
          "1111001" when 21,
          "0100100" when 22,
          "0110000" when 23,
          "0011001" when 24,
          "0010010" when 25,
          "0000010" when 26,
          "1111000" when 27,
          "0000000" when 28,
          "0011000" when 29,
          "1000000" when 30,
          "1111001" when 31,
          "0100100" when 32,
          "0110000" when 33,
          "0011001" when 34,
          "0010010" when 35,
          "0000010" when 36,
          "1111000" when 37,
          "0000000" when 38,
          "0011000" when 39,
          "1000000" when 40,
          "1111001" when 41,
          "0100100" when 42,
          "0110000" when 43,
          "0011001" when 44,
          "0010010" when 45,
          "0000010" when 46,
          "1111000" when 47,
          "0000000" when 48,
          "0011000" when 49,
          "1000000" when 50,
          "1111001" when 51,
```

```

    "0100100" when 52,
    "0110000" when 53,
    "0011001" when 54,
    "0010010" when 55,
    "0000010" when 56,
    "1111000" when 57,
    "0000000" when 58,
    "0011000" when 59,
    "1111111" when others;
end architecture;

```

### Archivo ssd:

```

library ieee;
use ieee.std_logic_1164.all;
entity ssd is
port(a: in integer;
      f: out std_logic_vector(6 downto 0));
end entity;
architecture arqssd of ssd is
begin
  with a select
    f<="1000000" when 0,
    "1000000" when 1,
    "1000000" when 2,
    "1000000" when 3,
    "1000000" when 4,
    "1000000" when 5,
    "1000000" when 6,
    "1000000" when 7,
    "1000000" when 8,
    "1000000" when 9,
    "1111001" when 10,
    "1111001" when 11,
    "1111001" when 12,
    "1111001" when 13,
    "1111001" when 14,
    "1111001" when 15,
    "1111001" when 16,
    "1111001" when 17,
    "1111001" when 18,
    "1111001" when 19,
    "0100100" when 20,
    "0100100" when 21,
    "0100100" when 22,
    "0100100" when 23,
    "0100100" when 24,
    "0100100" when 25,
    "0100100" when 26,
    "0100100" when 27,
    "0100100" when 28,
    "0100100" when 29,
    "0110000" when 30,
    "0110000" when 31,
    "0110000" when 32,
    "0110000" when 33,
    "0110000" when 34,
    "0110000" when 35,
    "0110000" when 36,
    "0110000" when 37,
    "0110000" when 38,
    "0110000" when 39,
    "0011001" when 40,
    "0011001" when 41,
    "0011001" when 42,
    "0011001" when 43,
    "0011001" when 44,
    "0011001" when 45,
    "0011001" when 46,
    "0011001" when 47,
    "0011001" when 48,
    "0011001" when 49,
    "0010010" when 50,
    "0010010" when 51,
    "1111111" when others;
end architecture;

```

### Archivo divf:

```

library ieee;
use ieee.std_logic_1164.all;

entity divf is
port(clk: in std_logic;
      clk1: buffer std_logic:='0');
end entity;

architecture arqdivf of divf is
signal contador: integer range 0 to 25000000;
begin
  process(clk)
  begin
    if(rising_edge(clk)) then
      if(contador=25000000) then
        contador<=0;
      else
        contador<= contador+1;
      end if;
    end process;
  end architecture;

```

### Archivo cont2:

```

library ieee;
use ieee.std_logic_1164.all;

entity cont2 is
port(a: in std_logic;
      conteo: buffer integer:=0;
      b: buffer std_logic;
      rst: in std_logic;
      sel: in std_logic);
end entity;

architecture arqcont2 of cont2 is
begin
  process(a)
  begin
    if (rst = '1') then
      if(rising_edge(a)) then
        if (sel='1')then
          conteo <= 3;
        b <='1';
      else
        conteo <= 5;
        b <='1';
      end if;
    end if(b = '1') then
      if(conteo = 0) then

```

```

--conteo <= conteo;
b <='0';

else
conteo <= conteo -1;
b <='1';

end if;
end if;
end if;
else

b <='1';
if (sel='1')then
b <='1';
conteo <= 3;
else
b <='1';
conteo <= 5;

end if;
end if;
end process;
end architecture;

```

### Archivo: cont1:

```

library ieee;
use ieee.std_logic_1164.all;

entity cont1 is
port(clk: in std_logic;
conteo: buffer integer;
f: out std_logic;
pausa: in std_logic;
rst: in std_logic;
b: in std_logic;
turno:in std_logic);
end entity;

architecture arqcont1 of cont1 is
begin

```

```

process(clk)
begin

if(rising_edge(clk)) then
if(b = '0')then
conteo <= 00;

else

if (rst = '1') then
if (turno = '0') then
conteo <= conteo;
else
if (pausa = '1') then
conteo <= conteo;
else
if(conteo = 00) then
f<='1';
conteo <= 59;

else
conteo <= conteo -1;
f<='0';
end if;
end if;
end if;
else
conteo <= 0;

```

### Archivo: cont1B:

```

library ieee;
use ieee.std_logic_1164.all;
```

```

entity cont1B is
port(clk: in std_logic;
conteo: buffer integer;
f: out std_logic;
pausa: in std_logic;
rst: in std_logic;
b: in std_logic;
turno:in std_logic);
end entity;
```

```

architecture arqcont1B of cont1b is
begin
```

```

process(clk)
begin

if(rising_edge(clk)) then
if(b = '0')then
conteo <= 00;

else

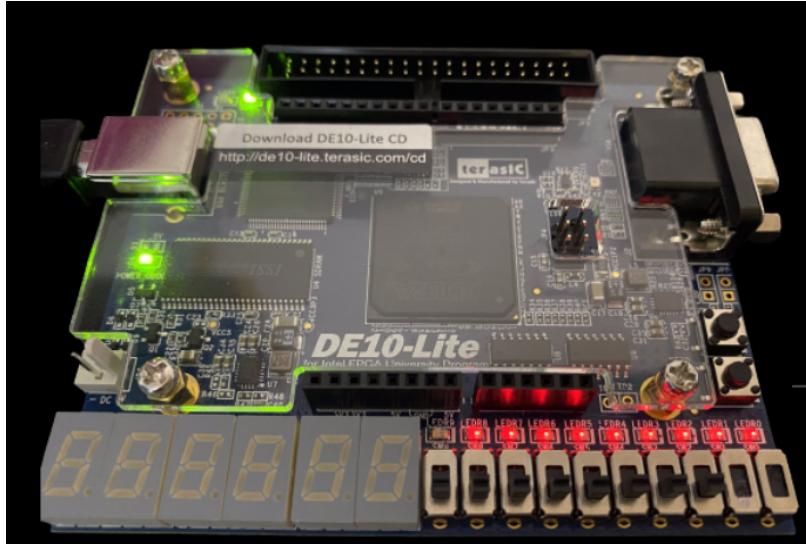
if (rst = '1') then
if (turno = '1') then
conteo <= conteo;
else
if (pausa = '1') then
conteo <= conteo;
else
if(conteo = 00) then
f<='1';
conteo <= 59;

else
conteo <= conteo -1;
f<='0';
end if;
end if;
end if;
else
conteo <= 0;
f<='1';
end if;
end if;
end process;

end architecture;
```

## *Simulación:*

Será diseñado e implementado en la tarjeta DE10-Lite junto con Intel Quartus II



Pin planner en nuestro software para definir los pines como entrada...

Pin Planner - C:/Users/ghj\_m/Dropbox/Mi PC (LAPTOP-6MTP1VSE)/Downloads/Top/Top - Top

File Edit View Processing Tools Window Help

Search altera.com



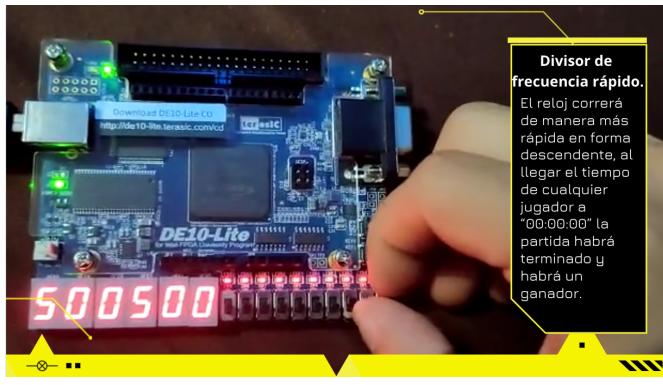
**Named:** \* **Edit:** X **Filter:** Pins: all

**Node Name** **Direction** **Location** **I/O Bank** **VREF Group** **Filter Location** **I/O Standard** **Reserved** **Current Strength** **Slew Rate** **Differential Pair** **Strict Preservation**

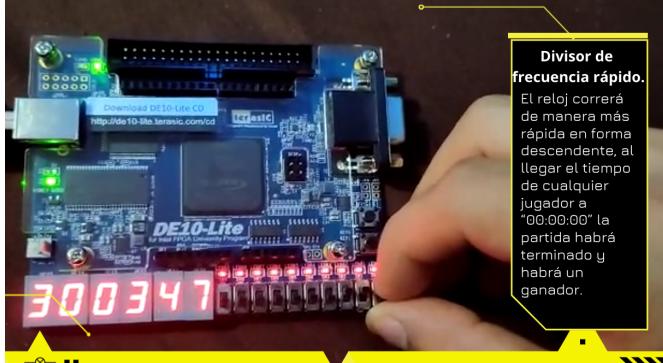
**All Pins**

clk	Input	PIN_N14	6	B6_N0	PIN_N14	2.5 V		12mA (default)			
pausa	Input	PIN_C10	7	B7_N0	PIN_C10	2.5 V		12mA (default)			
rst	Input	PIN_A7	7	B7_N0	PIN_A7	2.5 V		12mA (default)			
sel	Input	PIN_C11	7	B7_N0	PIN_C11	2.5 V		12mA (default)			
out ssd1[6]	Output	PIN_B17	7	B7_N0	PIN_B17	2.5 V		12mA (default)	2 (default)		
out ssd1[5]	Output	PIN_A18	7	B7_N0	PIN_A18	2.5 V		12mA (default)	2 (default)		
out ssd1[4]	Output	PIN_A17	7	B7_N0	PIN_A17	2.5 V		12mA (default)	2 (default)		
out ssd1[3]	Output	PIN_B16	7	B7_N0	PIN_B16	2.5 V		12mA (default)	2 (default)		
out ssd1[2]	Output	PIN_E18	6	B6_N0	PIN_E18	2.5 V		12mA (default)	2 (default)		
out ssd1[1]	Output	PIN_D18	6	B6_N0	PIN_D18	2.5 V		12mA (default)	2 (default)		
out ssd1[0]	Output	PIN_C18	7	B7_N0	PIN_C18	2.5 V		12mA (default)	2 (default)		
out ssd2[6]	Output	PIN_F20	6	B6_N0	PIN_F20	2.5 V		12mA (default)	2 (default)		
out ssd2[5]	Output	PIN_F19	6	B6_N0	PIN_F19	2.5 V		12mA (default)	2 (default)		
out ssd2[4]	Output	PIN_H19	6	B6_N0	PIN_H19	2.5 V		12mA (default)	2 (default)		
out ssd2[3]	Output	PIN_J18	6	B6_N0	PIN_J18	2.5 V		12mA (default)	2 (default)		
out ssd2[2]	Output	PIN_E19	6	B6_N0	PIN_E19	2.5 V		12mA (default)	2 (default)		
out ssd2[1]	Output	PIN_E20	6	B6_N0	PIN_E20	2.5 V		12mA (default)	2 (default)		
out ssd2[0]	Output	PIN_F18	6	B6_N0	PIN_F18	2.5 V		12mA (default)	2 (default)		
out ssu1[6]	Output	PIN_C17	7	B7_N0	PIN_C17	2.5 V		12mA (default)	2 (default)		
out ssu1[5]	Output	PIN_D17	7	B7_N0	PIN_D17	2.5 V		12mA (default)	2 (default)		
out ssu1[4]	Output	PIN_E16	7	B7_N0	PIN_E16	2.5 V		12mA (default)	2 (default)		
out ssu1[3]	Output	PIN_C16	7	B7_N0	PIN_C16	2.5 V		12mA (default)	2 (default)		
out ssu1[2]	Output	PIN_C15	7	B7_N0	PIN_C15	2.5 V		12mA (default)	2 (default)		
out ssu1[1]	Output	PIN_E15	7	B7_N0	PIN_E15	2.5 V		12mA (default)	2 (default)		
out ssu1[0]	Output	PIN_C14	7	B7_N0	PIN_C14	2.5 V		12mA (default)	2 (default)		
out ssu2[6]	Output	PIN_B22	6	B6_N0	PIN_B22	2.5 V		12mA (default)	2 (default)		
out ssu2[5]	Output	PIN_C22	6	B6_N0	PIN_C22	2.5 V		12mA (default)	2 (default)		
out ssu2[4]	Output	PIN_B21	6	B6_N0	PIN_B21	2.5 V		12mA (default)	2 (default)		
out ssu2[3]	Output	PIN_A21	6	B6_N0	PIN_A21	2.5 V		12mA (default)	2 (default)		
out ssu2[2]	Output	PIN_B19	7	B7_N0	PIN_B19	2.5 V		12mA (default)	2 (default)		
out ssu2[1]	Output	PIN_A20	7	B7_N0	PIN_A20	2.5 V		12mA (default)	2 (default)		
out ssu2[0]	Output	PIN_B20	6	B6_N0	PIN_B20	2.5 V		12mA (default)	2 (default)		
out ssu3[6]	Output	PIN_E17	6	B6_N0	PIN_E17	2.5 V		12mA (default)	2 (default)		
out ssu3[5]	Output	PIN_D19	6	B6_N0	PIN_D19	2.5 V		12mA (default)	2 (default)		
out ssu3[4]	Output	PIN_C20	6	B6_N0	PIN_C20	2.5 V		12mA (default)	2 (default)		
out ssu3[3]	Output	PIN_C19	7	B7_N0	PIN_C19	2.5 V		12mA (default)	2 (default)		
out ssu3[2]	Output	PIN_E21	6	B6_N0	PIN_E21	2.5 V		12mA (default)	2 (default)		
out ssu3[1]	Output	PIN_E22	6	B6_N0	PIN_E22	2.5 V		12mA (default)	2 (default)		
out ssu3[0]	Output	PIN_F21	6	B6_N0	PIN_F21	2.5 V		12mA (default)	2 (default)		
out ssu4[6]	Output	PIN_N20	6	B6_N0	PIN_N20	2.5 V		12mA (default)	2 (default)		
out ssu4[5]	Output	PIN_N19	6	B6_N0	PIN_N19	2.5 V		12mA (default)	2 (default)		
out ssu4[4]	Output	PIN_M20	6	B6_N0	PIN_M20	2.5 V		12mA (default)	2 (default)		
out ssu4[3]	Output	PIN_N18	6	B6_N0	PIN_N18	2.5 V		12mA (default)	2 (default)		
out ssu4[2]	Output	PIN_L18	6	B6_N0	PIN_L18	2.5 V		12mA (default)	2 (default)		
out ssu4[1]	Output	PIN_K20	6	B6_N0	PIN_K20	2.5 V		12mA (default)	2 (default)		
out ssu4[0]	Output	PIN_J20	6	B6_N0	PIN_J20	2.5 V		12mA (default)	2 (default)		
turno	Input	PIN_D12	7	B7_N0	PIN_D12	2.5 V		12mA (default)			
<<new node>>											

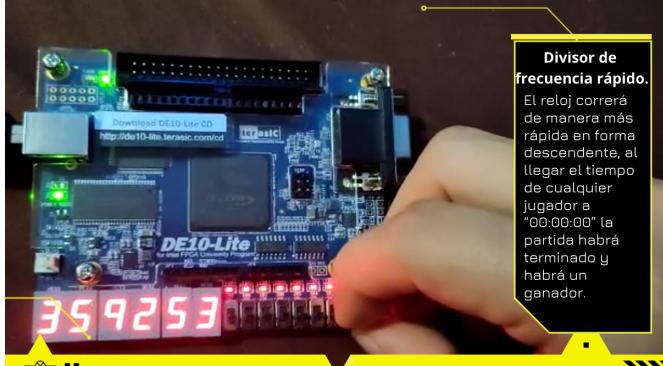
100% 00:00:50



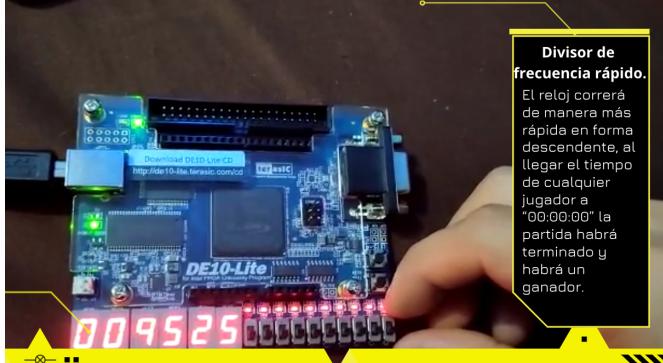
Modo 1: 5 minutos



Modo 2: 3 minutos



Modo: cambio de jugador



Modo: pausa

*Conclusión:*

Como podemos apreciar las tarjetas fpga manejan una máquina de estados, cada estado siempre será función del estado anterior y de las entradas, pero a la vez atendiendo la forma en que se generan las salidas, es por esto por lo que el lenguaje VHDL esta pensado en ayudar a resolver cualquier problema por medio de esta técnica de desarrollo de sistemas.

VHDL es un lenguaje estructural, complejo, pero sistemático por lo cual es conveniente desarrollar el proyecto y practicar constantemente para un buen desarrollo posterior en lo que sea que deseemos.

*Bibliografía:*

*Diseño digital.* John F. Wakerley. 3<sup>a</sup> Edición. Prentice Hall, 2000.

*Fundamentos de sistemas digitales.* 7<sup>a</sup> Edición. Thomas Floyd. Prentice Hall, 2000.

[http://www.dte.upct.es/personal/andres\\_iborra/docencia/sis\\_elec/](http://www.dte.upct.es/personal/andres_iborra/docencia/sis_elec/)

[http://www.dte.uvigo.es/logica\\_programable/Recursos.htm](http://www.dte.uvigo.es/logica_programable/Recursos.htm)