UNIVERSIDAD
NEBRIJA

TRABAJO FINAL LABORATORIO

Por:
ARTURO REQUEJO PORTILLA
SARA MARCOS CORNEJO

UNIVERSIDAD
NEBRIJA

Introducción al proyecto:

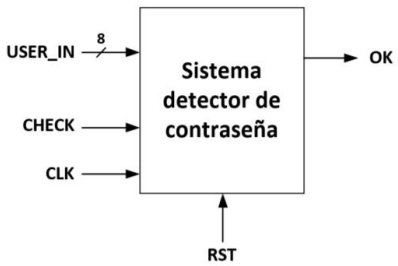
Para realizar este proyecto se ha de diseñar un sistema que sea capaz de comprobar una contraseña de 8 bits que estará almacenada en una memoria ROM (Read-only memory). Posteriormente se proseguirá implementando algunas ampliaciones a este sistema. Para hacer esto se ha usado el programa "Vivado", con código vhdl.

Descripción del trabajo realizado detallando los pasos seguidos. Sin ampliaciones

Como se ha explicado previamente, la base del trabajo consiste en diseñar un sistema de comprobación de contraseña. El sistema tendrá una contraseña de 8 bits almacenada en memoria ROM y, si el usuario introduce la misma contraseña, se deberá encender un LED de "contraseña correcta". Tendrá las siguientes características:

- La contraseña que estará almacenada en la memoria ROM será: 0xEE. Esta contraseña en binario equivale a: "11101110".
- Para que el usuario introduzca la contraseña se utilizarán interruptores.
- Habrá un botón de validación de contraseña, que se pulsará después de haber colocado los interruptores en sus respectivas posiciones. Hasta que no se pulse dicho botón, no se comprobará la contraseña.
- Si la contraseña introducida es correcta se encenderá un LED de aviso que permanecerá encendido hasta que se resetee el sistema.

La entidad del programa tendrá un esquema similar a este:



Siendo,
USER_IN: Señal de entrada de 8 bits para que el usuario introduzca la contraseña.
CHECK: Señal para validar si la contraseña introducida es correcta o no.
CLK: Señal de reloj utilizada para sincronizar el funcionamiento del sistema.
RST: Señal de reinicio.
OK: Señal conectada a un LED que muestra si la contraseña introducida es correcta.

El primer paso que hicimos fue diseñar la ROM, con las siguientes entradas y salidas:

```
Port(CLK: in  std_logic;                        -- Clock
     Read: in  std_logic;                       -- Read
     D_out: out std_logic_vector(7 downto 0)    -- Data Out (8 bits)
     );
```

Para establecer la contraseña en la ROM hicimos lo siguiente:

```
constant ROM_content: ROM_Array:= (
        0=> "11101110",
        others=>"00000000");
```

*El código completo de la ROM en vhdl se encuentra en el apartado posterior: Códigos VHDL usados. Simulación del trabajo

Una vez definida la entidad y el comportamiento de la memoria ROM, creamos el sistema con sus respectivas entradas y salidas. Instanciamos el componente de la ROM creada mapeando los puertos y comenzamos a describir su comportamiento con el uso de dos process. El primero de ellos tenía como lista de sensibilidad la señal CLK, lo que hacía que fuese un sistema síncrono, y que cada vez que se pulsase RST se reseteara el sistema de la siguiente forma:

```
process(CLK)begin
   if(rising_edge(CLK))then
      if(RST = '1')then
         user_in_signal <= "ZZZZZZZZ";
      else
         user_in_signal <= user_in;
      end if;
   end if;
end process;
```

El segundo process tenía como lista de sensibilidad la señal CHECK (botón de validación de contraseña). Si se pulsaba este botón y la contraseña que se introdujo era la correcta, la señal de OK se encendía.

```
process(CHECK) begin
Read_aux <= '1';
   if(CHECK = '1')then
      if(user_in_signal = D_aux)then
         OK <= '1';
      else
         OK <= '0';
      end if;
   else
      OK<= '0';
   end if;
end process;
```

También usamos un fichero de constraints para conectar la entrada de USER_IN con interruptores y la señal de salida OK con un LED para verificar si la contraseña es la correcta.

Códigos VHDL usados. Simulación del trabajo. Sin ampliaciones

ROM.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity ROM is
Port(CLK   : in  std_logic;                          -- Clock
     Read  : in  std_logic;                          -- Read
     D_out : out std_logic_vector(7 downto 0)        -- Data Out (8 bits)
     );
end ROM;

architecture Behavioral of ROM is
type ROM_Array is array (0 to 0) of std_logic_vector(7 downto 0);
constant ROM_content : ROM_Array := (
          0=> "11101110",
          others=>"00000000");
begin
process(CLK)
          begin
          if (rising_edge(CLK)) then
               if (Read = '1') then
               D_out <= ROM_content(0);
               else
               D_out <= "ZZZZZZZZ";
               end if;
          end if;

end process;
end Behavioral;
```

Constraints.xdc

```
# Map Input
set_property PACKAGE_PIN J15     [get_ports USER_IN[0]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[0]]

set_property PACKAGE_PIN L16     [get_ports USER_IN[1]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[1]]

set_property PACKAGE_PIN M13     [get_ports USER_IN[2]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[2]]

set_property PACKAGE_PIN R15     [get_ports USER_IN[3]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[3]]

set_property PACKAGE_PIN R17     [get_ports USER_IN[4]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[4]]

set_property PACKAGE_PIN T18     [get_ports USER_IN[5]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[5]]

set_property PACKAGE_PIN U18     [get_ports USER_IN[6]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[6]]

set_property PACKAGE_PIN R13     [get_ports USER_IN[7]]
set_property IOSTANDARD LVCMOS33 [get_ports USER_IN[7]]



# Map Output
set_property PACKAGE_PIN L18     [get_ports OK]
set_property IOSTANDARD LVCMOS33 [get_ports OK]
```

sistema.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sistema is
Port (CLK: in std_logic;
      RST: in std_logic;
       USER_IN: in std_logic_vector(7 downto 0);
      CHECK: in std_logic;
      OK: out std_logic);
end sistema;

architecture Behavioral of sistema is
   component ROM is
         port(CLK   : in  std_logic;                          -- Clock
              Read  : in  std_logic;                          -- Read
              D_out : out std_logic_vector(7 downto 0)     -- Data Out (8 bits)
         );
   end component;

signal Read_aux: std_logic;
signal D_aux : std_logic_vector(7 downto 0);
signal user_in_signal: std_logic_vector(7 downto 0);
signal Y: std_logic;

begin
   ROM1 : ROM  port map(
      CLK   => CLK,
      Read   => Read_aux,
      D_out => D_aux
   );

process(CLK)begin
   if(rising_edge(CLK))then
      if(RST = '1')then
         user_in_signal <= "ZZZZZZZZ";
      else
         user_in_signal <= user_in;
      end if;
   end if;
end process;

process(CHECK) begin
Read_aux <= '1';
   if(CHECK = '1')then
      if(user_in_signal = D_aux)then
         OK <= '1';*
*  else
         OK <= '0';
      end if;
   else
      OK<= 'U';
   end if;
end process;
end Behavioral;
```

sistema_tb.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_sistema is
end tb_sistema;

architecture Behavioral of tb_sistema is
component sistema is
    Port(CLK: in std_logic;
      RST: in std_logic;
       USER_IN: in std_logic_vector(7 downto 0);
      CHECK: in std_logic;
      OK: out std_logic);
end component;
signal CLK,RST,CHECK,OK: std_logic;
signal USER_IN: std_logic_vector(7 downto 0);
begin
DUT: sistema port map(CLK,RST,USER_IN,CHECK,OK);

process
begin
CLK<='0'; wait for 5ns;
CLK<='1'; wait for 5ns;
end process;

process begin
CHECK<='0'; wait for 10 ns;
USER_IN<="11111111"; wait for 10 ns;
CHECK<='1'; wait for 10 ns;
CHECK<='0'; wait for 10 ns;
USER_IN<="11101110"; wait for 10 ns;
CHECK<='1'; wait for 10 ns;
end process;

process begin
RST<='0'; wait for 10 ns;
RST<='1'; wait for 10 ns;
RST<='0'; wait;
end process;

end Behavioral;
```

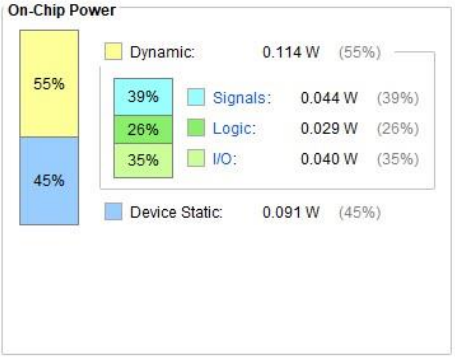
En esta simulación se puede observar un ejemplo en el que la contraseña es incorrecta y otro en el que la contraseña es correcta. En el primer caso, cuando se pulsa el botón de CHECK la señal de OK vale 0. Cuando es correcta se puede ver que al pulsar CHECK, OK = 1. Si CHECK no está pulsado, OK estará sin definir al no saber su valor.

Análisis de tiempo, área y consumo. Sin ampliaciones

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.205 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**25,9°C**
Thermal Margin:	59,1°C (12,8 W)
Effective ϑJA:	4,6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



Name	Slice LUTs (63400)	Slice Registers (126800)	Slice (15850)	LUT as Logic (63400)	Block RAM Tile (135)	Bonded IPADs (2)	BUFIO (24)
sistema	7	9	3	7	9	11	1

1.1 On-Chip Components

On-Chip	Power (W)	Used	Available	Utilization (%)
Slice Logic	0.029	23	---	---
LUT as Logic	0.021	7	63400	0.01
BUFG	0.006	1	32	3.13
Register	0.003	9	126800	<0.01
Others	0.000	2	---	---
Signals	0.044	30	---	---
I/O	0.040	11	210	5.24
Static Power	0.091			
Total	0.205			

1. Summary

Total On-Chip Power (W)	0.205
Design Power Budget (W)	Unspecified*
Power Budget Margin (W)	NA
Dynamic (W)	0.114
Device Static (W)	0.091
Effective TJA (C/W)	4.6
Max Ambient (C)	84.1
Junction Temperature (C)	25.9
Confidence Level	Low
Setting File	---
Simulation Activity File	---
Design Nets Matched	NA

Descripción del trabajo realizado detallando los pasos seguidos.

Ampliación 1:

Para la primera ampliación, se propuso realizar un sistema que cuente el número de contraseñas incorrectas introducidas. En el caso de que se introduzca 3 veces seguidas la contraseña de forma incorrecta, el sistema se bloquearía y la señal de OK no se debe activar aunque se introduzca posteriormente la contraseña correcta. La forma de salir de este estado de bloqueo seía mediante la señal de reset.

Así, diseñamos un contador capaz de realizar esta cuenta:

```
signal cuenta : integer range 0 to 3; --Cuenta hasta 4
begin
process (CLK) begin
if (rising_edge(CLK)) then
   if (RST = '1') then --Se reinicia la cuenta al llegar reset a 1
      cuenta <= 0;
      Y <= '0';
   else
      if(OK = '1')then
         cuenta <= 0; --Se reinicia la cuenta al introducir la contraseña correcta
      else
         if (cuenta < 3) then
            if(CHECK = '1')then
               cuenta <= cuenta + 1;
               Y <= '0';
            end if;
         else
            Y <= '1';
         end if;
      end if;
   end if;
end if;
end process;
```

En el sistema instanciamos los componentes de la ROM y el contador creados y declaramos las señales nuevas necesarias.

Códigos VHDL usados. Simulación del trabajo. Ampliación 1:

Contador.vdh

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity contador is
Port ( CLK : in std_logic;
RST : in std_logic;
CHECK: in std_logic;
OK: in std_logic;
Y : out std_logic );
end contador;

architecture Behavioral of contador is
signal cuenta : integer range 0 to 3; --Cuenta hasta 3
begin
process (CLK) begin
if (rising_edge(CLK)) then
   if (RST = '1') then --Se reinicia la cuenta al llegar reset a 1
      cuenta <= 0;
      Y <= '0';
   else
      if(OK = '1')then
         cuenta <= 0; --Se reinicia la cuenta al introducir la contraseña correcta
      else
         if (cuenta < 3) then
            if(CHECK = '1')then
               cuenta <= cuenta + 1;
               Y <= '0';
            end if;
         else
            Y <= '1';
         end if;
      end if;
   end if;
end if;
end process;

end Behavioral;
```

ROM. vdh

Mismo código vhdl que en el trabajo sin ampliaciones

Constraints.xdc

Mismo código vhdl que en el trabajo sin ampliaciones

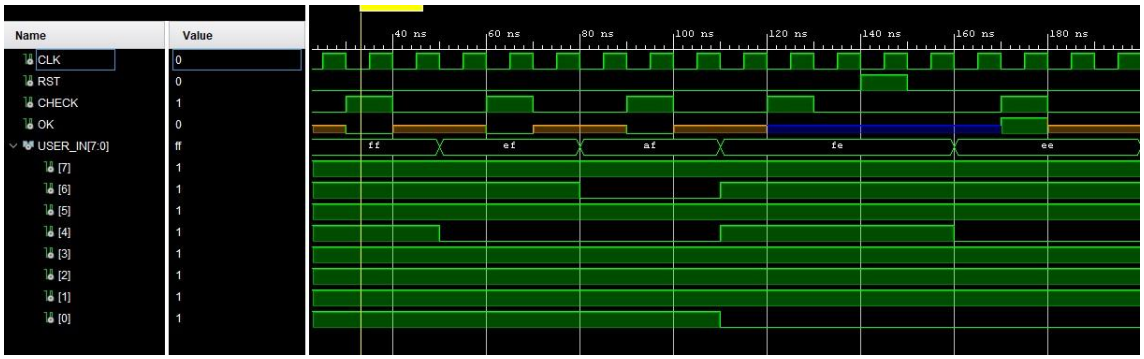
Sistema.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity sistema is
Port (CLK: in std_logic;
      RST: in std_logic;
       USER_IN: in std_logic_vector(7 downto 0);
      CHECK: in std_logic;
      OK: out std_logic
      );
end sistema;

architecture Behavioral of sistema is
   component ROM is
         port(CLK   : in  std_logic;                  -- Clock
              Read  : in  std_logic;                  -- Read
               D_out : out std_logic_vector(7 downto 0)      -- Data Out (8 bits)
         );
   end component;

   component contador is
      Port ( CLK : in std_logic;
RST : in std_logic;
      CHECK: in std_logic;
      OK: in std_logic;
      Y : out std_logic );
   end component;

signal D_aux, user_in_aux : std_logic_vector(7 downto 0);
signal Y_aux, OK_aux, Read_aux: std_logic;

begin
   ROM1 : ROM  port map(
      CLK   => CLK,
      Read  => Read_aux,
      D_out => D_aux
   );

   CONTADOR1: contador port map(
   CLK=>CLK,
   RST=>RST,
   CHECK=>CHECK,
   OK=>OK_aux,
   Y=>Y_aux
   );*
```

```
            *process(CLK)begin
      if(rising_edge(CLK))then
         if(RST = '1')then
            user_in_aux <= "ZZZZZZZZ";
         else
            user_in_aux <= user_in;
         end if;
      end if;
   end process;

   process(CHECK) begin
   if(y_aux /= '1')then
   Read_aux <= '1'; --Encendemos la ROM
      if(CHECK = '1')then
         if(user_in_aux = D_aux)then
            OK <= '1';
            OK_aux <= '1';
         else
            OK <= '0';
         end if;
      else
         OK<= 'U'; --Salida sin definir
      end if;
   else
      OK <='Z'; --Sistema bloqueado
   end if;
   end process;
   end Behavioral;
```

tb_sistema.vhd

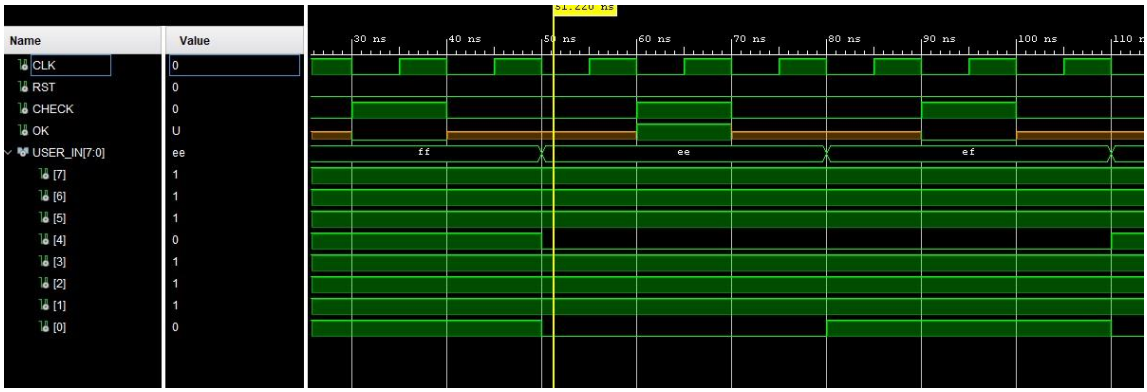
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_sistema is
end tb_sistema;

architecture Behavioral of tb_sistema is
component sistema is
    Port(CLK: in std_logic;
      RST: in std_logic;
       USER_IN: in std_logic_vector(7 downto 0);
      CHECK: in std_logic;
      OK: inout std_logic);
end component;
signal CLK,RST,CHECK,OK: std_logic;
signal USER_IN: std_logic_vector(7 downto 0);
begin
DUT: sistema port map(CLK => CLK,RST => RST,USER_IN =>
USER_IN,CHECK => CHECK,OK => OK);

process
begin
CLK<='0'; wait for 5ns;
CLK<='1'; wait for 5ns;
end process;
process begin
  CHECK <= '0'; wait for 10ns;
  RST <= '0'; wait for 10ns;
  user_in <= "11111111"; wait for 10ns;
  check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;-- si check no cambia no se vuelve a entrar en el
process(check)
  user_in <= "11101111"; wait for 10ns;
  check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;
  user_in <= "11101111"; wait for 10ns;
  check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;
  user_in <= "11101110"; wait for 10ns;
  check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;
  RST <= '1'; wait for 10ns;--reseteamos y probamos de nuevo a ver si cambia ok
  RST <= '0'; wait for 10ns;-- 10 ns es el tiempo predeterminado del rising edge
si lo bajamos hay problemas
  end process;
end Behavioral;
```



En este ejemplo, se puede ver como al introducir tres contraseñas erróneas se bloquea el sistema. Cuando RST = 1, el sistema se desbloquea y se activa la señal de OK. Z representa que el sistema está bloqueado.

Otro ejemplo de testbench sería:

```
process begin
  CHECK <= '0'; wait for 10ns;
  RST <= '0'; wait for 10ns;
  user_in <= "11111111"; wait for 10ns;
    check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;-- si check no cambia no se vuelve a entrar en el
process(check)
    user_in <= "11101111"; wait for 10ns;
  check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;
  user_in <= "10101111"; wait for 10ns;
  check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;
  user_in <= "11111110"; wait for 10ns;
  check <= '1'; wait for 10ns;
  check <= '0'; wait for 10ns;
  RST <= '1'; wait for 10ns;--reseteamos y probamos de nuevo a ver si cambia ok
    RST <= '0'; wait for 10ns;-- 10 ns es el tiempo predeterminado del rising edge
si lo bajamos hay problemas
   user_in <= "11101110"; wait for 10ns;
  check <= '1'; wait for 10ns;
end process;
```



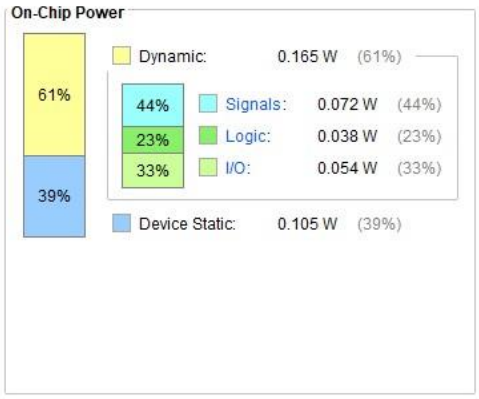
En este otro ejemplo, la primera contraseña es errónea, la segunda es correcta y la cuenta se resetea, haciendo que posteriormente OK esté definida.

Análisis de tiempo, área y consumo. Ampliación 1:

Power estimation from Synthesized netlist. Activity derived from constraints files, simulation files or vectorless analysis. Note: these early estimates can change after implementation.

Total On-Chip Power:	**0.27 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**26,2°C**
Thermal Margin:	58,8°C (12,8 W)
Effective ϑJA:	4,6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



Name	Slice LUTs (63400)	Slice Registers (126800)	Bonded IOB (210)	BUFGCTRL (32)
sistema	10	13	12	1
CONTADOR1 (contador)	3	3	0	0

1.1 On-Chip Components

On-Chip	Power (W)	Used	Available	Utilization (%)
Slice Logic	0.037	35	---	---
LUT as Logic	0.028	10	63400	0.02
BUFG	0.006	1	32	3.13
Register	0.003	13	126800	0.01
Others	0.000	6	---	---
Signals	0.085	36	---	---
I/O	0.045	12	210	5.71
Static Power	0.098			
Total	0.264			

1. Summary

Total On-Chip Power (W)	0.264
Design Power Budget (W)	Unspecified*
Power Budget Margin (W)	NA
Dynamic (W)	0.167
Device Static (W)	0.098
Effective TJA (C/W)	4.6
Max Ambient (C)	83.8
Junction Temperature (C)	26.2
Confidence Level	Low
Setting File	---
Simulation Activity File	---
Design Nets Matched	NA

Descripción del trabajo realizado detallando los pasos seguidos. Ampliación 2:

Para la segunda ampliación, se propuso implementar un mecanismo que permita cambiar la contraseña 0xEE almacenada en la memoria ROM por cualquier otra introducida por el usuario. La memoria ROM usada era de solo lectura (Read-only memory), por lo que no era posible realizar esta ampliación con este tipo de memoria. Se necesitaría una memoria RAM, dándonos la posibilidad de leer y escribir datos en memoria.

Códigos VHDL usados. Simulación del trabajo. Ampliación 2:

RAM.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity RAM is
    Port( CLK   : in  std_logic;                  -- Clock
      w_r_en : in  std_logic;                    -- Write
      D_in  : in  std_logic_vector(7 downto 0);
      --addr: in std_logic_vector(1 downto 0);
      D_out : out std_logic_vector(7 downto 0)      -- Data Out (8 bits)
    );
end RAM;

architecture Behavioral of RAM is
 type RAM_Array is array (0 to 1) of std_logic_vector(7 downto 0);
    signal RAM_content : RAM_Array:=(
        0=>"11101110",
        others=>"00000000"
        );

begin
    process(CLK)
    begin
        if(rising_edge(CLK))then
            if(w_r_en = '1')then
                RAM_CONTENT(0) <= D_in;
            end if;
            D_out <= RAM_CONTENT(0);
         end if;
    end process;
end Behavioral;
```

sistema.vhd

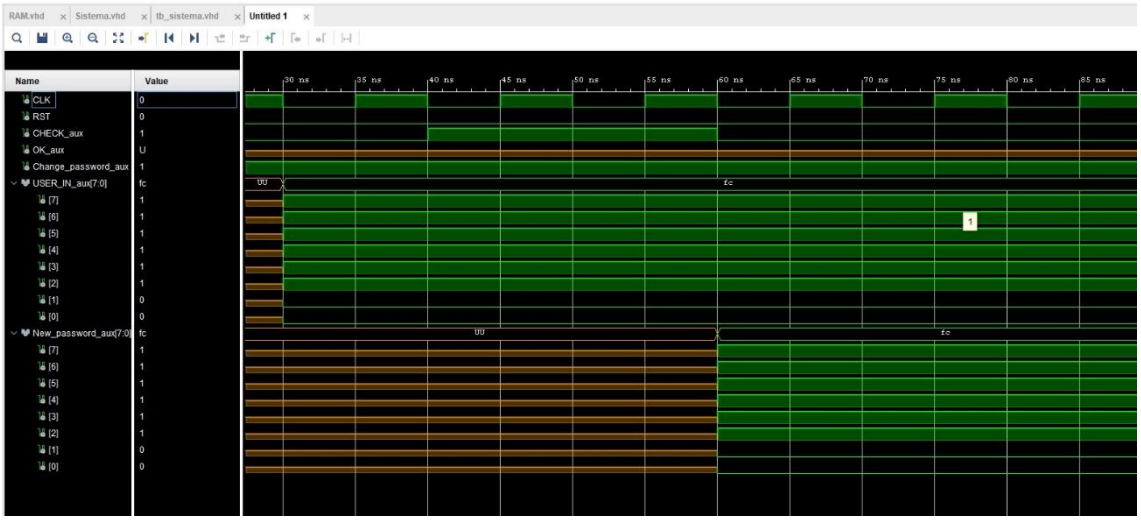
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Sistema is
Port (CLK: in std_logic;
      RST: in std_logic;
       USER_IN: in std_logic_vector(7 downto 0);
      CHECK: in std_logic;
      Change_password: in std_logic;
      New_password: out std_logic_vector(7 downto 0);
      OK: out std_logic
      );
end Sistema;

architecture Behavioral of Sistema is
   component RAM is
         Port( CLK   : in  std_logic;                      -- Clock
            w_r_en : in  std_logic;                        -- Write
            D_in  : in  std_logic_vector(7 downto 0);      -- Data In (8 bits)
            D_out : out std_logic_vector(7 downto 0)       -- Data Out (8 bits)
            );
   end component;

signal D_aux, user_in_aux, New_password_aux : std_logic_vector(7 downto 0);
signal Y_aux, OK_aux, E_aux, Read_aux, Write_aux: std_logic;

begin
   RAM1 : RAM  port map(
      CLK   => CLK,
      w_r_en => Write_aux,
      D_in => New_password_aux,
      D_out => D_aux
   );

process(CLK)begin
   if(rising_edge(CLK))then
      if(RST = '1')then
         user_in_aux <= "ZZZZZZZZ";
      else
         user_in_aux <= user_in;
      end if;
   end if;
end process;*
```

```
*process(CHECK) begin
if(y_aux /= '1')then
    if(Change_password = '0')then
        Write_aux <= '0';
        if(CHECK = '1')then
          if(user_in_aux = D_aux)then
            OK <= '1';
            OK_aux <= '1';
        else
            OK <= '0';
        end if;
    else
        OK<= 'U'; --Salida sin definir
    end if;
    else
    Write_aux <= '1';
      if(Write_aux <= '1')then --Se activa la
señal de escritura
          New_password_aux <= user_in_aux;
          New_password <=
New_password_aux;
       else
          New_password <= "ZZZZZZZZ";
       end if;
 end if;
end if;
end process;
end Behavioral;
```

sistema_tb.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_sistema is
end tb_sistema;

architecture Behavioral of tb_sistema is
component sistema is
    Port(CLK: in std_logic;
       RST: in std_logic;
        USER_IN: in std_logic_vector(7 downto 0);
       CHECK: in std_logic;
       Change_password: in std_logic;
       New_password: out std_logic_vector(7 downto 0);
       OK: out std_logic);
end component;
signal CLK,RST,CHECK_aux,OK_aux, Change_password_aux: std_logic;
signal USER_IN_aux, New_password_aux: std_logic_vector(7 downto 0);
begin*
```

```
*DUT: sistema port map(CLK => CLK,RST => RST,USER_IN =>
USER_IN_aux,CHECK => CHECK_aux,Change_password =>
Change_password_aux,New_password => New_password_aux,OK => OK_aux);

process begin
    CLK<='0'; wait for 5ns;
    CLK<='1'; wait for 5ns;
end process;

process begin
change_password_aux <= '1'; wait for 10ns;
 CHECK_aux <= '0'; wait for 10ns;
 RST <= '0'; wait for 10ns;
 user_in_aux <= "11111100"; wait for 10ns;
 check_aux <= '1'; wait for 10ns;
end process;

end Behavioral;
```

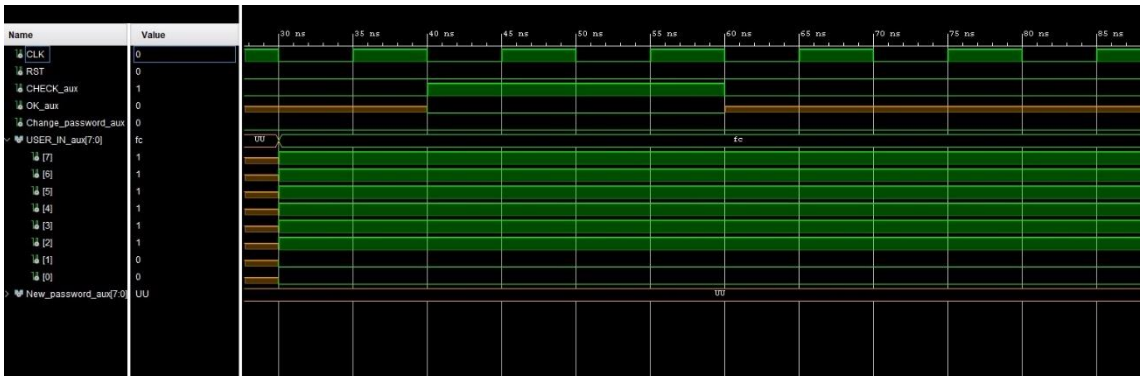


Como se puede observar, se produce el cambio de contraseña ya que la contraseña introducida por el usuario se iguala a la salida.

sistema_tb.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity tb_sistema is
end tb_sistema;*
```

```
*architecture Behavioral of tb_sistema is
component sistema is
    Port(CLK: in std_logic;
       RST: in std_logic;
        USER_IN: in std_logic_vector(7 downto 0);
       CHECK: in std_logic;
       Change_password: in std_logic;
       New_password: out std_logic_vector(7 downto 0);
       OK: out std_logic);
end component;
signal CLK,RST,CHECK_aux,OK_aux, Change_password_aux: std_logic;
signal USER_IN_aux, New_password_aux: std_logic_vector(7 downto 0);
begin

DUT: sistema port map(CLK => CLK,RST => RST,USER_IN =>
USER_IN_aux,CHECK => CHECK_aux,Change_password =>
Change_password_aux,New_password => New_password_aux,OK => OK_aux);

process begin
    CLK<='0'; wait for 5ns;
    CLK<='1'; wait for 5ns;
end process;

process begin
change_password_aux <= '0'; wait for 10ns;
 CHECK_aux <= '0'; wait for 10ns;
 RST <= '0'; wait for 10ns;
 user_in_aux <= "11111100"; wait for 10ns;
 check_aux <= '1'; wait for 10ns;
end process;

end Behavioral;
```



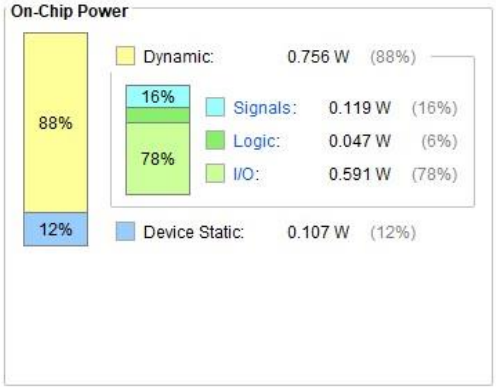
En este otro ejemplo de testbench, el sistema se comporta de forma que lee el contenido de la memoria, como si fuese una memoria ROM , activándose la señal de lectura.

Análisis de tiempo, área y consumo. Ampliación 2:

Power analysis from Implemented netlist. Activity derived from constraints files, simulation files or vectorless analysis.

Total On-Chip Power:	**0.863 W**
Design Power Budget:	**Not Specified**
Power Budget Margin:	**N/A**
Junction Temperature:	**28,9°C**
Thermal Margin:	56,1°C (12,2 W)
Effective ϑJA:	4,6°C/W
Power supplied to off-chip devices:	0 W
Confidence level:	Low

Launch Power Constraint Advisor to find and fix invalid switching activity



Name	Slice LUTs (63400)	Slice Registers (126800)	Slice (15850)	LUT as Logic (63400)	Block RAM Tile (135)	Bonded IPADs (2)	BUFIO (24)
Sistema	15	48	20	15	48	21	1
CONTADOR1 (Contador)	4	3	3	4	0	0	0
RAM1 (RAM)	5	16	5	5	0	0	0

On-Chip	Power (W)	Used	Available	Utilization (%)
Slice Logic	0.047	78	---	---
LUT as Logic	0.038	15	63400	0.02
BUFG	0.006	1	32	3.13
Register	0.003	48	126800	0.04
Others	0.000	8	---	---
Signals	0.119	77	---	---
I/O	0.591	21	210	10.00
Static Power	0.107			
Total	0.863			

Total On-Chip Power (W)	0.863
Design Power Budget (W)	Unspecified*
Power Budget Margin (W)	NA
Dynamic (W)	0.756
Device Static (W)	0.107
Effective TJA (C/W)	4.6
Max Ambient (C)	81.1
Junction Temperature (C)	28.9
Confidence Level	Low
Setting File	---
Simulation Activity File	---
Design Nets Matched	NA

Conclusiones y comentarios finales.

Lo que diferencia la memoria RAM de la ROM, no solo es porque es una memoria fija (es decir, no es volátil sino que los datos permanecen ahí incluso aun sin energía), sino que además es de solo lectura y no se puede volver a escribir nada en ella. En este tipo de memoria se almacenan cosas como la BIOS o el firmware de los dispositivos.

En los análisis hechos previamente, se puede observar la gran diferencia de consumo de la memoria RAM con respecto a la memoria ROM. El consumo va progresivamente aumentando según vamos avanzando con los apartados: En el sistema diseñado sin ampliaciones el consumo es el más bajo, en la ampliación 1 aumenta debido a que se añadió un contador y en la ampliación 2 el consumo es el más alto (memoria RAM).

Gracias a este trabajo hemos podido profundizar más en muchos campos de la asignatura, especialmente en el tema de memorias.