24/9/2013

ÁREA DE TECNOLOGÍA ELECTRÓNICA UBU

GUÍA PRÁCTICA DEL SOFTWARE ACTIVE-HDL .

Manual de Inicio | Miguel Ángel Lozano Pérez

Contenido

1.1 GUÍA DE SOFTWARE

Mediante esta guía se pretende obtener un aprendizaje rápido, para el diseño de dispositivos, mediante la programación del diseño de tres formas diferentes:

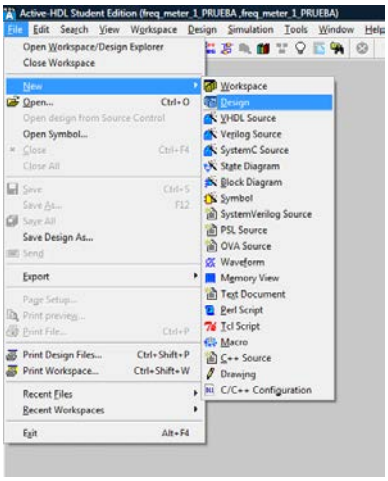
- Programación en lenguaje de descripción hardware, VHDL.
- Programación mediante la realización de máquinas de estado finitos.
- Mediante bloques.

La implementación de los diseños también se puede desarrollar mediante la combinación de cualquiera de

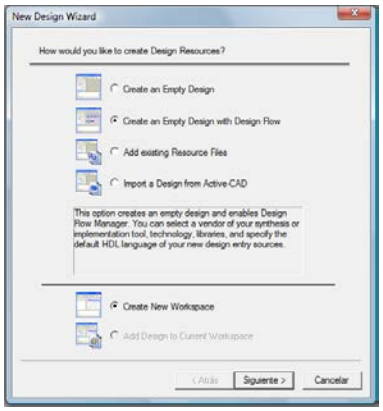
1.1.1 Primer paso: crear un diseño nuevo.

Desde el menú de inicio o mediante un acceso directo se abre

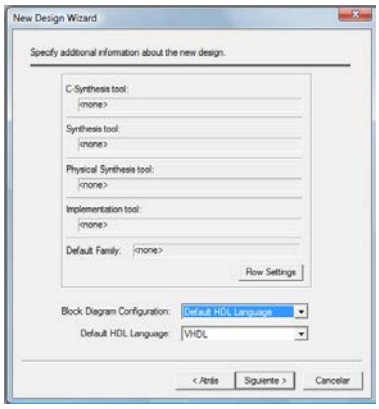
File->Nuevo->Diseño



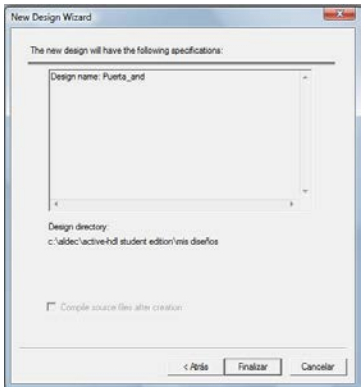
Mediante el asistente, se elige Create un Empty Desing with Desing Flow.

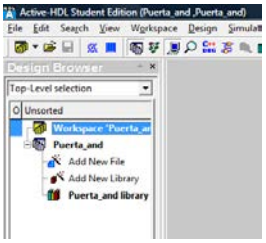


Se crea a partir de un diseño vacio. Pulsar siguiente y poner un nombre al diseño, por ejemplo puerta_and.



Se pulsa siguiente y finalizar

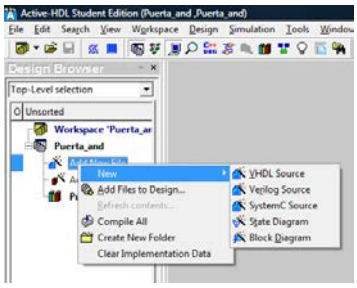




Se crea el navegador del diseño.

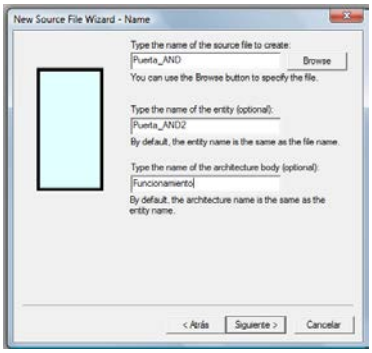
1.1.2 Siguiente paso: añadir un fichero en VHDL.

Se elige desde el navegador y añadimos un fichero vhdl como fuente.



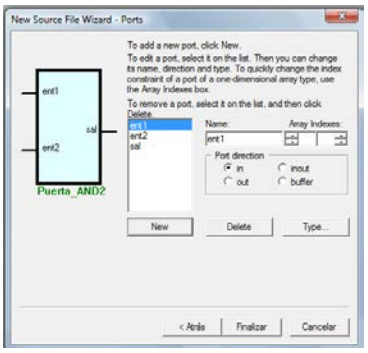
Aparece el asistente. Se pulsa siguiente

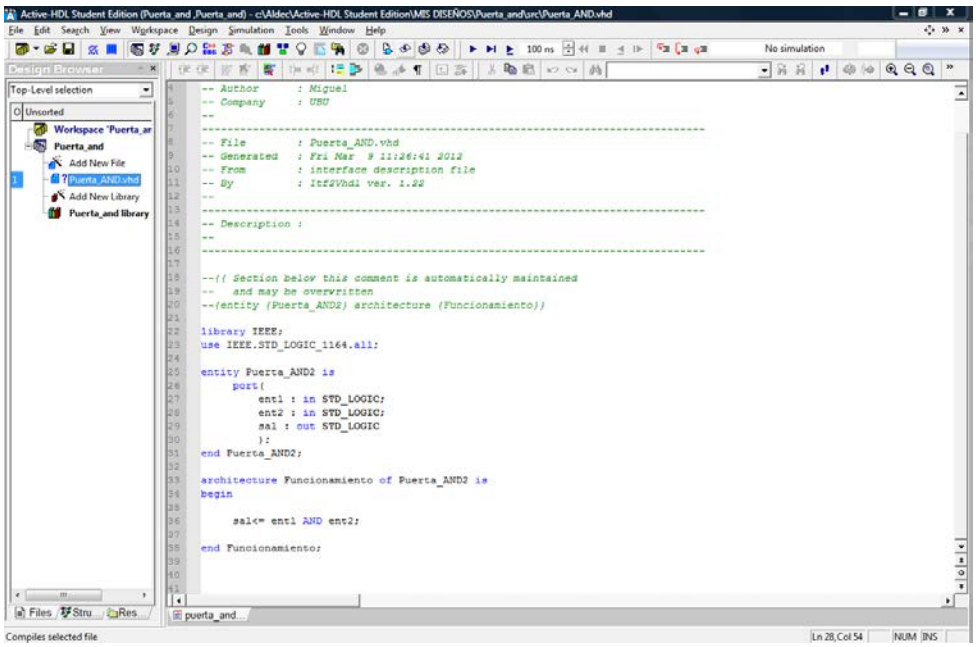
Se dispone un nombre para el fichero y de forma opcional el de la entidad y la arquitectura. Si éstos últimos no se pone se realiza por defecto.



Se pulsa siguiente y se van añadiendo los puertos de entrada y salida que posteriormente se integrarán en la entidad.

Se pulsa finalizar y acaba el asistente con la obtención de un fichero vhdl, preparado para ser confeccionado.





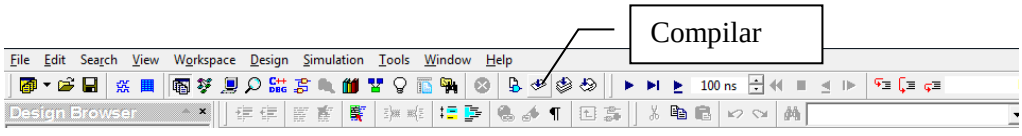
1.1.3 Siguiente paso: Programación.

A partir de esta plantilla se cumplimenta la arquitectura, En esta caso se elige la descripción de flujo de datos o RTL.

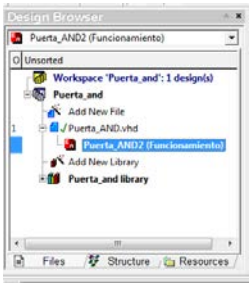
```
entity Puerta_AND2 is
port(        ent1 : in STD_LOGIC;
ent2 : in STD_LOGIC;
sal : out STD_LOGIC );
end Puerta_AND2;
architecture Funcionamiento of Puerta_AND2 is
begin
            sal<= ent1 AND ent2 AFTER 4ns;
end Funcionamiento;
```

Se programa la arquitectura.

Se compila para determinar errores.



En la ventana de consola se visualiza si hay o no errores.



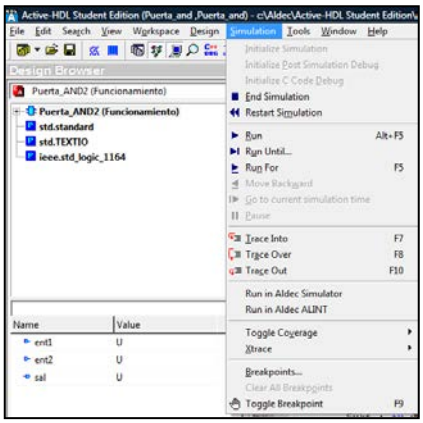
Despues de la compilación desaparece una interrogación para indicar que ha sido realizada.

Aparece un bloque en rojo indicando entidad y arquitectura compilada.

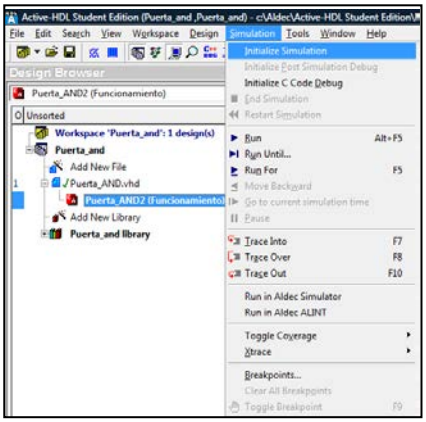
1.1.4 Siguiente paso: SIMULACIÓN.

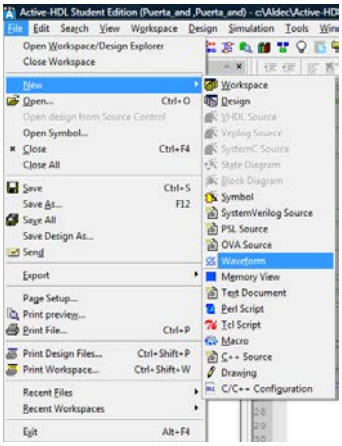
Del menú simulación, se elige inicializar la simulación.

De esta forma aparecen nuevas opciones para realizar la simulación.



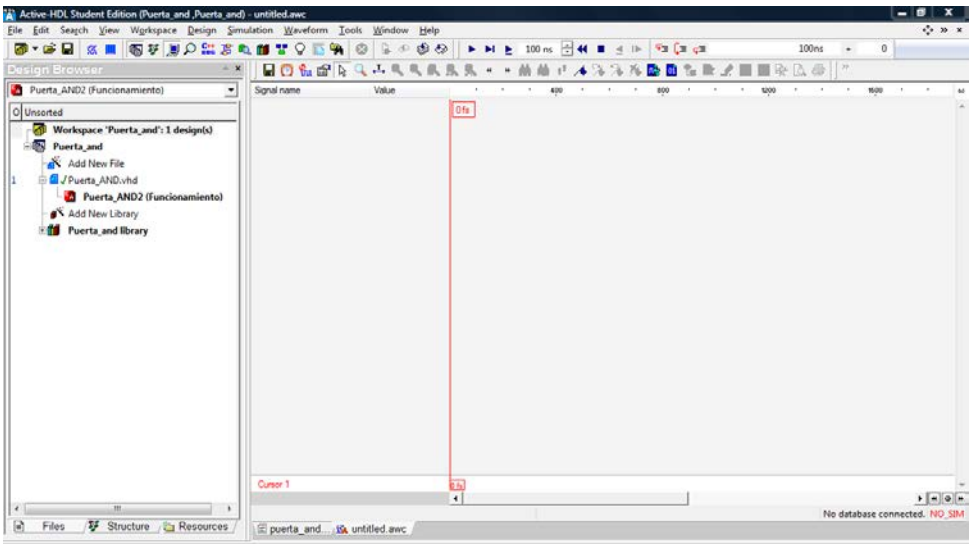
Estas mismas opciones aparecen en la barra de menú.



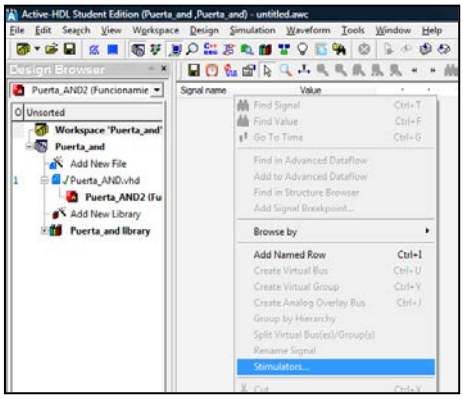


Se introduce una nueva ventana para visualizar las formas de onda, dentro del menú elegir: file -> new-> waveform

Aparece una nueva ventana para realizar la simulación

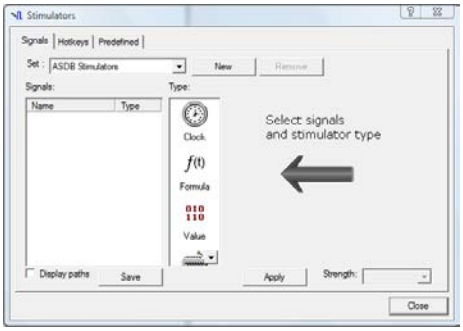


Sobre esta ventana hacer clic con el botón derecho del ratón y elegir la opción Stimulators.



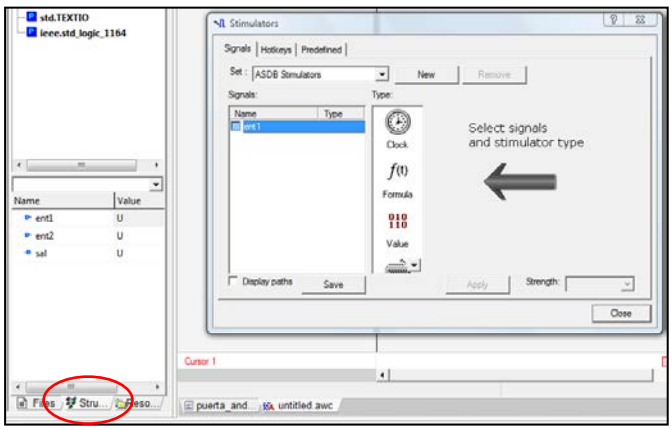
Stimulators: Se eligen las señales que se desean estimular y como.

Aparece una ventana para poder definir la excitación de las señales de entrada.



Se definen mediante dos teclas, (type: elegir dibujo de teclado) que al pulsalas, cambian de estado de '0' a '1' y viceversa.

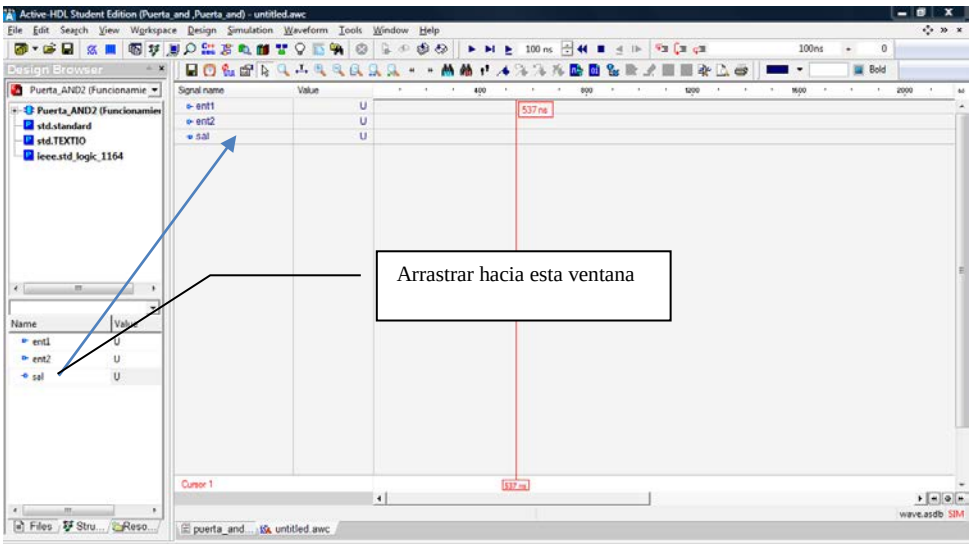
Para ello, dentro del navegador (abajo), se elige la ficha estructura, de esta forma aparecen las señales de entrada y salida entre otras si las hubiere de nuestro diseño.



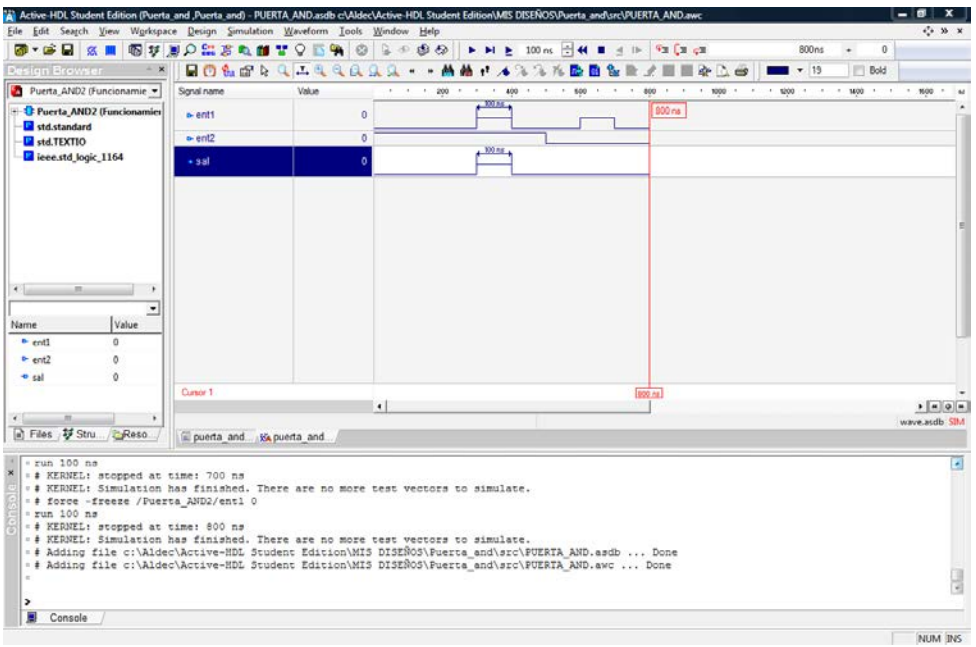
En el stimulator se han elegido las teclas A y B para excitar a las entradas ent1 y ent2 respectivamente, tal y como se aprecia en la figura.

Cada vez que se define una señal se ha de aplicar y elegir una nueva señal del navegador para definir el tipo de estimulo. Una vez creadas todas se cierra la ventada stimulus.

Posteriormente arrastrar las señales que se desean visualizar hacia la ventana de las formas de onda (waveform) desde el navegador.



Al variar las diferentes señales (con las teclas A y B) se consiguen variar los valores de las señales ***ent1*** y ***ent2*** y ver el cambio producido en la salida ***sal***, tal y como se aprecia en la figura.



PROBLEMA RESUELTO 1.1-1

Diseñar y simular con el programa Active-*HDL Student Edition* una máquina de estados.

Objetivos: Conocer la metodología de diseño SECUENCIAL en un dispositivo PAL.

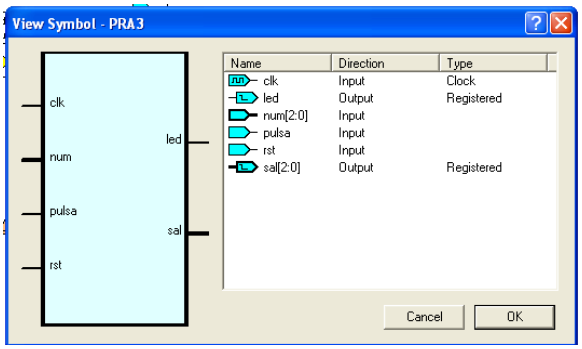
Conocer los procedimientos de programación de estos dispositivos.

Diseño del fichero VHDL con Active HDL

Diseño del dispositivo en VHDL con arquitectura serie o algorítmica.

Diseñar una máquina de estados (con ACTIVE-FSM), con las señales indicadas.

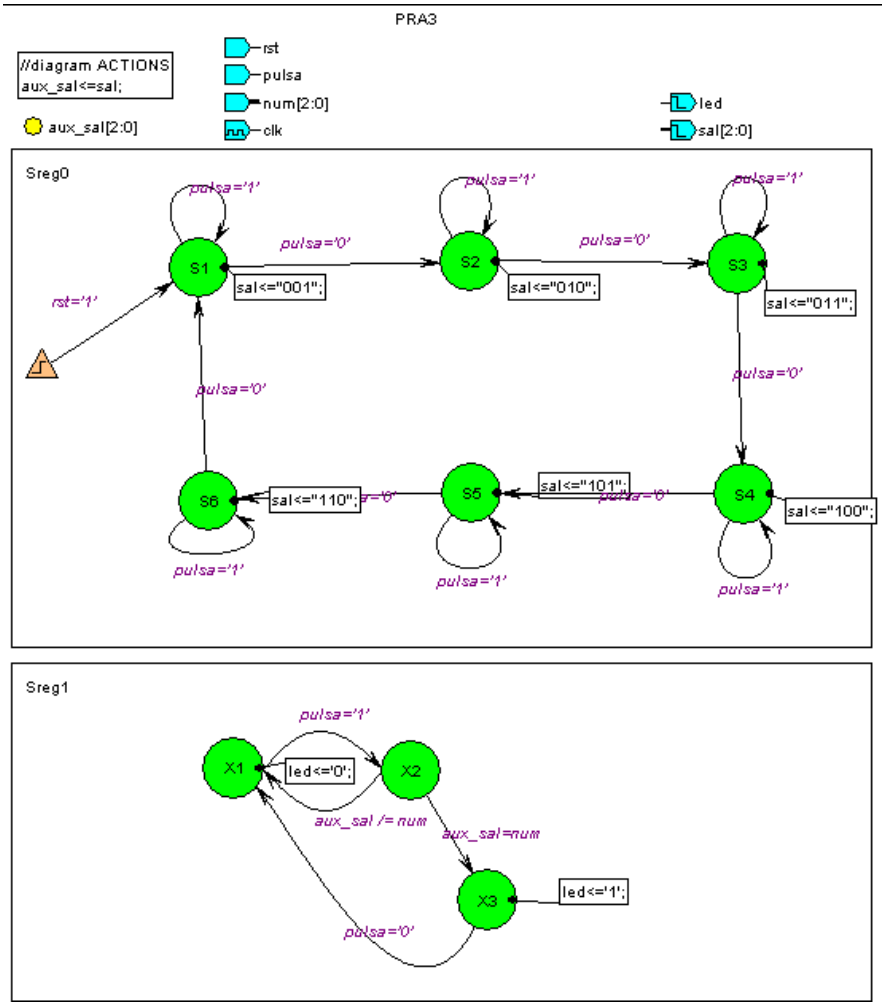
Se trata de disponer un número cualquiera num de valor entre 1 y 6. (ej 6D “110”B). Si éste coincide con el valor o estado de un dado (contador de 1 a 6) que evoluciona con la señal de reloj, activa la señal de salida led=’1’ (coincidencia).



El funcionamiento es el siguiente con la señal p=’0’ el contador evoluciona de 1 a 6.

Se pulsa p, y el contador se para en ese estado, si con el flanco de subida de ésta señal coincide el número dispuesto con el estado del contador se activa la señal de salida led=’1’

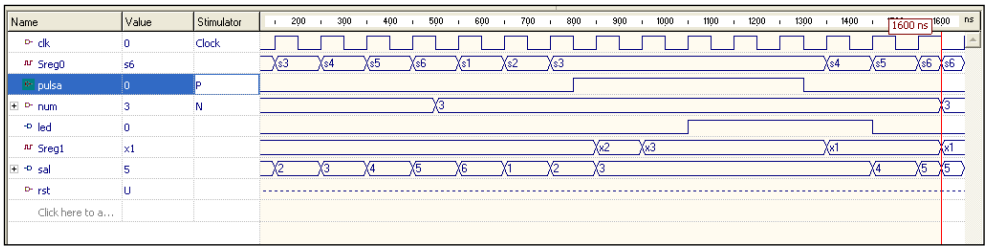
Se pide realizar el diseño como máquina de estado y simularlo.




```
library IEEE;
use IEEE.std_logic_1164.all;

entity pra3 is
  port (clk: in STD_LOGIC;
        num: in STD_LOGIC_VECTOR (2 downto 0);
        pulsa: in STD_LOGIC;
        rst: in STD_LOGIC;
        led: out STD_LOGIC;
        sal: out STD_LOGIC_VECTOR (2 downto 0));
end;

architecture pra3_arch of pra3 is

--diagram signal declarations
signal aux_sal: STD_LOGIC_VECTOR (2 downto 0);

-- SYMBOLIC ENCODED state machine: Sreg0
type Sreg0_type is (S1, S2, S3, S4, S5, S6);
signal Sreg0: Sreg0_type;

-- SYMBOLIC ENCODED state machine: Sreg1
type Sreg1_type is (X1, X2, X3);
signal Sreg1: Sreg1_type;
```

```
begin
--concurrent signal assignments
--diagram ACTIONS
aux_sal<=sal;


Sreg0_machine: process (clk)

begin

if clk'event and clk = '1' then
        if rst='1' then
                Sreg0 <= S1;
                sal<="001";
        else
        case Sreg0 is
                when S1 =>
                        sal<="001";
                        if pulsa='0' then
                                Sreg0 <= S2;
                        elsif pulsa='1' then
                                Sreg0 <= S1;
                        end if;
                when S2 =>
                        sal<="010";
                        if pulsa='0' then
                                Sreg0 <= S3;
                        elsif pulsa='1' then
                                Sreg0 <= S2;
                        end if;
                when S3 =>
                        sal<="011";
                        if pulsa='0' then
                                Sreg0 <= S4;
                        elsif pulsa='1' then
                                Sreg0 <= S3;
                        end if;
                when S4 =>
                        sal<="100";
                        if pulsa='0' then
                                Sreg0 <= S5;
                        elsif pulsa='1' then
                                Sreg0 <= S4;
                        end if;
                when S5 =>
                        sal<="101";
                        if pulsa='0' then
                                Sreg0 <= S6;
                        elsif pulsa='1' then
                                Sreg0 <= S5;
                        end if;
                when S6 =>
                        sal<="110";
                        if pulsa='0' then
                                Sreg0 <= S1;
                        elsif pulsa='1' then
                                Sreg0 <= S6;
                        end if;
                when others =>
                        null;
```

```
            end case;
            end if;
end if;
end process;



Sreg1_machine: process (clk)

begin

if clk'event and clk = '1' then
            case Sreg1 is
                        when X1 =>
                                    led<='0';
                                    if pulsa='1' then
                                                Sreg1 <= X2;
                                    end if;
                        when X2 =>
                                    if aux_sal /= num then
                                                Sreg1 <= X1;
                                    elsif aux_sal=num then
                                                Sreg1 <= X3;
                                    end if;
                        when X3 =>
                                    led<='1';
                                    if pulsa='0' then
                                                Sreg1 <= X1;
                                    end if;
                        when others =>
                                    null;
            end case;
end if;
end process;

end pra3_arch;
```



PROBLEMA RESUELTO 1.1-2

Diseñar y simular con el programa Active-*HDL Student Edition* una puerta lógica and y un contador bcd..

Realizar y simular un sistema digital, que dispone de una señal de entrada, denominada *GATE*, y un vector de salida de 4 dígitos en BCD, para visualizar el tiempo que ha estado activa a '1' la señal de entrada *GATE*.

Resolver mediante vhdl.

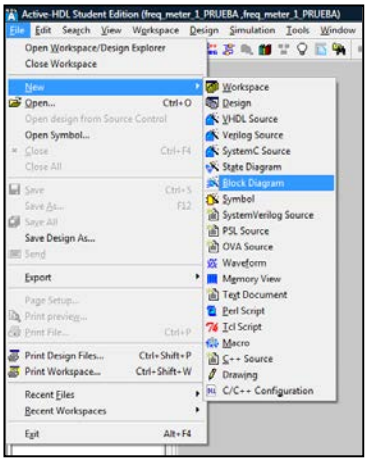
Pasos:

Diseñar una puerta and y un contador BCD.

Integrar mediante bloques. Utilizando los diseños anteriores, realizar el esquema de bloques necesario.

Solución:

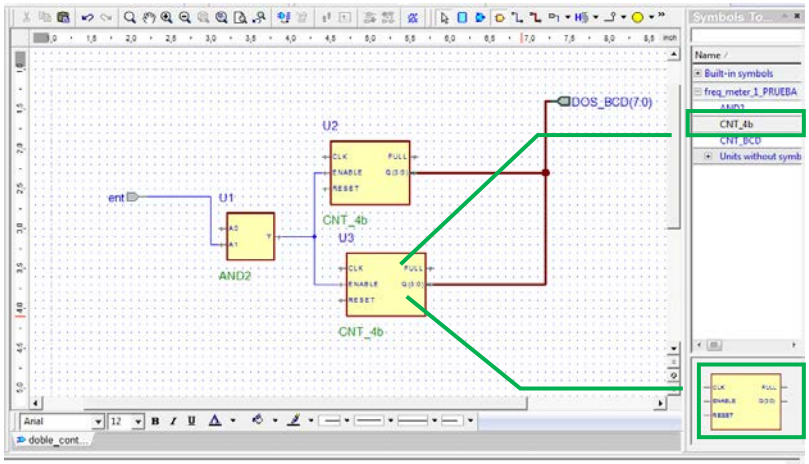
Se diseña en vhdl la puerta and y un contador bcd.



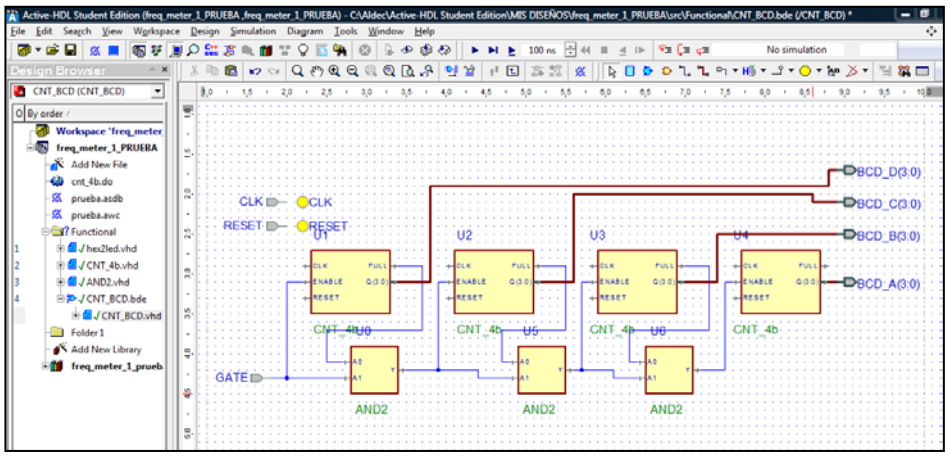
Se crea un nuevo disgrama de bloques. Mediante un asistente se pone nombre al diseño y se generan las entradas y salidas al diseño, al igual que la entidad en vhdl.

Al nuevo diseño se le añaden nuevos bloques a partir de entidades realizadas en vhdl.

Las conexiones se realizan de forma sencilla, como en cualquier captura de esquemas y con ayuda del ratón.

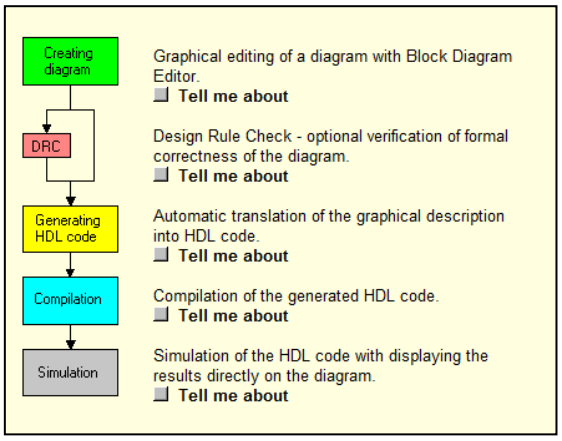


Al final se obtiene el disgrama de bloques deseado en un fichero .bde.



ANEXO A EDICCIÓN POR BLOQUES.

Diagrama de flujo para el diseño.



Ventana del editor.