Universidad Tecnológica de Panamá

Facultad de Ingeniería Eléctrica

Laboratorio de Microprocesadores

Proyecto final
Diseño de un Microprocesador de 4 bits

Fernando Guiraud
8-945-692

Profesor Elías Mendoza

Grupo: 4EE141

Semestre I 2022

I. INTRODUCCIÓN

El microprocesador hace parte del microcontrolador, el cual es prácticamente un computador en un chip, pues dispone de microprocesador (CPU), sensores y actuadores.

Las características principales del microprocesador son su UNIVERSALIDAD y su PROGRAMABILIDAD, lo que le dan el carácter de versatilidad, lo cual lo hace como el elemento inteligente o cerebro en diversas aplicaciones que van desde un computador personal hasta sistemas de encendido en vehículos, instrumentos médicos, etc.

La función de un microprocesador se define a través de un programa, que consta de una serie de ordenes o instrucciones relacionadas, ejecutadas secuencialmente (una a la vez) por el microprocesador y que pueden implicar operaciones lógicas o aritméticas. Las instrucciones se especifican a través de un código especial que constituye el lenguaje del microprocesador.

Esencialmente un microprocesador en su Hardware o parte física es un circuito de alta escala de integración compuesto de circuitos más simples cono flip flops, contadores, registros, decodificadores, ALU, etc. Por otro lado, los programas o software determinan el comportamiento del microprocesador adaptándolo a un campo especifico de aplicación.

II. OBJETIVO

- Diseñe el siguiente microprocesador básico de 4 bits, cuya arquitectura se muestra a continuación:

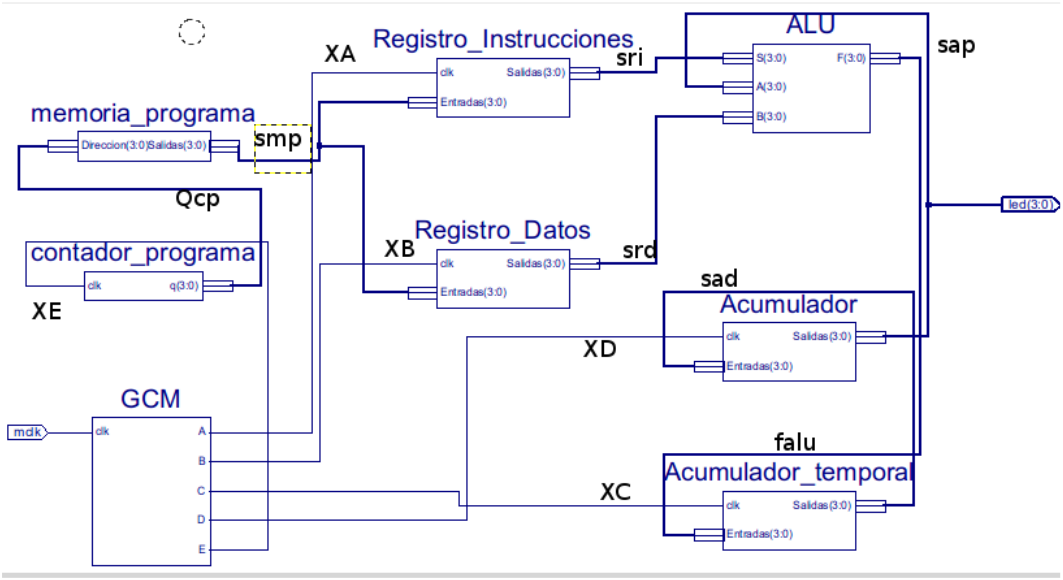


Ilustración 1: Diagrama de Bloques de un Microprocesador Básico de 4 bits

III. DESARROLLO

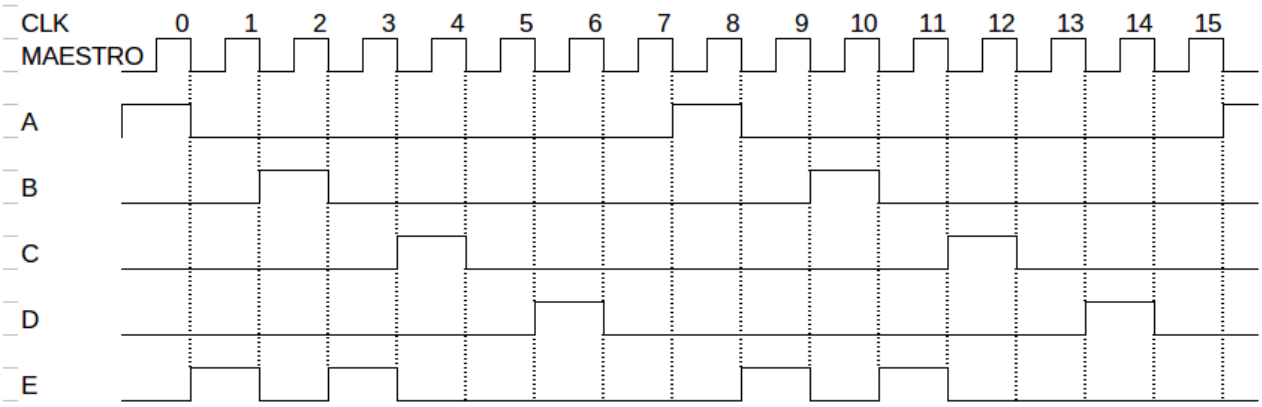
A continuación, se describen cada una de las partes que forman el microprocesador: GENERADOR DE CICLO DE MAQUINA (GCM). Este circuito es el que marca el paso del procesador; su función es sincronizar el sistema por medio de señales de control que van a todos los registros o sea al contador de programa para incrementarlo cuando se requiere, al registro de instrucciones, al registro de datos, al acumulador temporal y al acumulador principal. El GCM es alimentado por una señal de reloj proveniente de un oscilador de onda cuadrada.

Por el momento vamos a considerar la GCM como una caja negra con una entrada de reloj maestro proveniente del oscilador y cinco salidas de señal A, B, C, D y E, desfasadas una respecto a la otra como se observa en el siguiente diagrama de tiempos.

Luego tenemos el CONTADOR DEL PROGRAMA, el cual es un contador ascendente de 4 bits. Luego seguimos con la Memoria del Programa. En nuestro caso de acuerdo a lo deseado como ejercicio de aplicación para verificar en la tarjeta y teniendo en cuenta las funciones lógicas y aritméticas propias de la ALU se deberá generar el código correspondiente al siguiente algoritmo en Ensamblador:

LDA #12
ADDA #2
SUBA #5
ORA #3
EORA #5
ANDA #11
SUBA #10
ADDA #3
(# implica en forma inmediata)

El microprocesador diseñado e implementado responde a la arquitectura de von Neumann que es una familia de arquite3cturas de computadoras que utilizan el mismo dispositivo de almacenamiento tanto para las instrucciones como para los datos (a diferencia de la arquitectura Harvard que utiliza dos memorias: una para las instrucciones y otra memoria para los datos).



Estas señales de control sincronizan la operación de cada uno de los registros en la forma siguiente:

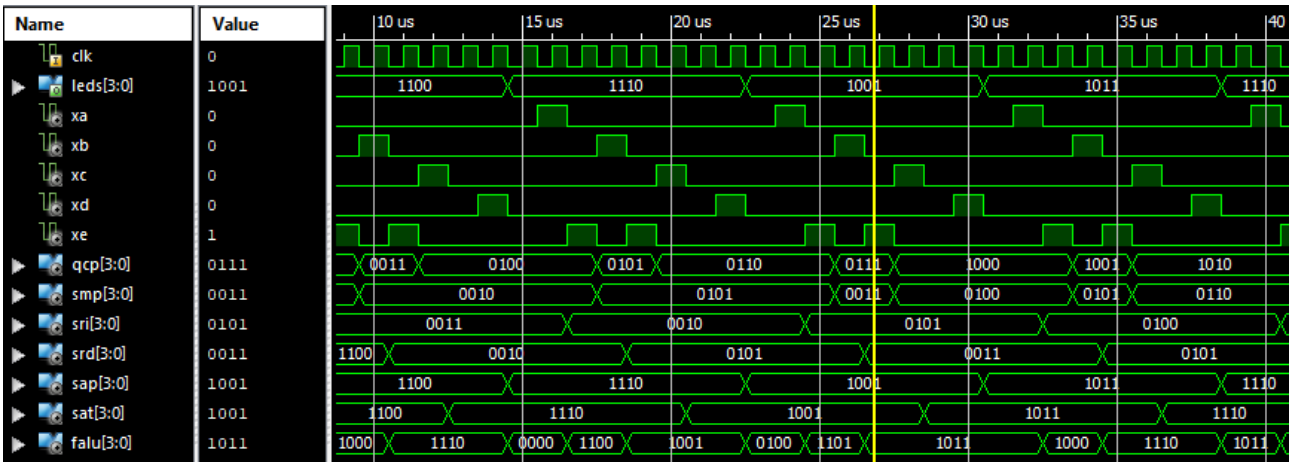
- El contador del programa se incrementa en cada flanco de bajada de la señal E.
- El registro de instrucciones retiene la información proveniente de la memoria cuando la señal A pasa de 1 a 0.
- El registro de datos retiene la información proveniente de la memoria cuando la señal B pasa de 1 a 0.
- El acumulador temporal guarda la función generada en la ALU cuando la señal C pasa de 1 a 0.
- El acumulador principal almacena la información que sale del acumulador temporal en el flanco de bajada de la señal D.

Además del BLOQUE GCM que genera las señales de control del microprocesador otro bloque muy importante en el funcionamiento del microprocesador es la Unidad Aritmético Lógica ALU, cuyo juego de instrucción se muestra a continuación:

S	Función
0000	B
0001	B - A
0010	A - B
0011	A + B
0100	A XOR B
0101	A OR B
0110	A AND B

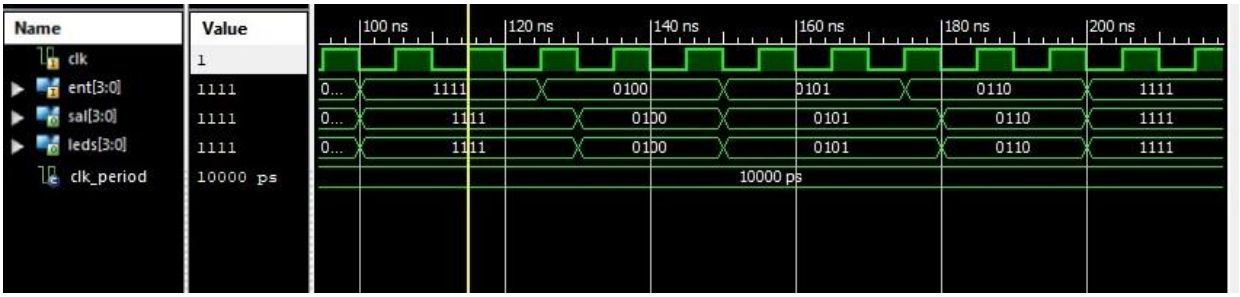
Microprocesador Código Principal

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

entity micro is
   Port (clk : in bit;
         Leds: out STD_LOGIC_VECTOR (3 downto 0):= (others => '0')
         );
end micro;
architecture Behavioral of micro is
   component acum
   Port (clk : in bit;
      ent: in STD_LOGIC_VECTOR (3 downto 0);
      sal:out STD_LOGIC_VECTOR (3 downto 0):= (others => '0');
      Leds:out STD_LOGIC_VECTOR (3 downto 0));
   end component;

   component ALU
   Port (A: in STD_LOGIC_VECTOR (3 downto 0);
         B: in STD_LOGIC_VECTOR (3 downto 0);
         S: in STD_LOGIC_VECTOR (3 downto 0);
         F: out STD_LOGIC_VECTOR (3 downto 0));
   end component;

   component cont
   Port (clk : in bit;
   Q: out STD_LOGIC_VECTOR (3 downto 0));
   end component;
   component GCM
   Port (clk : in bit; A: out bit; B: out bit;
         C: out bit; D: out bit; E: out bit);
   end component;

   component memo
   Port (addr : in STD_LOGIC_VECTOR (3 downto 0);
         sal : out STD_LOGIC_VECTOR (3 downto 0));
   end component;

   component reg
   Port (clk : in bit;
         ent: in STD_LOGIC_VECTOR (3 downto 0);
         sal:out STD_LOGIC_VECTOR (3 downto 0));
   end component;

signal XA,XB,XC,XD,XE: bit;
signal Qcp,Smp,Sri,Srd,Sap,Sat,Falu: STD_LOGIC_VECTOR (3 downto 0);
begin
   Inst_acum: acum port map (XD,Sat,Sap,Leds);
   Inst_ALU: ALU port map (Sap,Srd,Sri,Falu);
   Inst_cont: cont port map (XE,Qcp);
   Inst_GCM: GCM port map(CLK,XA,XB,XC,XD,XE);
   Inst_memo: memo port map (Qcp,Smp);
   Inst_reg_INS: reg port map (XA,Smp,Sri);
   Inst_reg_DAT: reg port map (XB,Smp,Srd);
   Inst_acum_TEMP: reg port map (XC,Falu,Sat);
end Behavioral;
```



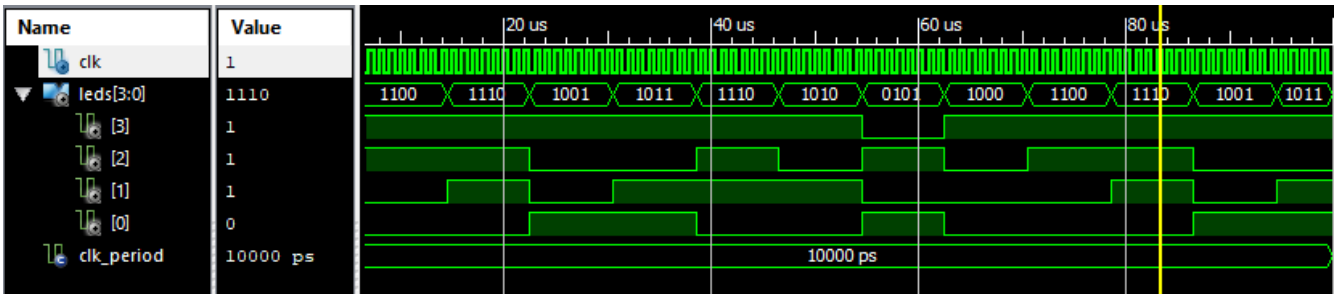
Acumulador

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity acum is
   Port (clk : in bit;
         ent: in STD_LOGIC_VECTOR (3 downto 0);
         sal:out STD_LOGIC_VECTOR (3 downto 0):= (others => '0');
         Leds:out STD_LOGIC_VECTOR (3 downto 0));
end acum;
architecture Behavioral of acum is
begin
 process(clk)
 begin
    if(clk'event and clk='0') then
   sal <= ent;
   Leds <= ent;
    end if;
 end process;
end Behavioral;
```



Contador

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity cont is
   port ( clk : in bit;
          Q : out STD_LOGIC_VECTOR (3 downto 0));
end cont;
architecture Behavioral of cont is
signal count: STD_LOGIC_VECTOR (3 downto 0):="0000";
begin
process(clk)
   begin
      if clk='0' and clk'event then
         count <= count + 1;
      end if;
end process;
Q <= count;
end Behavioral;
```

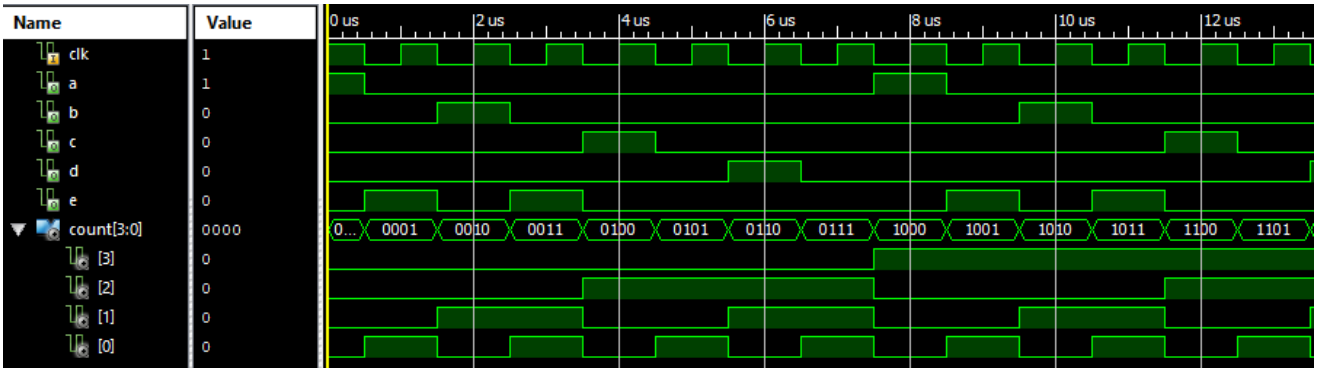


Registro

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity reg is
   Port (clk : in bit;
         ent: in STD_LOGIC_VECTOR (3 downto 0);
         sal:out STD_LOGIC_VECTOR (3 downto 0));
end reg;
architecture Behavioral of reg is
begin
 process(clk,ent)
 begin
    if(clk'event and clk='0') then
      sal <= ent;
    end if;
 end process;
end Behavioral;
```

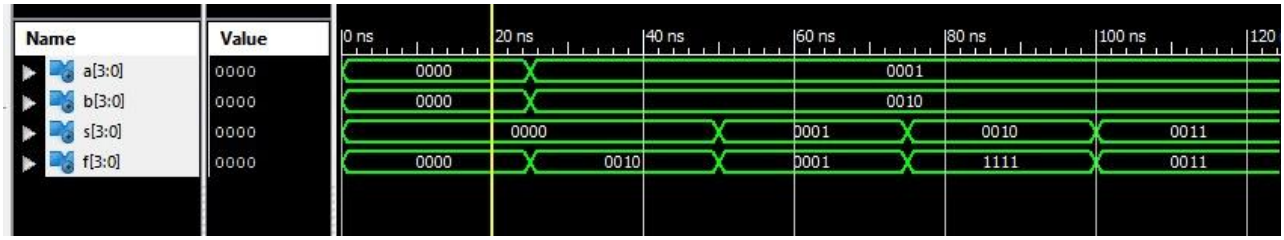
GCM

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity GCM is
   Port (clk : in bit;
            A: out bit;
            B: out bit;
            C: out bit;
            D: out bit;
            E: out bit);
end GCM;
architecture Behavioral of GCM is
signal count: STD_LOGIC_VECTOR (3 downto 0):= (others => '0');
begin
 process(clk,count)
   begin
      if(clk'event and clk='0') then
      count <= count + 1;
      end if;
 end process;
A <= '1' when (count="0000") else
     '1' when (count="1000") else
     '0';
B <= '1' when (count="0010") else
     '1' when (count="1010") else
     '0';
C <= '1' when (count="0100") else
     '1' when (count="1100") else
     '0';
D <= '1' when (count="0110") else
     '1' when (count="1110") else
     '0';
E <= '1' when (count="0001") else
     '1' when (count="1001") else
     '1' when (count="1011") else
     '1' when (count="0011") else
     '0';
end Behavioral;
```



ALU

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity ALU is
    Port (A: in STD_LOGIC_VECTOR (3 downto 0);
          B: in STD_LOGIC_VECTOR (3 downto 0);
          S: in STD_LOGIC_VECTOR (3 downto 0);
          F: out STD_LOGIC_VECTOR (3 downto 0));
end ALU;
architecture Behavioral of ALU is
begin
    process(A,B,S)
    begin
        case S is
            when "0000" =>
                F <= B;
            when "0001" =>
                F <= B - A;
            when "0010" =>
                F <= A - B;
            when "0011" =>
                F <= A + B;
            when "0100" =>
                F <= A xor B;
            when "0101" =>
                F <= A or B;
            when "0110" =>
                F <= A and B;
            when others =>
                F <= "1111";
        end case;
    end process;
end Behavioral;
```



IV. CONCLUSIÓN

Se puede demostrar que con la ayuda de las herramientas de síntesis se logra una gran productividad en el diseño de sistemas digitales, ahorrando costos y tiempo de desarrollo. Si bien el diseño no llega a nivel de máscaras para el proceso de fabricación de chips, se logra tener lo implementado a nivel de hardware ya sea en un CPLD o como en un FPGA.

Los sistemas de microprocesador forman el corazón de los dispositivos de una computadora, leen y actúan según las instrucciones que le da un programador, tienen tres buses: dirección, datos y control, y operan según las instrucciones que se les dan en forma de código de máquina. El código de máquina es generado por un lenguaje de nivel superior como C o lenguaje ensamblador. Un diseñador de sistemas debe considerar el uso de un microprocesador en lugar de circuitos lógicos siempre que una aplicación implique realizar cálculos, tomar decisiones basadas en estímulos externos y mantener la memoria de eventos pasados.

V. REFERENCIAS

D. L. Perry, VHDL. New York: McGraw-Hill, 1991