Actividad 1 - Unidad 1

Itzel Berenice Cabrera Vazquez

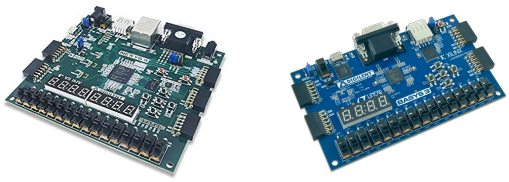
I. Introducción

A. Disposiivos Logicos Programables

Los dispositivos lógicos programables [1] surgen gracias a dos diseños: el diseño tradicional de sistemas digitales (bajo costo y tiempo de desarrollo corto, aunque con arquitecturas inflexibles y recursos poco versátiles) y al diseño de aplicación específica (ASIC). Estos dispositivos tienen la característica de tener diseños concentrados en pequeñas áreas y para ello se requiere el uso de los lenguajes de descripción de hardware[2]. De esta forma es como el desarrollo de sistemas digitales se transforma a una descripción de alto nivel de los componentes que lo conforman, luego esta descripción configura apropiadamente los recursos del dispositivo, para que finalmente se obtenga un sistema funcional. [1]

Es por ello, que es fundamental para un Ingeniero en Telemática aprender el funcionamiento de los PLDs, los más usados y los lenguajes HDL (actualmente el más usado en la industria es el VHDL).

Existen clasificaciones de los PLDs, entre ellos se encuentran los: PLAs, PROMs, PALs, GALs, CPLDs y FPGAs. Las FPGA son matrices de puertas eléctricamente programables que contienen múltiples niveles de lógica. Se caracterizan por altas densidades de elementos, alto rendimiento, un gran número de entradas y salidas disponibles por el usuario y un esquema de interconexión flexible.



(a) Digilent Nexys4 Artix-7 FPGA Trainer Board **(b)** Digilent Basys3 Artix-7 FPGA

Fig. 1: Algunos FPGAs boards

B. Vivado Xilinx

Vivado Design Suite es un software producido por Xilinx para la simulación, síntesis, análisis e implementación de los diseños de HDL. Para este último, la implementación, se requiere la documentación del board que se esté usando; en esta actividad usaré el board Nexys4[3].

[1] PLDs, Programmable Logic Devices

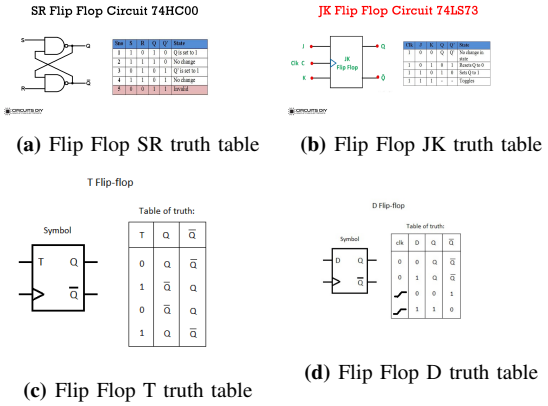
[2] Hardware Description Language

[3] https://digilent.com/reference/programmable-logic/nexys-4-ddr/reference-manual

C. Flip Flops

Los flip flps son dispositivos de almacenamiento, almacenan 1 bit, es por ello que tambien se le conoce como un multivibrador biestable.

A continuación se presentan los tipos de flip flops:



(a) Flip Flop SR truth table **(b)** Flip Flop JK truth table

(c) Flip Flop T truth table **(d)** Flip Flop D truth table

Fig. 2: Clasificación de Flip Flops, diagrama y tabla de verdad

D. Display de 7 segmentos

El display 7 Segmentos es un dispositivo opto-electrónico que permite visualizar números del 0 al 9. Existen dos tipos de display, de cátodo común y de ánodo común. Surge en los 70's y debido a su facilidad de uso, mantenimiento y costo, actualmente se sigue usando con mucha frecuencia.

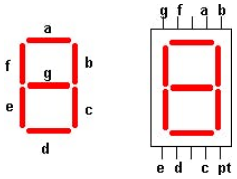


Fig. 3: Display de 7 segmentos

II. Objetivos

Realizar un proyecto[4] en Vivado, de tal forma que puedan generarse bloques de VHDL que tengan jerarquía entre sí. Para ello se deberá usar descripciones de comportamiento, de flujo de datos y de estructura. A medida de lo posible, realizar la síntesis, simulación e implementación del proyecto.

III. Diseño de solución

Primeramente, se muestra el diagrama de estados en la fig. 5.

[4] contador de 3 bits síncrono mediante el uso de flip flops, y mediante el uso de un decodificador de 7 segmentos, mostrar el numero decimal en un display

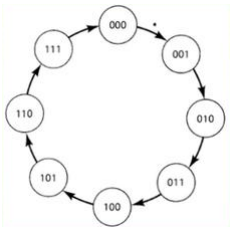


Fig. 4: Diagrama de estados

A continuación, se muestra en un diagrama de la solución propuesta.

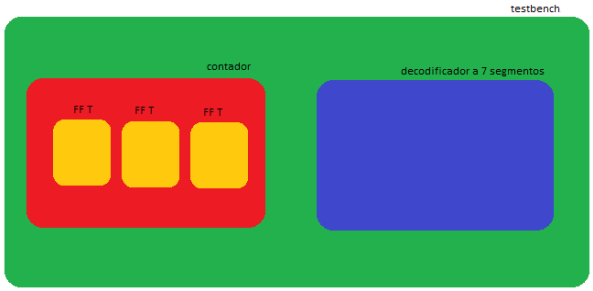


Fig. 5: Jerarquía de programas vhdl

IV. Desarrollo y Resultados

1) Programar en VHDL un flip flop tipo T. Analicé que para construir un contador de 3 bits debía usar 3 flip flops JK, con las entradas JK de cada flip flop iguales, esto es lo mismo que usar un flip flop tipo T.6

```vhdl
--flip flop tipo T; clear asíncrono y preset
libraryoIEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ff_T is Port (
    clear,preset,clk,T: in std_logic;
    Q: inout std_logic:='0');
end ff_T;

architecture arq_ffT of ff_T is
begin
process (clk,clear,preset) begin
    if clear = '0' then Q<='0';
    elsif(clk'event and clk = '1')then
        if preset = '0' then Q<='1';
        else Q<= T xor Q;
        end if;
    end if;
end process;
end arq_ffT;
```

Fig. 6: Código del flip flop T en VHDL

2) Programar en VHDL un contador de tres bits usando los flip flops programados en inciso anterior. Para que el contador fuera sincrónico, las entradas de reloj de cada flip flop debía ser la misma. Además de ello, la entrada T0 (entrada T del primer flip flop) debía estar siempre en un nivel alto, la entrada T1(entrada T del segundo flip flop) debía ser Q0 (la salida del primer flip flop) y finalmente, la entrada T2 (entrada del tercer flip flop) tenía que ser 'Q0 and Q1'. Por otra parte, el 'reset' y 'set' del contador, ambos sincrónicos, se activan a un nivel bajo.

```vhdl
--contador de 3 bits con preset síncrono y reset síncrono
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity contador is Port (
    clk,preset,reset: in std_logic ;
    L0,L1,L2: out std_logic);
end contador;

architecture arqCont of contador is
    signal Q0,Q1,Q2:std_logic;
    signal T0: std_logic:='1';
    signal T1,T2:std_logic;
begin
    F0: entity work.ff_T PORT MAP(
        Q=>Q0,
        T=>T0,
        clk=>clk,
        preset=>'1',
        clear=>'1'
    );
    F1: entity work.ff_T PORT MAP(
        Q=>Q1,
        T=>T1,
        clk=>clk,
        preset=>'1',
        clear=>'1'
    );
    F2: entity work.ff_T PORT MAP(
        Q=>Q2,
        T=>T2,
        clk=>clk,
        preset=>'1',
        clear=>'1'
    );

    process(preset,reset,clk)begin
        if(clk'event and clk='1')then
            if preset='0' then T0<= not Q0;
            elsif reset ='0' then T0 <= Q0;
            end if;
        end if;
    end process;

    L0 <= Q0;
    L1 <= Q1;
    L2 <= Q2;
    T1 <=Q0;
    T2<=Q0 and Q1;
end arqCont;
```

Fig. 7: Código del contador de 3 bits en VHDL

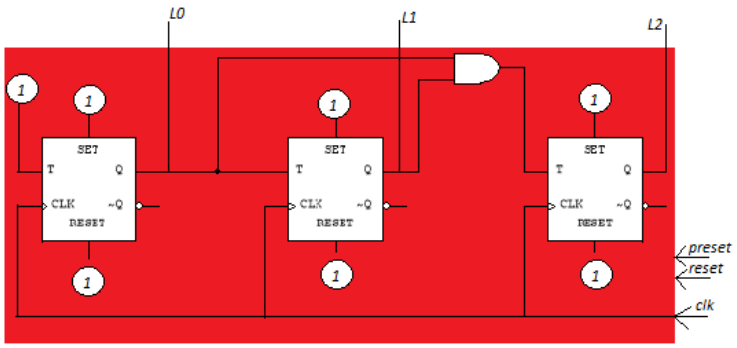


Fig. 8: contador

3) Programar en VHDL un decodificador a 7 segmentos. Para ello hay que tener en cuenta que la entrada al decodificador son 3 bits por separado, es por ello que para usar el 'case-when', concateno estos tres bits de entrada.

Además de ello, hay que considerar que en esta FPGA, los segmentos se activan a niveles bajos. Finalmente, no hay que olvidar el punto del display, es por ello que se consideran 8 bits en la salida.

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

entity dec7seg is port (
 b0,b1,b2: in std_logic;
 d: out std_logic_vector (7 downto 0); --son 8 bits, incluyendo el punto
 AN: out std_logic);
end dec7seg;

architecture arq_dec of dec7seg is
begin
    process(b0,b1,b2)
        variable bcat : std_logic_vector(2 downto 0);
    begin
        bcat:= b0&b1&b2;
        case bcat is
            when "000"=> d<='0'&"0000001";--0
            when "001"=> d<='0'&"1001111";--1
            when "010"=> d<='0'&"0010010";--2
            when "011"=> d<='0'&"0000110";--3
            when "100"=> d<='0'&"1001100";--4
            when "101"=> d<='0'&"0100100";--5
            when "110"=> d<='0'&"0100000";--6
            when others => d<='0'&"0001111"; --7
        end case;
    end process;

    AN<='0';

end arq_dec;
```

Fig. 9: Código del decodificador a 7 segmentos en VHDL

4) Test Bench. Genero mi banco de pruebas o banco de estímulos, recordando que en este archivo no existen puertos, sólo señales. Fig.10

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity testbench is
end testbench;

architecture arq_TB of testbench is
    signal CLK:std_logic:='0';
    signal L0,L1,L2: std_logic;
    signal RESET,PRESET: std_logic:='1';
    signal ENABLE: std_logic:='1';
    signal D: std_logic_vector(7 downto 0);
    signal AN: std_logic;
begin
    CONT: entity work.contador PORT MAP(
        clk=>CLK,
        preset=>PRESET,
        reset=>RESET,
        L0=>L0,
        L1=>L1,
        L2=>L2
        --enable=>ENABLE
    );

    DECOD: entity work.dec7seg PORT MAP(
        b0=>L0,
        b1=>L1,
        b2=>L2,
        d=>D,
        AN=>AN
    );

    process
     begin
         if ENABLE = '1' then
             wait for 10 ns;
             CLOCK_LOOP : loop
                 CLK <= '0';
                    wait for 20ns;
                 CLK <= '1';
                    wait for 20ns;
             end loop CLOCK_LOOP;
         else CLK <= '0';
         end if;
     end process;
end arq_TB;= IO_L18N_T2_34,                    Sch name = AN0
set_property PACKAGE_PIN N6 [get_ports {AN}]
```

Fig. 10: Codificación de mi banco de pruebas

5) Simulación. Muestro los resultados de simulación, se puede ver como funciona de forma correcta.Fig.11
6) Síntesis. Realizado este paso se pueden obtener varios datos útiles como lo son: las esquematización, la lista de elementos, el diseño del dispositivo, entre otras cosas.Fig.12
7) Implementación. Mediante el uso de un archivo .xdc, realizar la implementación de los leds que muestran el número binario y el display que muestra el número decimal. Fig.13. Además de ello, el software

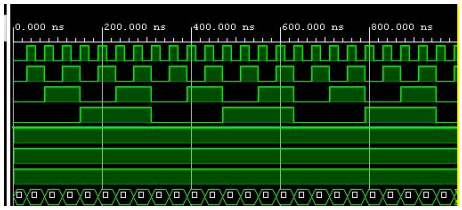
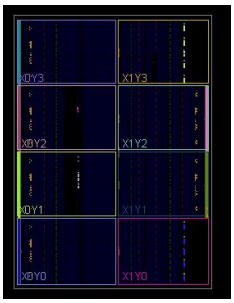
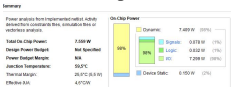


Fig. 11: Simulación de mi test bench

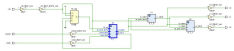


(a) El dispositivo FPGA

(b) La lista de elementos en nuestro proyecto



(c) Análisis del voltaje en nuestro proyecto



(d) La esquematización de nuestro proyecto

Fig. 12: Síntestis del proyecto, información entregada por el software

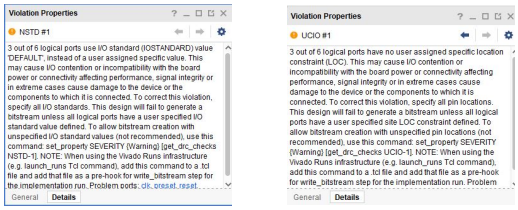
me mandó mensajes de advertencia del diseño de mi proyecto, indicándome que estos podrían producir un mal rendimiento.Fig.14

```
##Para encender mis tres leds
Bank = 34, Pin name = IO_L24N_T3_34,                    Sch name = LED0
set_property PACKAGE_PIN T8 [get_ports {L0}]
    set_property IOSTANDARD LVCMOS33 [get_ports {L0}]
Bank = 34, Pin name = IO_L21N_T3_DQS_34,                Sch name = LED1
set_property PACKAGE_PIN V9 [get_ports {L1}]
    set_property IOSTANDARD LVCMOS33 [get_ports {L1}]
Bank = 34, Pin name = IO_L24P_T3_34,                    Sch name = LED2
set_property PACKAGE_PIN R8 [get_ports {L2}]
    set_property IOSTANDARD LVCMOS33 [get_ports {L2}]
Bank = 34, Pin name = IO_L23N_T3_34,                    Sch name = LED3
set_property PACKAGE_PIN T6 [get_ports {L3}]
    set_property IOSTANDARD LVCMOS33 [get_ports {L3}]

##Para el display

Bank = 34, Pin name = IO_L2N_T0_34,                     Sch name = CA
set_property PACKAGE_PIN L3 [get_ports {D[0]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[0]}]
Bank = 34, Pin name = IO_L3N_T0_DQS_34,                 Sch name = CB
set_property PACKAGE_PIN N1 [get_ports {D[1]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[1]}]
Bank = 34, Pin name = IO_L6N_T0_VREF_34,                Sch name = CC
set_property PACKAGE_PIN L5 [get_ports {D[2]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[2]}]
Bank = 34, Pin name = IO_L5N_T0_34,                     Sch name = CD
set_property PACKAGE_PIN L4 [get_ports {D[3]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[3]}]
Bank = 34, Pin name = IO_L2P_T0_34,                     Sch name = CE
set_property PACKAGE_PIN K3 [get_ports {D[4]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[4]}]
Bank = 34, Pin name = IO_L4N_T0_34,                     Sch name = CF
set_property PACKAGE_PIN M2 [get_ports {D[5]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[5]}]
Bank = 34, Pin name = IO_L6P_T0_34,                     Sch name = CG
set_property PACKAGE_PIN L6 [get_ports {D[6]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[6]}]

Bank = 34, Pin name = IO_L16P_T2_34,                    Sch name = DP
set_property PACKAGE_PIN M4 [get_ports {D[7]}]
    set_property IOSTANDARD LVCMOS33 [get_ports {D[7]}]

Bank = 34, Pin name = IO_L18N_T2_34,                    Sch name = AN0
set_property PACKAGE_PIN N6 [get_ports {AN}]
```

Fig. 13: Modificación del archivo constraint .xdc

V. Conclusiones

La importancia de los dispositivos lógicos programables en el mundo digital es sumamente importante y como futuros Ingenieros Telemáticos es fundamental que aprendamos cómo funcionan y así mismo, practicar con estos. Lamentablemente,



(a) Violación 1 **(b)** Violación 2

Fig. 14: Implementación del proyecto, información entregada por el software

por el momento, el último paso posible a realizar es el de la implementación.

Por otra parte, me percaté de la importancia de la descripción de estructura en VHDL, esta te ayuda a generar ”segmentos jerárquicos”, y de esta forma le da estructura al proyecto que se este realizando.

Finalmente, esta práctica me ayudo a recordar los fundamentos del diseño digital y repasar el lenguaje VHDL.

REFERENCES

[1] Larrotta, D. M. B., and Aranguren, A. J. P. (2004). Diseño VHDL de sistemas digitales sobre dispositivos lógicos programables FPGAS. Umbral Científico, (4), 37-49.