Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores

Tutorial Simulación Vivado

Objetivos

- Lograr simular un circuito desde la aplicación Vivado.
- Visualizar las señales generadas en cada instancia.

Antes de empezar

En este documento se detallará cómo podrán comenzar a probar su código de VHDL sin necesidad de la placa. Tengan en consideración que se da por hecho de que ustedes ya entienden cómo funciona el lenguaje de VHDL a la hora de leer lo que sigue a continuación, si aun no lo han hecho por favor lean los pdfs introductorios publicados en el github del curso.

Introducción

Durante el semestre, los alumnos del curso deben simular en una placa `Basys3` un computador básico utilizando un lenguaje de simulación de hardware, es decir, simular los componentes y conexiones entre ellos, junto con manejar *input* y *output* de forma binaria para su procesamiento. El siguiente tutorial tiene como objetivo enseñar una alternativa de testing del computador básico que no requiere del proceso de simular el proyecto en en la placa entregada para probar el funcionamiento del mismo. Con esta herramienta, los alumnos podrán visualizar en su computador como van variando las señales generadas durante el tiempo con el fin de *debuggear* con mayor rapidez su computador.

Pasos Previos

Antes de crear un archivo de simulación, es necesario verificar:

1. Que todos los bloques de control de flujo, como **with/select** o **if/else** tengan la opción por default.
   - ***with/select***: si no cubren todas las opciones de la señal de control, deben incluir la opción *others*

     ```
     with a select b <=
     "1000" when "00",
     "0100" when "01",
     "0010" when "10",
     "0001" when "11";
     "0000" when others;
     ```

   - ***if/else*:** deben incluir el **else** aunque no se utilice.

```
if (X = 5) and (Y = 9) then
  Z <= A;
elsif (X >= 5) then
  Z <= B;
else
  -- no tienen porqué tener alguna instrucción, puede quedar vacío.
end if;
```

2. Deben inicializar todas las señales de sus componentes en 0 de la siguiente manera:

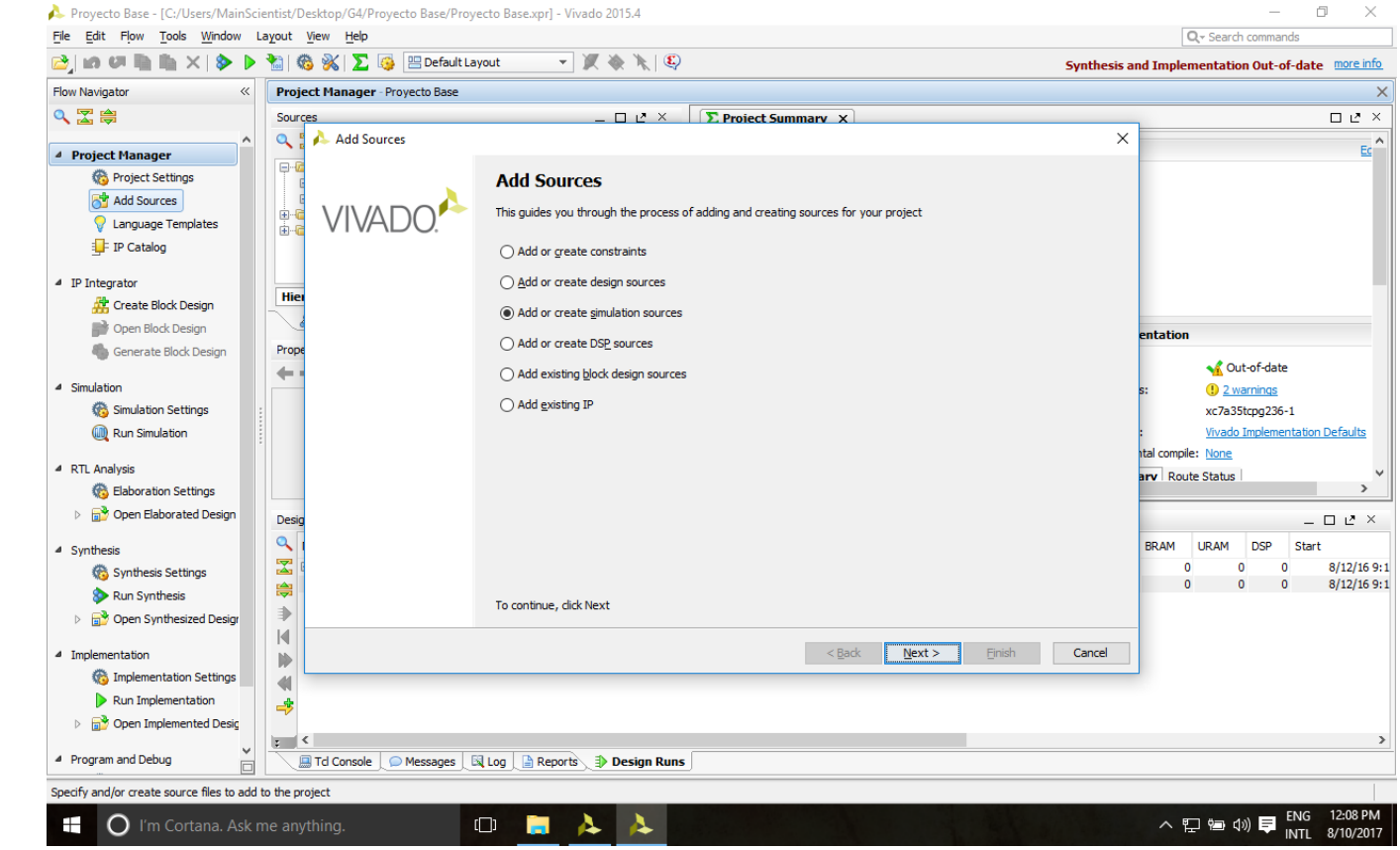
```
signal btn : std_logic_vector (4 downto 0) := (others => '0');
signal c : std_logic  := '0';
```

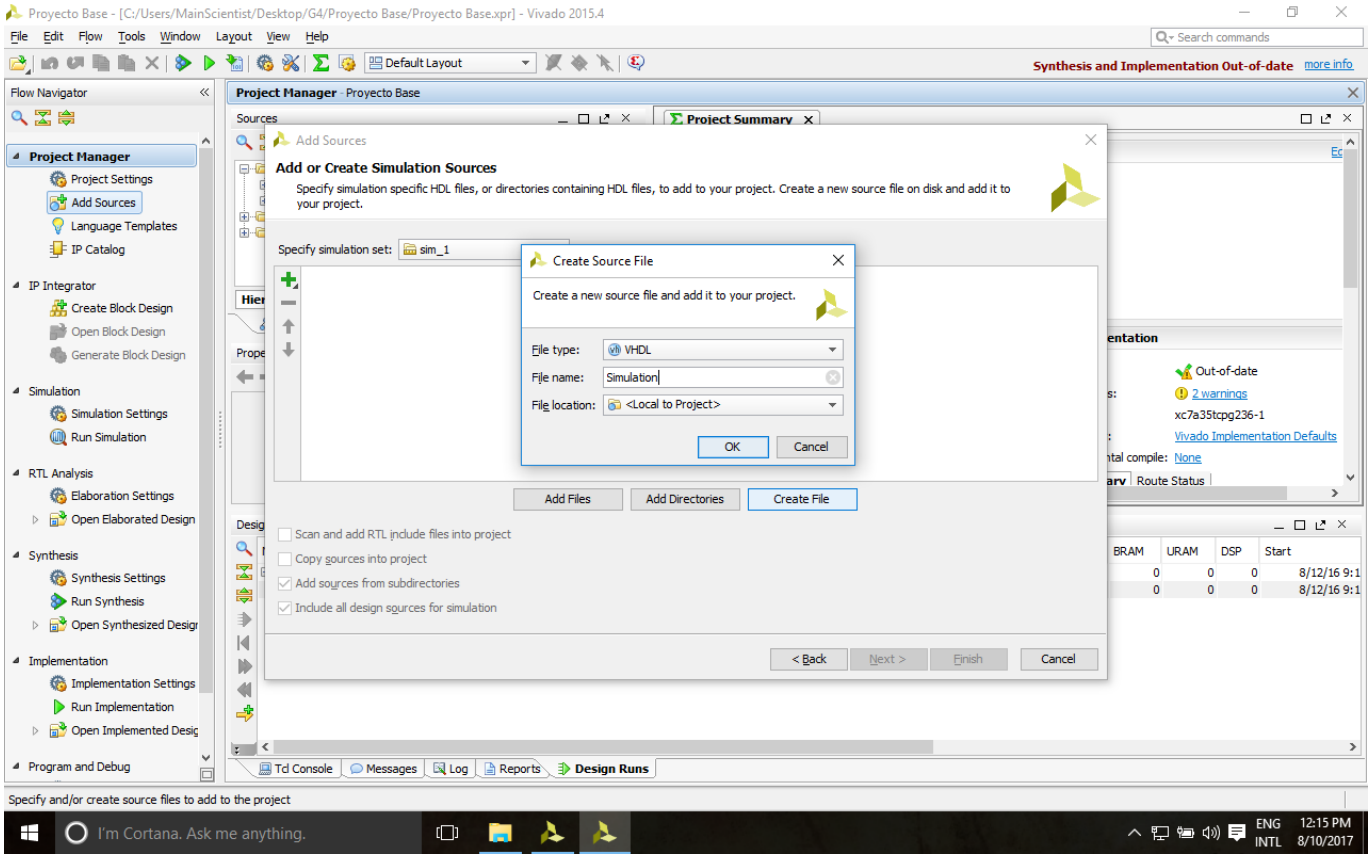
3. Cambiar el clock a full.

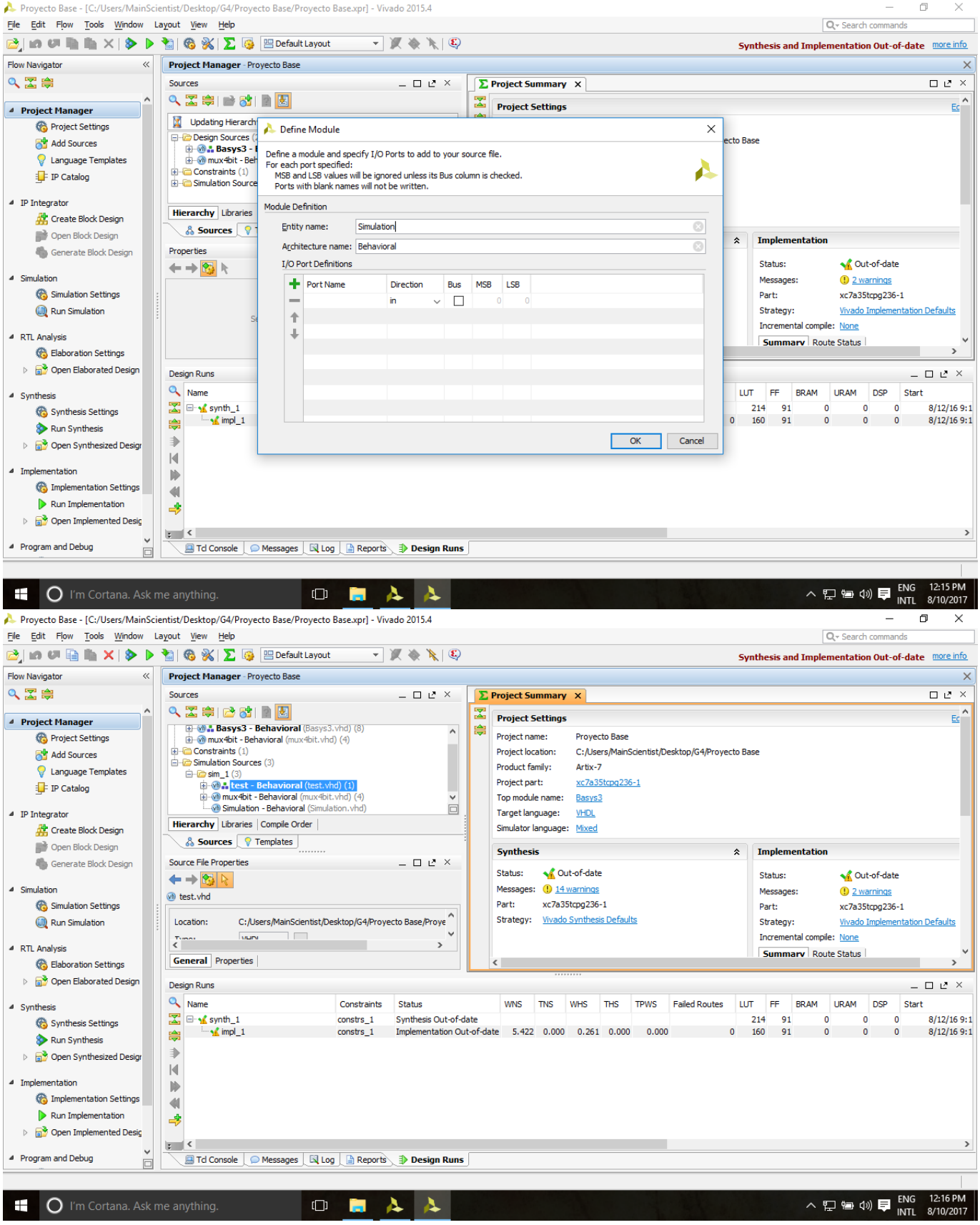
```
inst_Clock_Divider: Clock_Divider port map( -- No Tocar - Intancia de Clock_Divider.
    clk         => clk,  -- No Tocar - Entrada del clock completo (100Mhz).
    speed       => "00", -- Selector de velocidad: "00" full, "01" fast, "10" normal y "11" s
    clock       => clock -- No Tocar - Salida del clock reducido: 25Mhz, 8hz, 2hz y 0.5hz.
    );
```

Crear un archivo simulación

Para crear este archivo deben apretar el botón `add source...` y luego hacer click en `add simulation source`. Ponerle el nombre que estimen conveniente.


Proyecto Base - [C:/Users/MainScientist/Desktop/G4/Proyecto Base/Proyecto Base.xpr] - Vivado 2015.4
File Edit Flow Tools Window Layout View Help
Search commands
Default Layout
Synthesis and Implementation Out-of-date more info
Flow Navigator
Project Manager
Project Settings
Add Sources
Language Templates
IP Catalog
IP Integrator
Create Block Design
Open Block Design
Generate Block Design
Simulation
Simulation Settings
Run Simulation
RTL Analysis
Elaboration Settings
Open Elaborated Design
Synthesis
Synthesis Settings
Run Synthesis
Open Synthesized Design
Implementation
Implementation Settings
Run Implementation
Open Implemented Design
Program and Debug
Project Manager - Proyecto Base
Sources
Project Summary
Add Sources
VIVADO
Add Sources
This guides you through the process of adding and creating sources for your project
Add or create constraints
Add or create design sources
Add or create simulation sources
Add or create DSP sources
Add existing block design sources
Add existing IP
To continue, click Next
< Back
Next >
Finish
Cancel
Out-of-date
2 warnings
xc7a35tcpg236-1
Vivado Implementation Defaults
None
Route Status
BRAM URAM DSP Start
0 0 0 8/12/16 9:1
0 0 0 8/12/16 9:1
Tcl Console Messages Log Reports Design Runs
Specify and/or create source files to add to the project
I'm Cortana. Ask me anything.
ENG INTL 12:08 PM 8/10/2017


Proyecto Base - [C:/Users/MainScientist/Desktop/G4/Proyecto Base/Proyecto Base.xpr] - Vivado 2015.4
File Edit Flow Tools Window Layout View Help
Search commands
Default Layout
Synthesis and Implementation Out-of-date more info
Flow Navigator
Project Manager
Project Settings
Add Sources
Language Templates
IP Catalog
IP Integrator
Create Block Design
Open Block Design
Generate Block Design
Simulation
Simulation Settings
Run Simulation
RTL Analysis
Elaboration Settings
Open Elaborated Design
Synthesis
Synthesis Settings
Run Synthesis
Open Synthesized Design
Implementation
Implementation Settings
Run Implementation
Open Implemented Design
Program and Debug
Project Manager - Proyecto Base
Sources
Project Summary
Add Sources
Add or Create Simulation Sources
Specify simulation specific HDL files, or directories containing HDL files, to add to your project. Create a new source file on disk and add it to your project.
Specify simulation set: sim_1
Create Source File
Create a new source file and add it to your project.
File type: VHDL
File name: Simulation
File location: <Local to Project>
OK
Cancel
Add Files
Add Directories
Create File
Scan and add RTL include files into project
Copy sources into project
Add sources from subdirectories
Include all design sources for simulation
< Back
Next >
Finish
Cancel
Out-of-date
2 warnings
xc7a35tcpg236-1
Vivado Implementation Defaults
None
Route Status
BRAM URAM DSP Start
0 0 0 8/12/16 9:1
0 0 0 8/12/16 9:1
Tcl Console Messages Log Reports Design Runs
Specify and/or create source files to add to the project
I'm Cortana. Ask me anything.
ENG INTL 12:15 PM 8/10/2017



Se crerá una nueva carpeta que contendrá los archivos de simulación, como se aprecia en la cuarta figura.

Ahora, hay que abrir el archivo recién creado, que tendrá el nombre que dieron.

Escribir código en el archivo

1. El archivo recién generado se verá así:

```
entity sim1 is
--  Port ( );
end sim1;

architecture Behavioral of sim1 is

begin


end Behavioral;
```

2. En el archivo recién generado, deben crear una instancia del componente `Basys3`, para esto deben declararla como componente.

```
component Basys3
    Port (
        sw          : in   std_logic_vector (2 downto 0);
        btn         : in   std_logic_vector (4 downto 0);
        led         : out  std_logic_vector (3 downto 0)
        clk         : in   std_logic;
        seg         : out  std_logic_vector (7 downto 0);
        an          : out  std_logic_vector (3 downto 0)
    );
end component;
```

3. Agregar las señales que corresponden a este componente. A continuacoón, se muestran las señales de entrada y salida del componente Basys3 (ojo que estas señales cambiarán a lo largo del semestre):

```
signal sw  : std_logic_vector(2 downto 0)  := (others => '0');
signal btn : std_logic_vector (4 downto 0)  := (others => '0');
signal led : std_logic_vector (3 downto 0) := (others => '0');
signal clk : std_logic := '0';
signal seg : std_logic_vector (7 downto 0)  := (others => '0');
signal an  : std_logic_vector (3 downto 0)  := (others => '0');
```

4. Ahora que recordamos sus seáles, vamos a instanciar Basys3 despues del 'begin'

```
inst_Basys: Basys3 port map(
 sw     => sw,
 btn    => btn,
 led    => led,
 clk    => clk,
 seg    => seg,
 an     => an
);
```

5. Generar proceso donde cambiamos el clock cada 5 nanosegundos, (tambien podemos cambiar algo más como apretar los switch en algún momento o los botones)

```
proc: process
    begin
        btn <= "01010";
        clk <= '0';
        L1: loop
            if clk = '0' then
                clk <= '1';
            else
                clk <= '0';
            end if;
            wait for 5 ns;
        end loop L1;
    wait;
end process proc;
```

Archivo de ejemplo

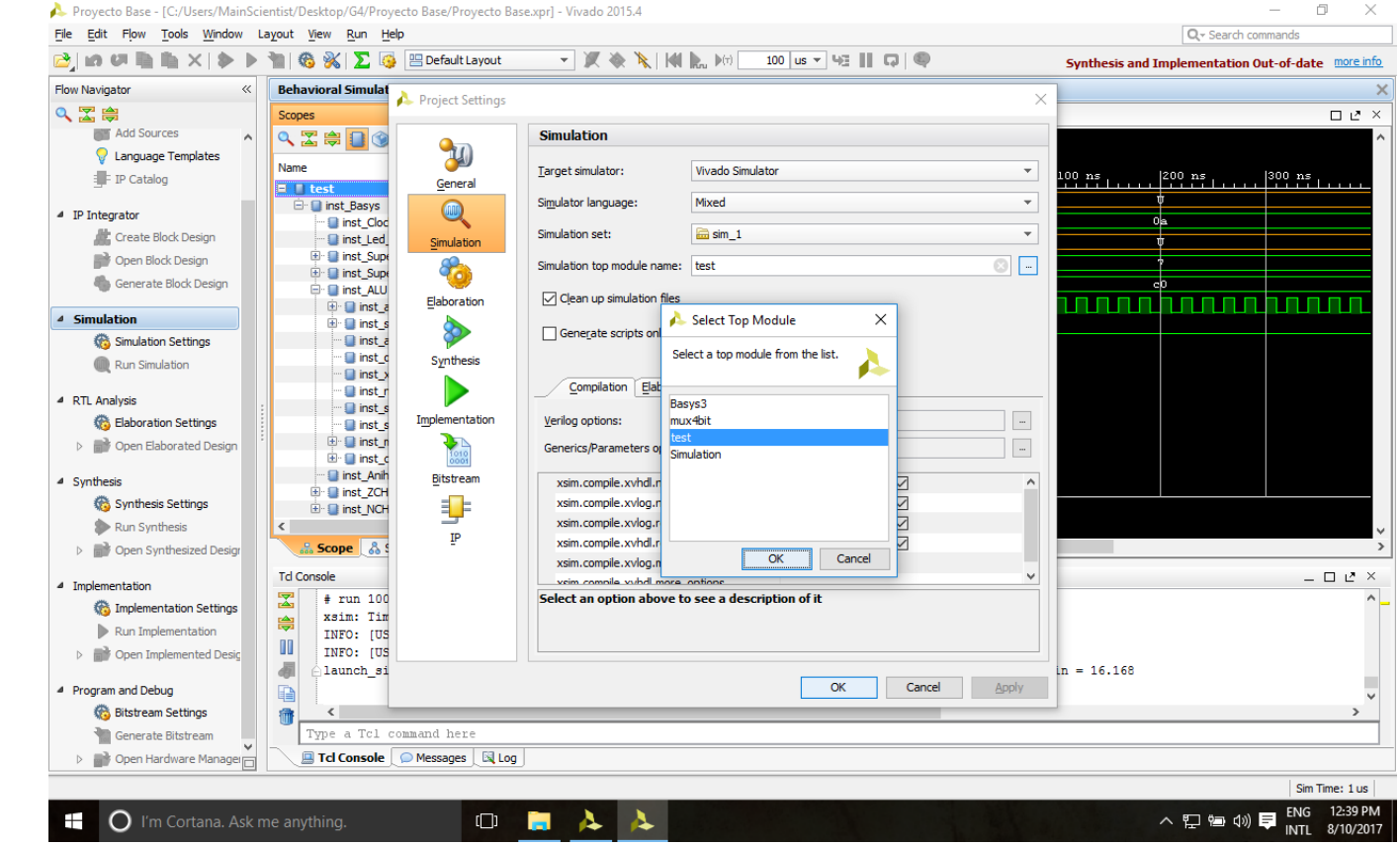
```
entity sim1 is
--  Port ( );
end sim1;

architecture Behavioral of sim1 is

component Basys3
    Port (
        sw          : in   std_logic_vector (3 downto 0);
        btn         : in   std_logic_vector (4 downto 0);
        led         : out  std_logic_vector (3 downto 0);
        clk         : in   std_logic;
        seg         : out  std_logic_vector (7 downto 0);
        an          : out  std_logic_vector (3 downto 0)
    );
end component;

signal sw  : std_logic_vector(3 downto 0)  := (others => '0');
signal btn : std_logic_vector (4 downto 0)  := (others => '0');
signal led : std_logic_vector (3 downto 0) := (others => '0');
signal clk : std_logic := '0';
signal seg : std_logic_vector (7 downto 0)  := (others => '0');
signal an  : std_logic_vector (3 downto 0)  := (others => '0');

BEGIN
 inst_Basys: Basys3 port map (sw, btn, led, clk, seg, an);
 proc: process
 begin
     btn <= "01010";
```

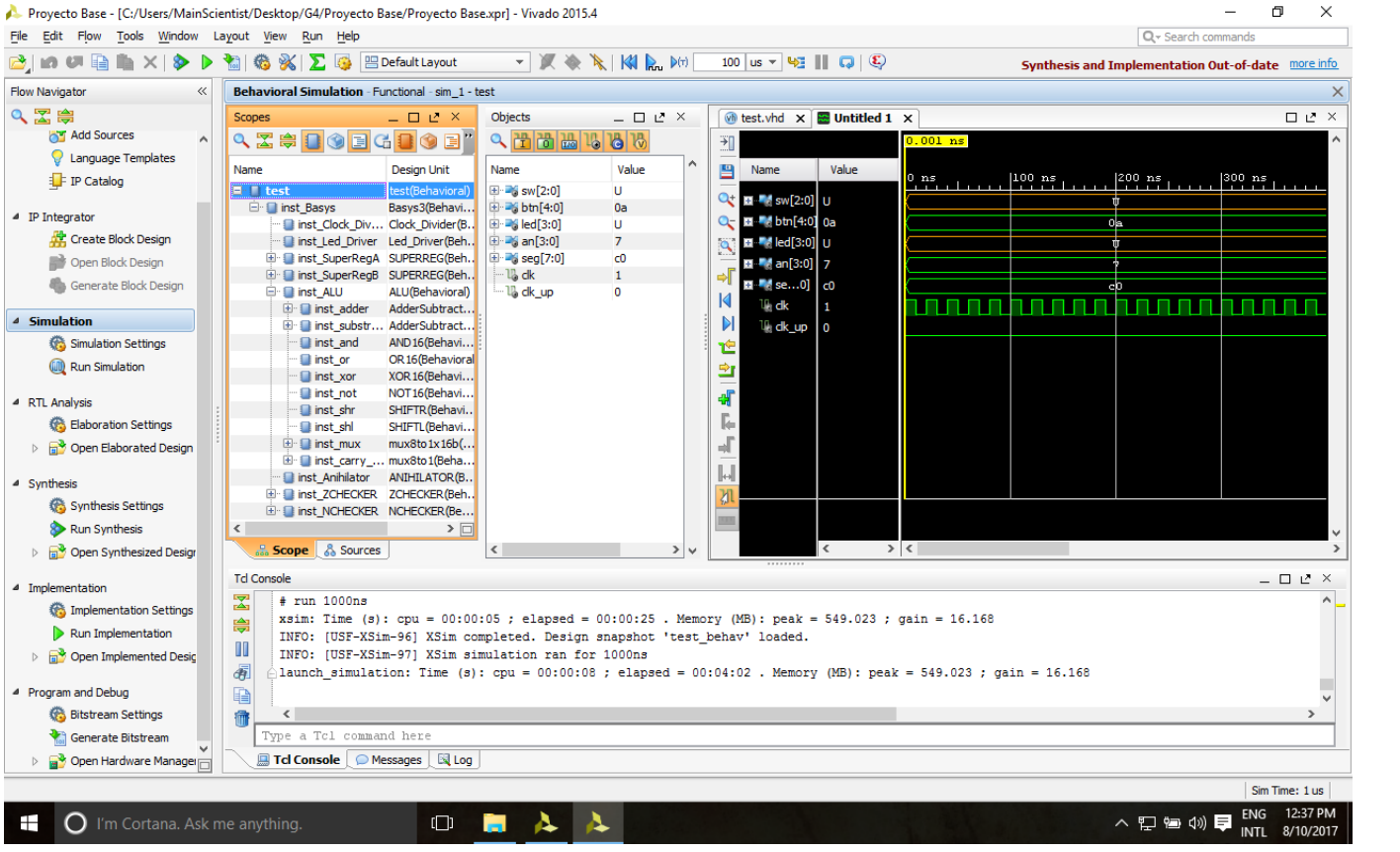
```
        sw <= "1001";
        clk <= '0';
        L1: loop
            if clk = '0' then
                clk <= '1';
            else
                clk <= '0';
            end if;
            wait for 10 ns;
        end loop L1;
    wait;
    end process proc;
END Behavioral;
```

Ejecutar la simulación

Para correr la simulación primero debemos seleccionar el archivo que acabamos de crear como archivo de simulación. Para esto se debe hacer click en **Simulation Settings** en el panel izquierdo de Vivado y luego, en la ventana que se abre, hacer click en los puntos suspensivos y seleccionar la entidad.

Una vez configurado esto se debe hacer click en **Run Simulation**. Esto compilará el código (toma tiempo) y abrirá una ventana con el resultado de la simulación, en esta verán una ventana negra que contiene las señales y sus valores durante el tiempo. Para agregar señales a esta ventana se pueden arrastrar desde los paneles a la izquierda que contienen las entidades (en el panel de más a la izquierda) y las señales respectivas a la entidad seleccionada (en el panel central).


Proyecto Base - [C:/Users/MainScientist/Desktop/G4/Proyecto Base/Proyecto Base.xpr] - Vivado 2015.4
Project Settings
Simulation
Target simulator: Vivado Simulator
Simulator language: Mixed
Simulation set: sim_1
Simulation top module name: test
Select Top Module
Select a top module from the list.
Basys3
mux4bit
test
Simulation
OK
Cancel
Select an option above to see a description of it
Tcl Console


Proyecto Base - [C:/Users/MainScientist/Desktop/G4/Proyecto Base/Proyecto Base.xpr] - Vivado 2015.4
Behavioral Simulation - Functional - sim_1 - test
Scopes
Objects
test.vhd
Untitled 1
Tcl Console
run 1000ns
xsim: Time (s): cpu = 00:00:05 ; elapsed = 00:00:25 . Memory (MB): peak = 549.023 ; gain = 16.168
INFO: [USF-XSim-96] XSim completed. Design snapshot 'test_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 1000ns
launch_simulation: Time (s): cpu = 00:00:08 ; elapsed = 00:04:02 . Memory (MB): peak = 549.023 ; gain = 16.168