

Taller introductorio VHDL

IIC2343

Información de contacto:

Sección 1:

- Felipe Valenzuela frvalenzuela@uc.cl (coordinador de proyecto)
- Matías López milopez8@uc.cl

Sección 2:

- Cristóbal Herreros ceherreros@uc.cl
- Raúl Del Río Jara rjdelrio@uc.cl

Formalidades

° Todas las entregas serán grupales (3 personas), al menos **una persona debe tener Vivado instalado y corriendo sin problemas.**

~~° No se permiten grupos intersección.~~

° En caso de grupo intersección, deberán ser responsable de los horarios de su organización para tomar horarios de apoyo

° Serán 5 entregas, donde se podrá eliminar una de las entregas grupales, **a excepción de la última. La primera entrega no se puede eliminar.**

Nota_proyecto: $0.15 * E1 + 0.2 * E2 + 0.2 * E3 + 0.2 * E4 + 0.25 * E5 - 0.2 \min(E2, E3, E4)$

CONTESTAR ENCUESTA

° Falta alrededor de un **19%** de personas que responda la encuesta inicio semestre

° El no responder puede ser perjudicial

<https://forms.gle/6bv6DGU6MQcLQhKr6>

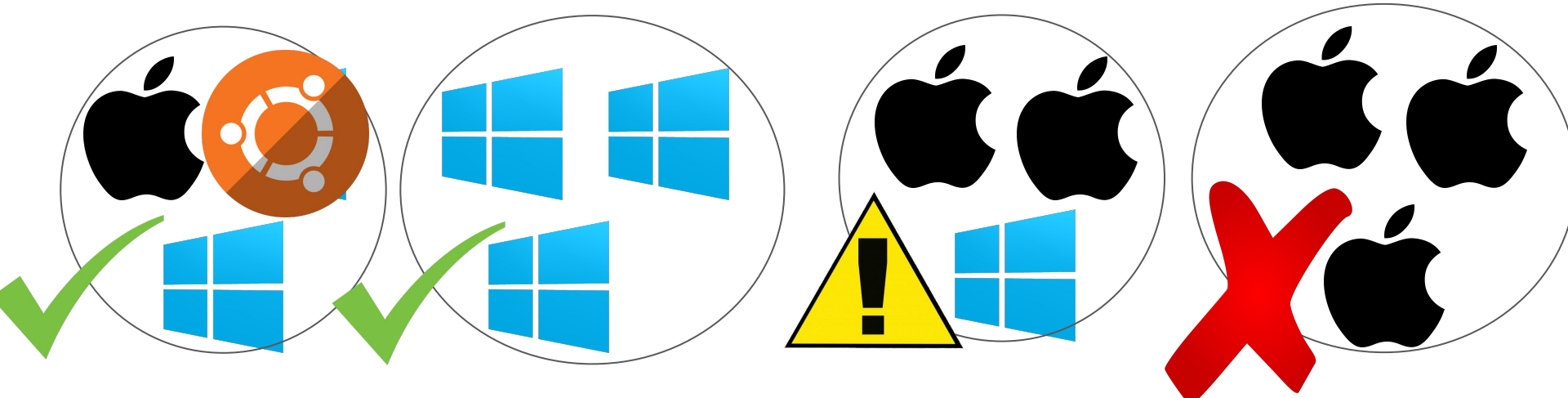


Contenidos

- ° Requisitos de formación de grupos
- ° Mostrar la placa del curso.
 - Componentes de una placa
 - Su conexión con el software Vivado
- ° Vivado y VHDL
- ° Trabajo en equipo remoto
- ° **Simular** funcionamiento de la placa
- ° Atender dudas iniciales

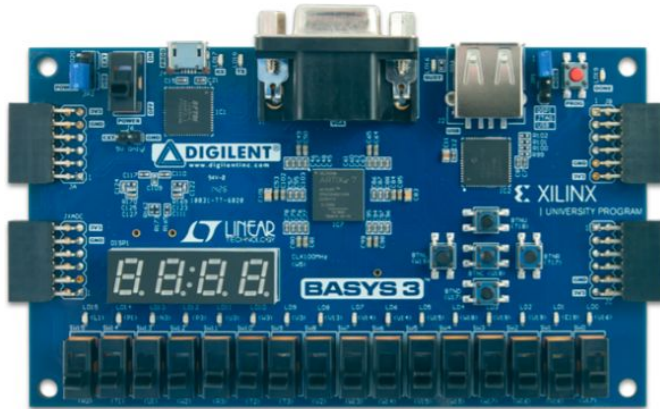
Requisitos de formación de grupos

- ° Se permite elegir los grupos, siempre y cuando al menos un integrante tenga Vivado en su computador funcionando correctamente.
- ° Se recomienda que los grupos existan dos personas con Windows o Ubuntu
- ° No se permite grupos conformados exclusivamente de computadores Apple
- ° En caso de no tener grupo el equipo de ayudante formará grupo según lo rellenado en el formulario inicial.

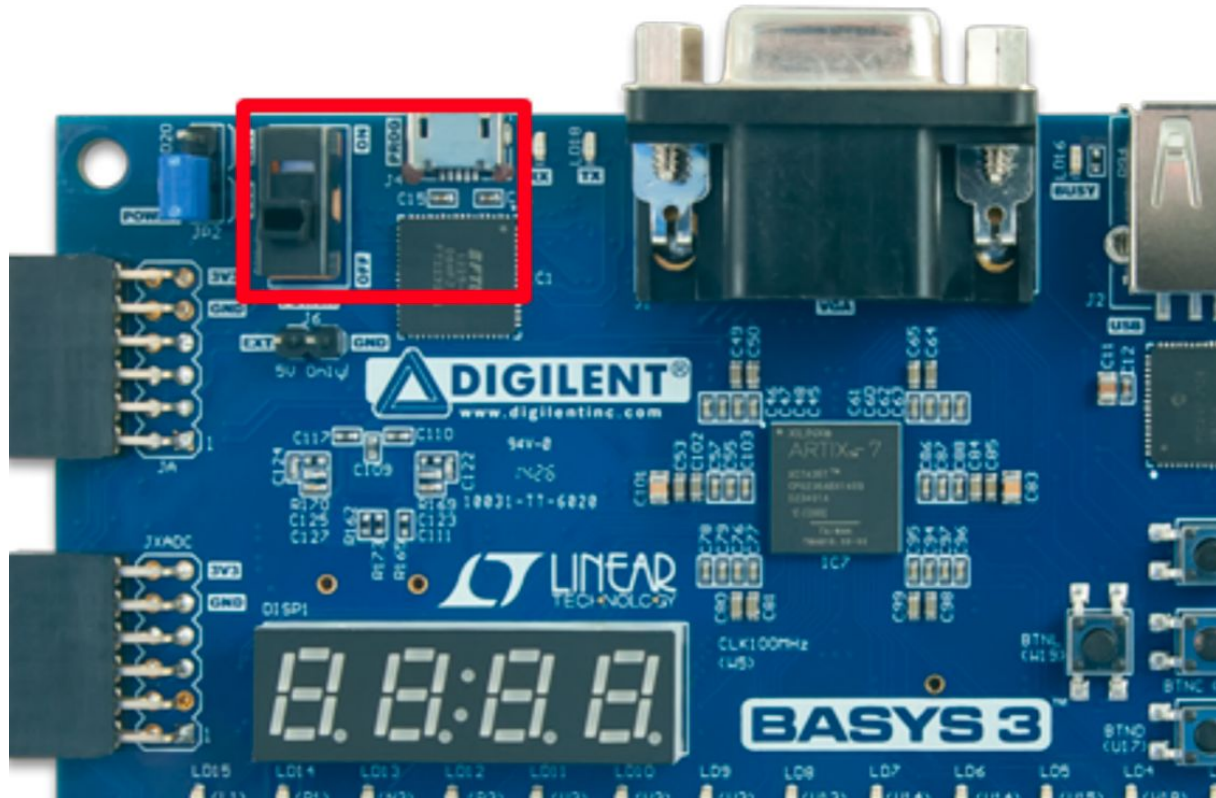


La placa: Basys3

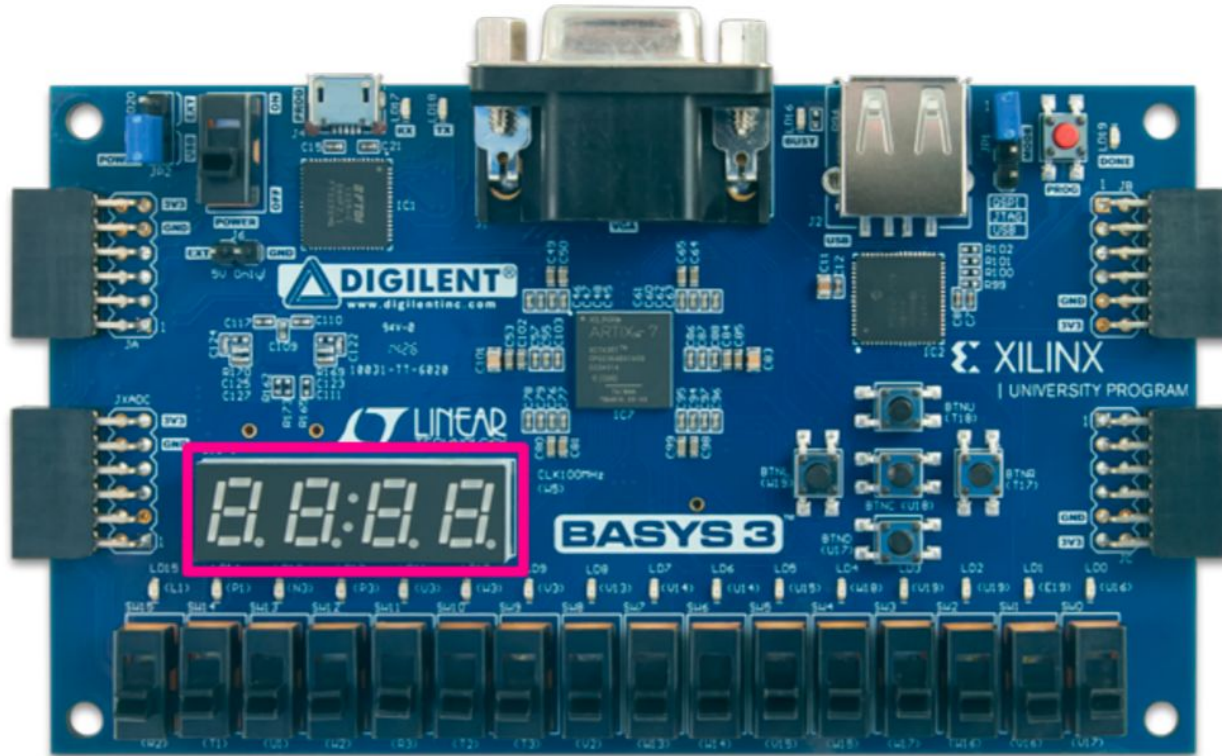
- ° Tratar con cuidado (no es un juguete)
- ° El daño o extravío del material será responsabilidad del estudiante



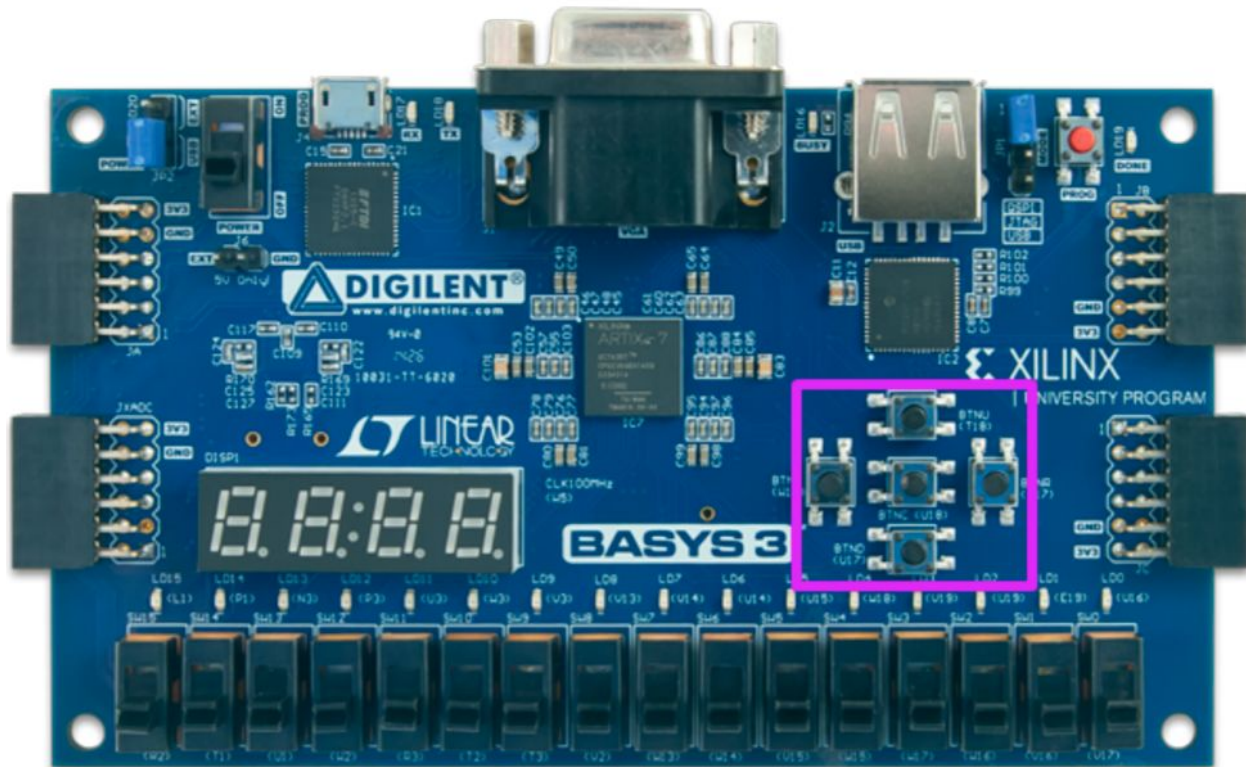
On-Off y fuente de energía



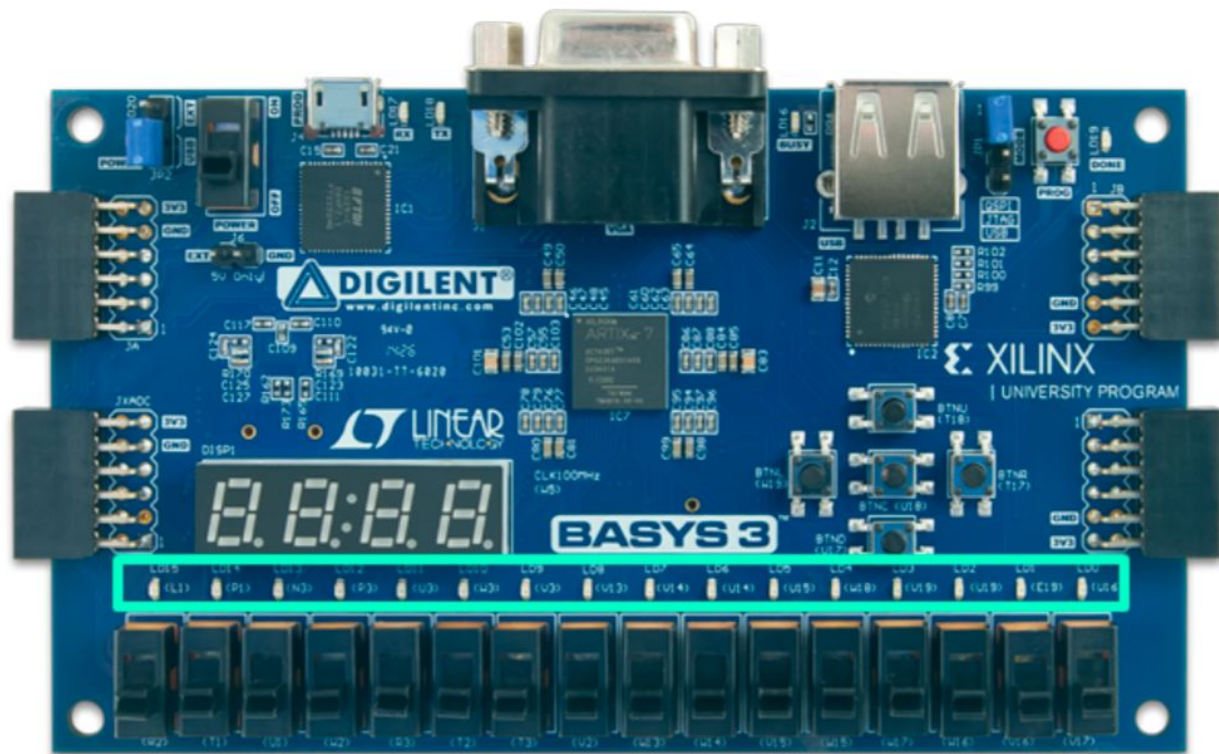
Display de 7 Segmentos



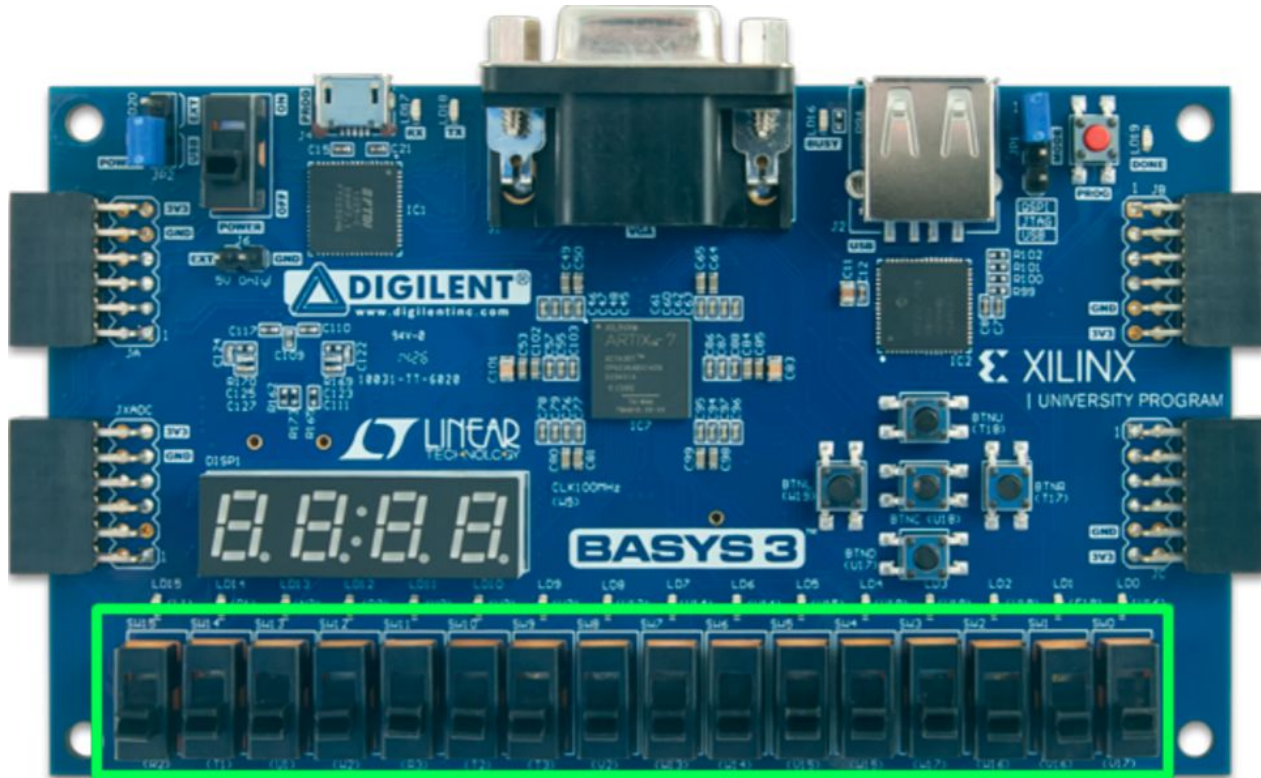
Botones



LEDs



Switches



¿Dudas?

¿Y cómo interactuamos con la placa?

- ° La respuesta es con un software llamado **Vivado**.
- ° El como instalarlo en sus computadores estará no será visto en este taller, pero habrá material adicional en la plataforma del curso para este propósito.
- ° Ahora procederemos a aprender cómo interactuar con este software y que lenguaje ocuparemos dentro del mismo.

Vivado

° Vivado es un programa de síntesis y análisis de lenguaje de **descripción** de hardware (como lo es VHDL que veremos más adelante).

° Será la herramienta que ocuparemos siempre para poder trabajar con la Basys3.

Iniciando el entorno de trabajo:

- ° Una vez iniciado el Vivado seleccionaremos la opción de Create Project



Quick Start

Create Project >

Open Project >

Open Example Project >

Iniciando el entorno de trabajo:

- ° Luego aparecerá una ventana como la siguiente, en ella daremos a Next:
- ° Después damos nombre a nuestro proyecto y asignamos una ruta

Project Name

Enter a name for your project and specify a directory where the project data files will be stored.



Project name: Tutorial

Project location: /home/felipe

☒ Create project subdirectory

Project will be created at: /home/felipe/Tutorial

Iniciando el entorno de trabajo:

° Ahora seleccionaremos el tipo de proyecto. Para las experiencias nosotros ocuparemos el tipo de proyecto llamado RTL Project

° Luego daremos a siguiente.

Project Type

Specify the type of project to create.



- ☒ **RTL Project**
You will be able to add sources, create block designs in IP Integrator, generate IP, run RTL analysis, synthesis, implementation, design planning and analysis.
 - ☒ Do not specify sources at this time
- ☐ **Post-synthesis Project:** You will be able to add sources, view device resources, run design analysis, planning and implementation.
 - ☐ Do not specify sources at this time
- ☐ **I/O Planning Project**
Do not specify design sources. You will be able to view part/package resources.
- ☐ **Imported Project**
Create a Vivado project from a Synplify, XST or ISE Project File.
- ☐ **Example Project**
Create a new Vivado project from a predefined template.

Seleccionar el modelo de placa

Categoría: General Purpose

Familia: Artix-7

Package: cpg236

Velocidad: - 1

Opción: **xc7a35tcpg236-1**

Parts | Boards

Reset All Filters

Category: General Purpose

Family: Artix-7

Package: cpg236

Speed: -1

Temperature: All Remaining

Search: Q-

Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transceivers	GTPE2 Transceivers
xc7a15tcpg236-1	236	106	10400	20800	25	0	45	2	2
xc7a35tcpg236-1	236	106	20800	41600	50	0	90	2	2
xc7a50tcpg236-1	236	106	32600	65200	75	0	120	2	2

¿Dudas?

VHDL

- ° Es un **lenguaje de descripción de hardware**. Usado para **describir** circuitos lógicos.
- ° Esto quiere decir que a través de este lenguaje podemos explicar y representar un circuito eléctrico.
- ° **¡ NO SE PROGRAMA !**
- ° Puede encontrarse similitudes a lenguajes de marcados como son HTML o LaTeX.

HTML



¿Qué es un circuito lógico?

- ° Son una serie de compuertas lógicas conectadas entre sí.
- ° Las compuerta lógica corresponden a componentes que implementan una de las condiciones lógica.

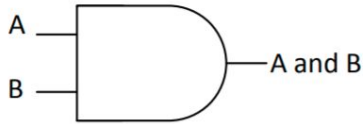


Tabla de valores:

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

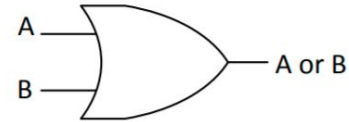


Tabla de valores:

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

¿Dudas?

VHDL

VHDL es un programa de modelación del comportamiento de circuitos eléctricos.
Tiene los siguientes elementos básicos:

1.- Librerías

2.- Entidades

3.- Componentes / Instancias

4.- Señales

5.- Operadores

Librerías

Para trabajar, todos los archivos importan la siguiente librería y usan todos los elementos que provee:

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

Entidad

La descripción de un circuito se conoce como *entity*.

- ° Describen el tipo de puertos que tiene un circuito.
- ° Describe el comportamiento interno.
- ° Tiene **instancias** de otros **componentes** dentro.
- ° **NO ES UNA CLASE.** (estamos describiendo, no programando)

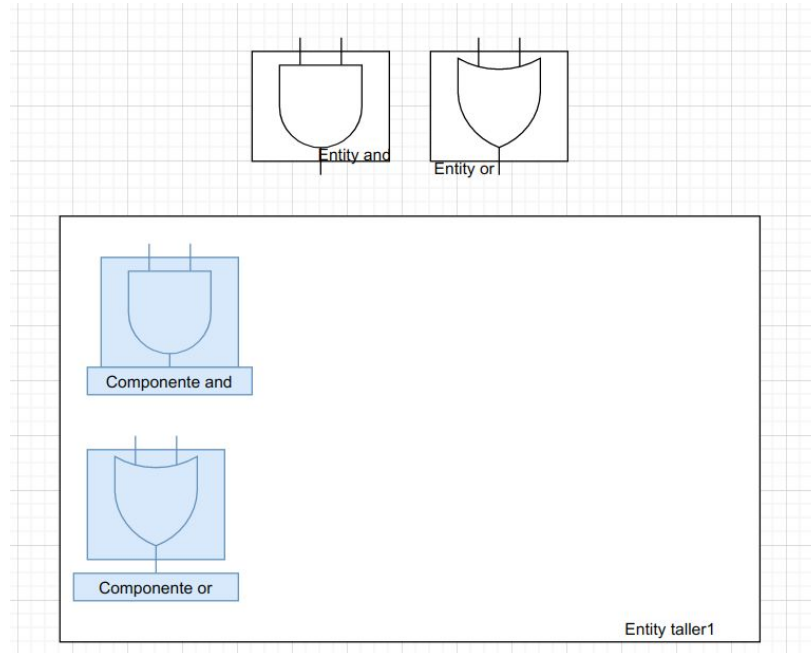
Entidad

Ahora crearemos la siguiente entidad:

```
1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4
5 entity entity_and is
6     Port (
7         entrada1 : in STD_LOGIC;
8         entrada2 : in STD_LOGIC;
9         salida    : out STD_LOGIC
10    );
11 end entity_and;
12
13 architecture Behavioral of entity_and is
14
15     -- Declaración de los componentes
16
17 begin
18
19     -- Declaración de instancias y comportamientos
20
21 salida <= entrada1 and entrada2;
22
23 end Behavioral;
24
25 end Behavioral;
```

Componentes

° Es la forma de describir un entidad dentro de otra.



Componentes

° Es la forma de describir un entidad dentro de otra.

```
entity taller1 is
  Port (
    entrada1 : in STD_LOGIC;
    entrada2 : in STD_LOGIC;
    entrada3 : in STD_LOGIC;
    entrada4 : in STD_LOGIC;
    salida   : out STD_LOGIC
  );
end taller1;

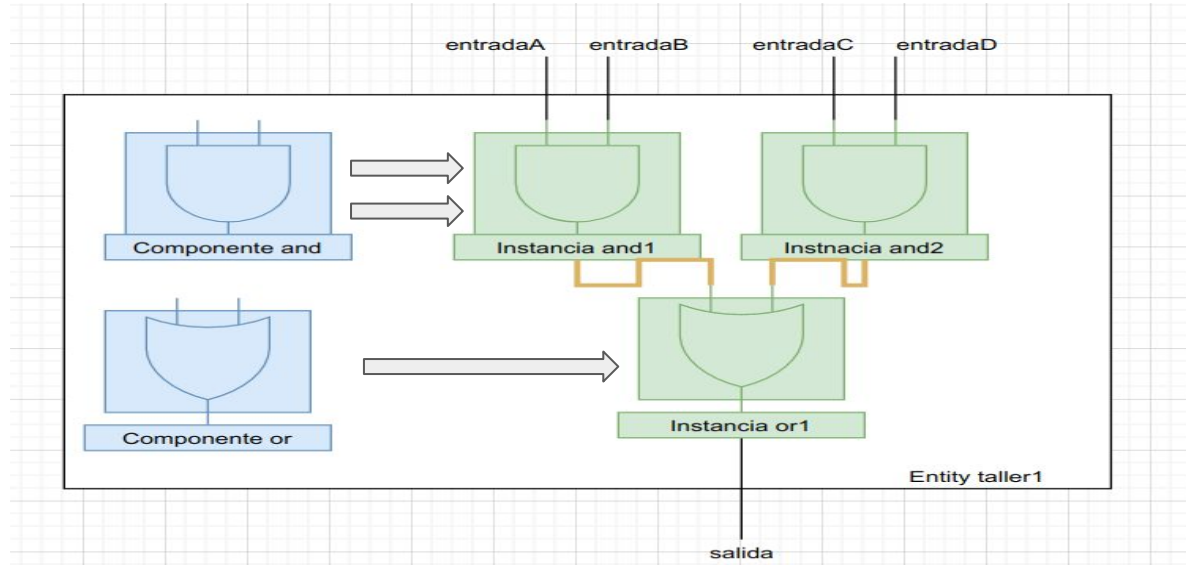
architecture Behavioral of taller1 is

  -- Declaración de los componentes

  component entity_and is
    Port (
      entrada1 : in STD_LOGIC;
      entrada2 : in STD_LOGIC;
      salida   : out STD_LOGIC
    );
  end component;
```

Instancias

- ° Es la forma de describir un el **uso y comportamiento** de un componente
- ° **NO SON CLASES**



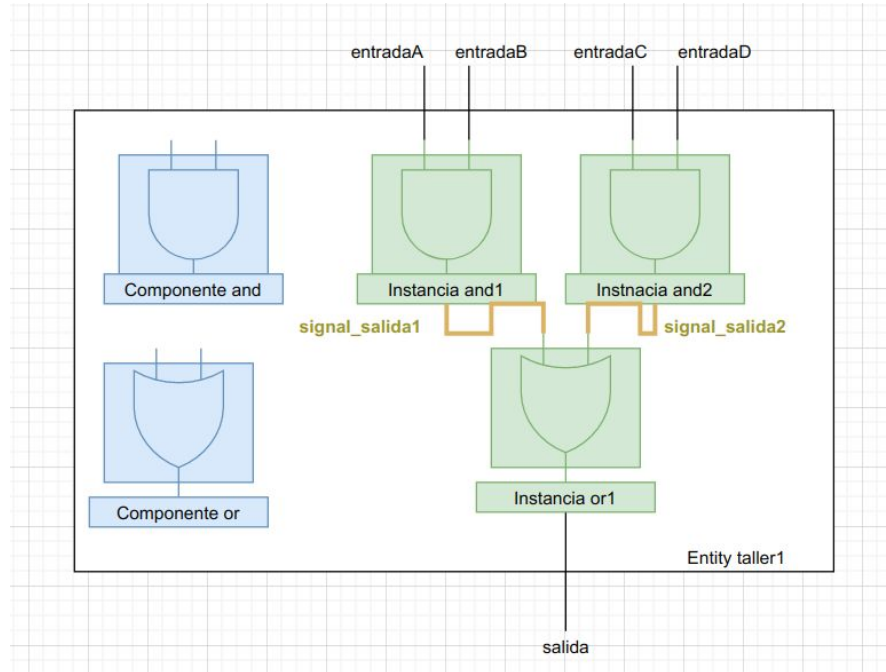
Instancias

° Es la forma de describir un el **uso y comportamiento** de un componente.

```
31 begin
32
33 salida <= signal_salida1 or signal_salida2;
34
35 -- Declaración de instancias y comportamientos
36
37 inst_and1: entity_and port map(
38     entrada1 => entradaA,
39     entrada2 => entradaB,
40     salida => signal_salida1
41
42 );
43
44 inst_and2: entity_and port map(
45     entrada1 => entradaC,
46     entrada2 => entradaD,
47     salida => signal_salida2
48
49 );
50
51
52 end Behavioral;
```

Señales

° Las señales o *signals* se usan para representar cables:



Señales

° Las señales o *signals* se usan para representar cables:

```
component entity_and is
  Port (
    entrada1  : in  STD_LOGIC;
    entrada2  : in  STD_LOGIC;
    salida    : out STD_LOGIC
  );
end component;

signal signal_salida1 : STD_LOGIC;
signal signal_salida2 : STD_LOGIC;

begin

salida <= signal_salida1 or signal_salida2;

-- Declaración de instancias y comportamientos
```

Operadores

VHDL cuenta con los operadores lógicos **and**, **or**, **xor**, **not** y de concatenación **&**

```
signal s1 : std_logic; -- Declaración de senales
signal c1 : std_logic;
signal c2 : std_logic;
signal s2 : std_logic_vector (3 downto 0);
signal s3 : std_logic_vector (3 downto 0);
signal s4 : std_logic_vector (3 downto 0);
signal c3 : std_logic_vector (3 downto 0);

begin

c <= c1 and c2;
s <= s1 or not s2(0);
s4 <= s2 xor s3;
c3 <= '1' & c2 & "01";
```

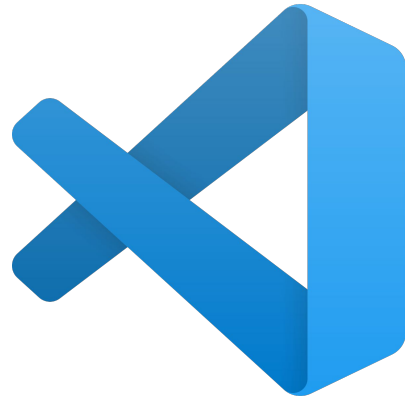
En Resumen:

- **Entidad:** La descripción de un circuito describen el tipo de puertos que tiene un circuito y su comportamiento interno
- **Componente:** Es la forma de describir un entidad dentro de otra.
- **Instancia:** Es la forma de describir un el **uso y comportamiento** de un componente.
- **Señales:** Representan los cables del circuito

¿Dudas?

Trabajo en equipo de forma remota

- ° Para trabajar en equipo de forma remota se contempla que no todos tienen acceso a Vivado, pero al menos a un integrante del grupo si tiene acceso.
- ° Para esto cada integrante deberá trabajar desarrollando entidades, que serán usadas como componentes.
- ° Para escribir código VHDL pueden ocupar editores de texto como Sublime o Visual Studio Code



Break 10 min

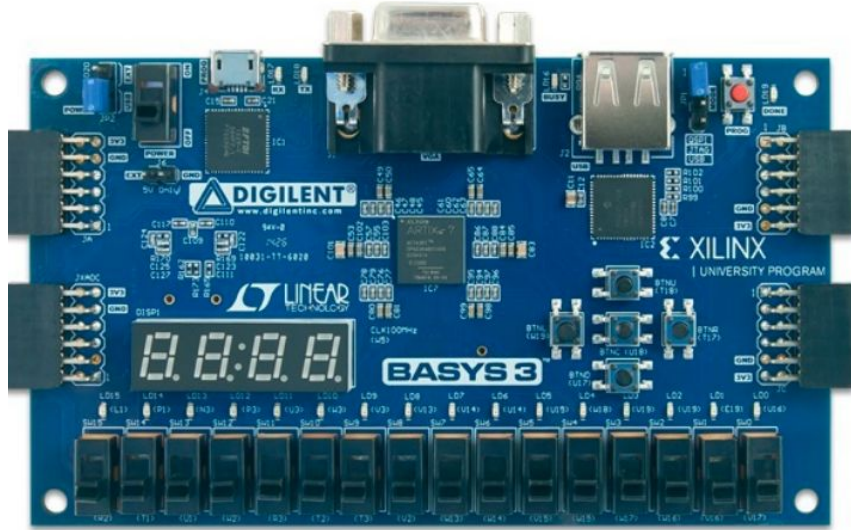
Se acabó VHDL

- ° Ahora estamos casi listos para interactuar con la placa.
- ° Solo falta describir los componentes del hardware, y para eso necesitamos comandos especiales que maneja Vivado.
- ° ¿Cómo accedemos a esos comandos?

Archivo XDC

- ° El archivo XDC por sus siglas en inglés *Xilinx Desing Constrains* es un archivo que contiene un conjunto de comandos que Vivado ocupa para definir las restricciones e interacciones con la placa.
- ° Este archivo será entregado por nosotros y siempre será el mismo, lo importante es que es un archivo con líneas comentadas, es decir, que por defecto no hará ninguna acción.
- ° Es su **deber** descomentar las líneas **específicas y necesarias** para el correcto funcionamiento de su circuito en la placa

Archivo XDC

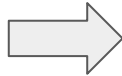
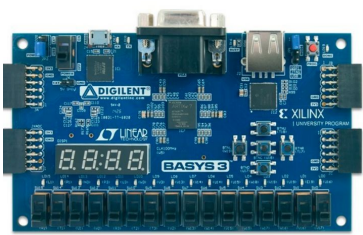


sw[0] V17
sw[1] V16
sw[2] W16
sw[3] W17

XDR

VHDL

Archivo XDC



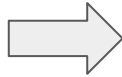
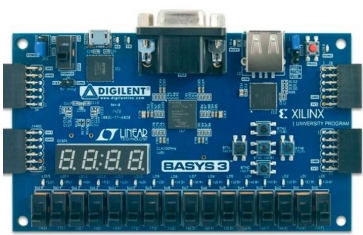
/home/felipe/Basys3.xdc

```
1  ## This file is a general .xdc for the Basys3 rev B board
2  ## To use it in a project:
3  ## - uncomment the lines corresponding to used pins
4  ## - rename the used ports (in each line, after get_ports) as
5
6  ## Clock signal
7  #set_property PACKAGE_PIN W5 [get_ports clk]
8  # set_property IOSTANDARD LVCMOS33 [get_ports clk]
9  # create_clock -name sys_clk_pin -period 10.00 -waveform {0 1}
10
11 ## Switches
12 set_property PACKAGE_PIN V17 [get_ports {sw[0]}]
13   set_property IOSTANDARD LVCMOS33 [get_ports {sw[0]}]
14 set_property PACKAGE_PIN V16 [get_ports {sw[1]}]
15   set_property IOSTANDARD LVCMOS33 [get_ports {sw[1]}]
16 set_property PACKAGE_PIN W16 [get_ports {sw[2]}]
17   set_property IOSTANDARD LVCMOS33 [get_ports {sw[2]}]
18 set_property PACKAGE_PIN W17 [get_ports {sw[3]}]
19   set_property IOSTANDARD LVCMOS33 [get_ports {sw[3]}]
20 #set_property PACKAGE_PIN W15 [get_ports {sw[4]}]
21 # set_property IOSTANDARD LVCMOS33 [get_ports {sw[4]}]
22 #set_property PACKAGE_PIN V15 [get_ports {sw[5]}]
23 # set_property IOSTANDARD LVCMOS33 [get_ports {sw[5]}]
24 #set_property PACKAGE_PIN W14 [get_ports {sw[6]}]
25 # set_property IOSTANDARD LVCMOS33 [get_ports {sw[6]}]
26 #set_property PACKAGE_PIN W13 [get_ports {sw[7]}]
27 # set_property IOSTANDARD LVCMOS33 [get_ports {sw[7]}]
28 ##set_property PACKAGE_PIN V2 [get_ports {sw[8]}]
29 # set_property IOSTANDARD LVCMOS33 [get_ports {sw[8]}]
30 #set_property PACKAGE_PIN T3 [get_ports {sw[9]}]
```

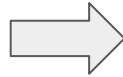
sw (3 downto 0)

VHDL

Archivo XDC



XDC



```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity Basys3 is
5      Port ( sw : in STD_LOGIC_VECTOR (4 downto 0);
6            led : out STD_LOGIC_VECTOR (7 downto 0)
7            );
8  end Basys3;
9
10 architecture Behavioral of Basys3 is
11
```

¿Dudas?

Generar el Bitstream

La placa funciona usando un archivo .bit (Bitstream), para generarlo hay tres etapas:

1. Run Synthesis
2. Run Implementation
3. Generate Bitstream

En Hardware Manager seleccionar la opción Program Device para programar la placa.

✓ SYNTHESIS

- ▶ Run Synthesis
- > Open Synthesized Design

✓ IMPLEMENTATION

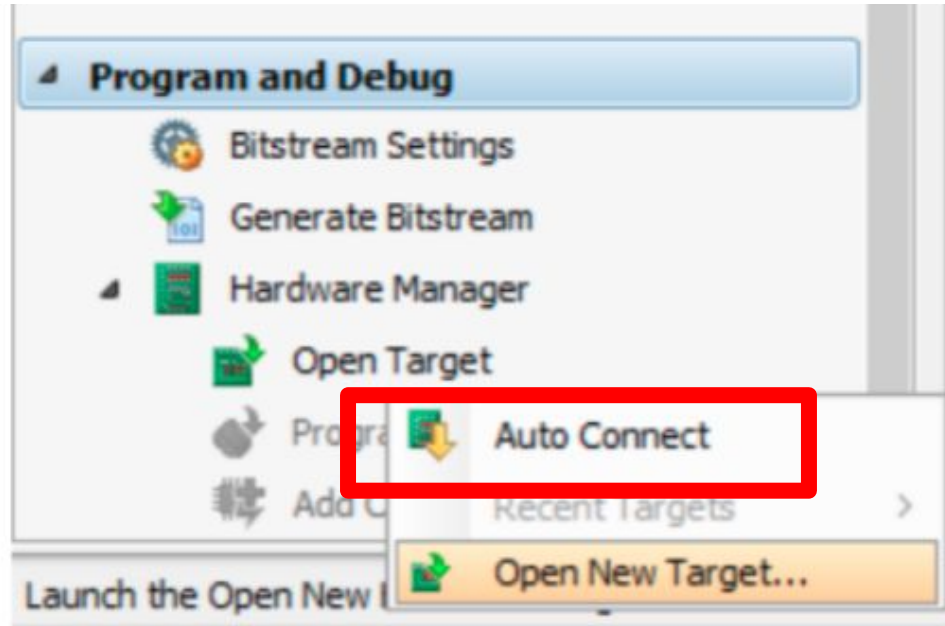
- ▶ Run Implementation
- > Open Implemented Design

✓ PROGRAM AND DEBUG

- ⬇ Generate Bitstream
- ✓ **Open Hardware Manager**
 - Open Target
 - Program Device

Conectar la placa

° Para poder conectar la placa vamos al costado inferior izquierdo, y apretamos en auto conect:



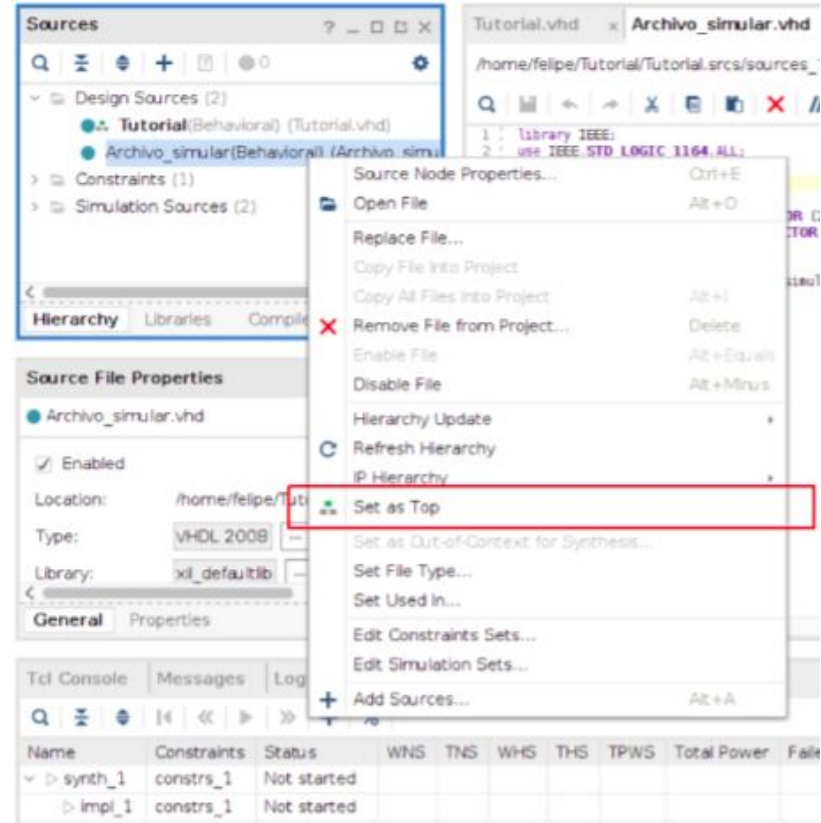
¿Dudas?

Simulación

◦ Vivado permite simular el comportamiento de un circuito, lo que es una herramienta bastante útil cuando se trata de la construcción de hardware.

◦ Para simular solo es necesario correr la síntesis.

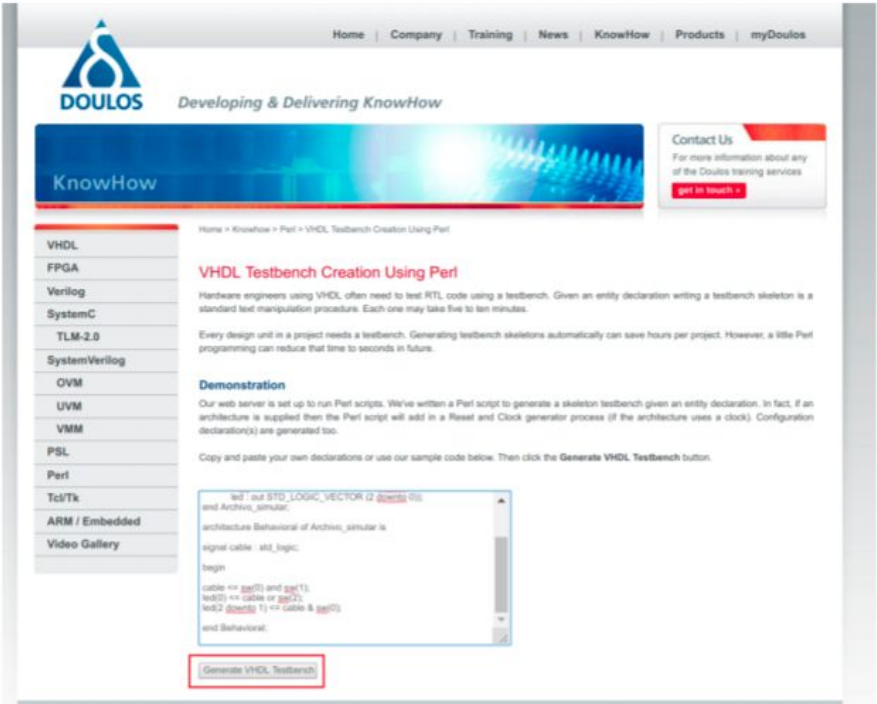
◦ Es necesario que el archivo a simular sea puesto como Set as a Top



Simulación

° Crear un archivo de simulación es similar a lo que vimos anteriormente solo que seleccionamos la opción de *simulation sources*

° Ocuparemos un sitio que nos creará el archivo de simulación:
https://www.doulos.com/knowhow/perl/testbench_creation/



The screenshot shows the Doulos KnowHow website. The header includes the Doulos logo and navigation links: Home, Company, Training, News, KnowHow, Products, and myDoulos. The main banner reads 'KnowHow' and 'Developing & Delivering KnowHow'. A sidebar on the left lists various topics: VHDL, FPGA, Verilog, SystemC, TLM-2.0, SystemVerilog, OVM, UVM, VMM, PSL, Perl, Tcl/Tk, ARM / Embedded, and Video Gallery. The main content area is titled 'VHDL Testbench Creation Using Perl' and contains text explaining the purpose of the tool, a 'Demonstration' section with a code editor showing VHDL testbench code, and a 'Generate VHDL Testbench' button at the bottom.

Home | Company | Training | News | KnowHow | Products | myDoulos

DOULOS Developing & Delivering KnowHow

KnowHow

Home > KnowHow > Perl > VHDL Testbench Creation Using Perl

VHDL Testbench Creation Using Perl

Hardware engineers using VHDL often need to test RTL code using a testbench. Given an entity declaration writing a testbench skeleton is a standard test manipulation procedure. Each one may take five to ten minutes.

Every design unit in a project needs a testbench. Generating testbench skeletons automatically can save hours per project. However, a little Perl programming can reduce that time to seconds in future.

Demonstration

Our web server is set up to run Perl scripts. We've written a Perl script to generate a skeleton testbench given an entity declaration. In fact, if an architecture is supplied then the Perl script will add in a Reset and Clock generator process (if the architecture uses a clock). Configuration declaration(s) are generated too.

Copy and paste your own declarations or use our sample code below. Then click the **Generate VHDL Testbench** button.

```
test : out STD_LOGIC_VECTOR (2 downto 0);
and Archivo_simular;
architecture Behavioral of Archivo_simular is
    signal cable : std_logic;
begin
    cable <= pp(0) and pp(1);
    test0 <= cable or pp(2);
    test2 (dout0 1) <= cable & pp(0);
end Behavioral;
```

Generate VHDL Testbench

Simulación

° Ahora escribimos los valores que tomarán los distintos inputs. Por ejemplo los switches.

```
begin
    -- Put initialisation code here
    SW <= "000";
    wait for 100 ns;
    SW <= "001";
    wait for 100 ns;
    SW <= "010";
    wait for 100 ns;
    SW <= "011";
    wait for 100 ns;
    -- Put test bench stimulus code here

    wait;
end process;
```

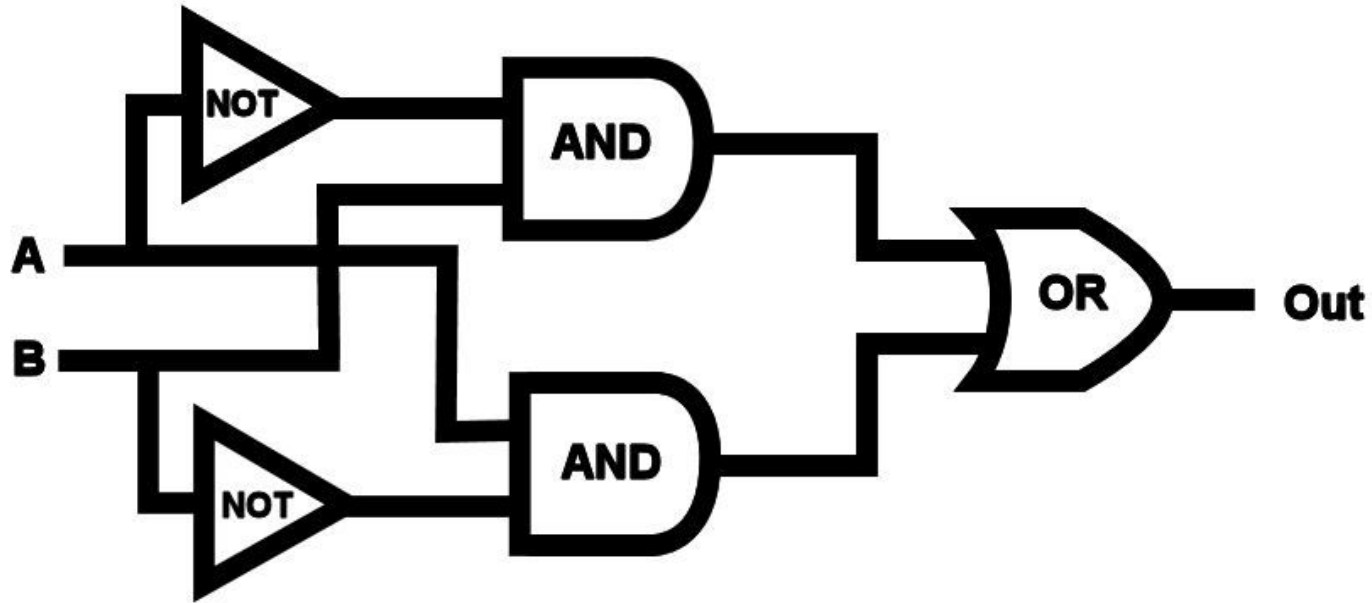
¿Dudas?

Ejercicio 1 del taller

- ° Con todo lo aprendido en este taller conectaremos a la entidad de ejemplo en la que estuvimos trabajando cuando explicamos los conceptos de componente instancia a la placa.
- ° Crea una nueva entidad llamada Basys3 que tenga como input los 4 primeros switches (los de más a la derecha), los cuales serán conectada a una sola instancia del componente taller1, el output tiene que ser el primer led (de derecha a izquierda).
- ° No olvides descomentar las líneas adecuadas del XDC.
- ° Sintetiza y pruébalo en la simulación!

Ejercicio 2 del taller

° Crear una entidad que haga lo siguiente y verifique su resultado en pizarra:



Ejercicio 3 del taller (Propuesto)

° En clases vieron un full adder, construya un full adder de 4 bits y simulelo

