

Sistemas Digitales para las Comunicaciones

Ejercicio 11

para la Arty A-35



Profesor: Federico Zacchigna

Alumno: Pablo Daniel Folino

Repositorio de trabajos prácticos: <https://github.com/MSE-SDC/trabajos-practicos-PabloFolino>

2021

Índice

Trabajo práctico ejercicio 11.....	2
Ejecutar el Vivado.....	2
El paso a paso.....	2
Configurar el ILA (Integrated Logic Analyzer).....	11
Configurar el VIO (Virtual Input/Output).....	14
Pruebas.....	20

Trabajo práctico ejercicio 11

Ejecutar el Vivado

Para ejecutar el vivado hay que ir al directorios:

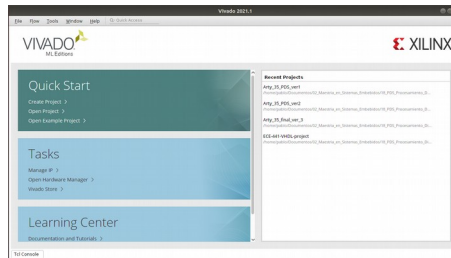
```
cd /tools/Xilinx/Vivado/2021.1/bin y  
escribir sudo ./vivado
```



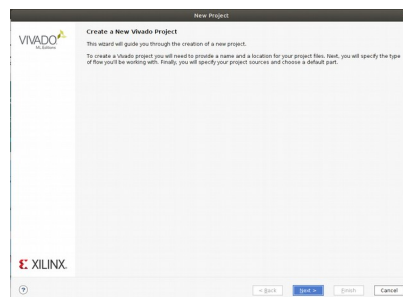
El paso a paso

El profesor trabaja con vivado **2019.2**, y tengo instalado el **2021.1**.

1) Al abrirlo se observa:



2) Se va a: **Create Project**, y hacer click en **Next**.



3) Se selecciona la carpeta(**Project location**)

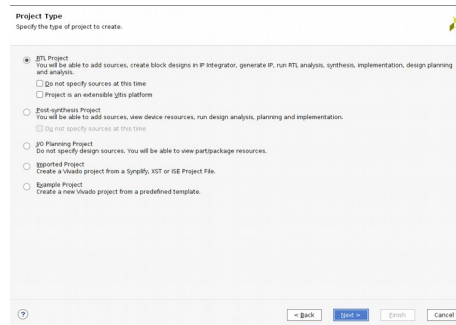
**/home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/
21_Sistemas_Digitales_para_las_Comunicaciones/workspace**

y un nombre del proyecto, que en mi caso será:

MSE-SDC_ejercicio11

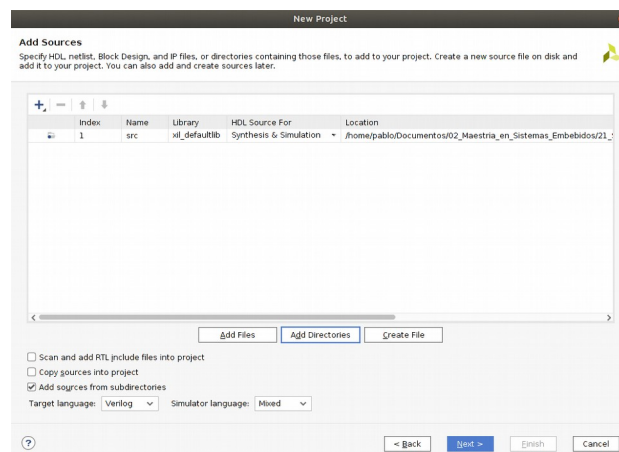
y hacer click en **Next**.

4) Seleccionar proyecto RTL, y hacer click en **Next**.



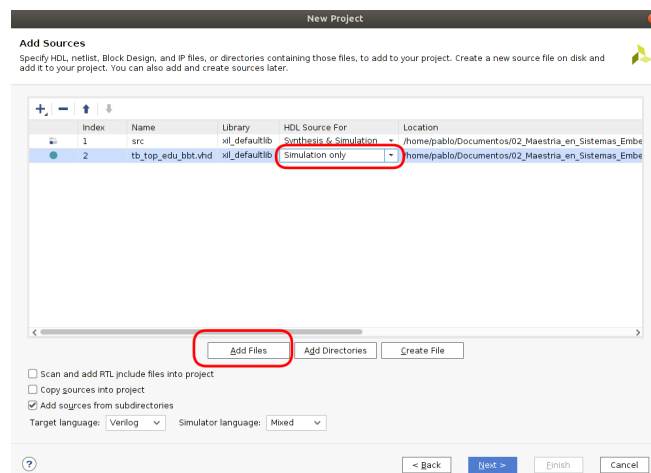
5) Ir al boton **"Add Directories"** y sumar el directorio **"src"**, en mi caso se encuentra en la ruta:

**/home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/
21_Sistemas_Digitales_para_las_Comunicaciones/Repositorio/MSE-SDC-
6Co2021/modem/src**



y hacer click en **Next**.

6) Ir a boton **"Add Files"** y agregamos un solo testbench **"tb_top_edu_bbt"**, y marcarlo como **"Simulation Only"**.



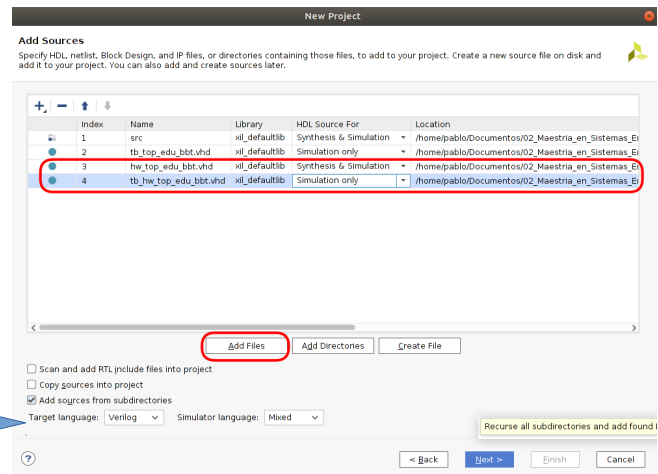
y hacer click en **Next**.

7) Ir a boton **"Add Files"** ir a la carpeta de **"/HW/artyz7-10/edu_bbt"** y agregamos los archivos **"hw_top_edu_bbt.vhd"**, y **"tb_hw_top_edu_bbt.vhd"**, este último lo marcamos como sólo simulación ...

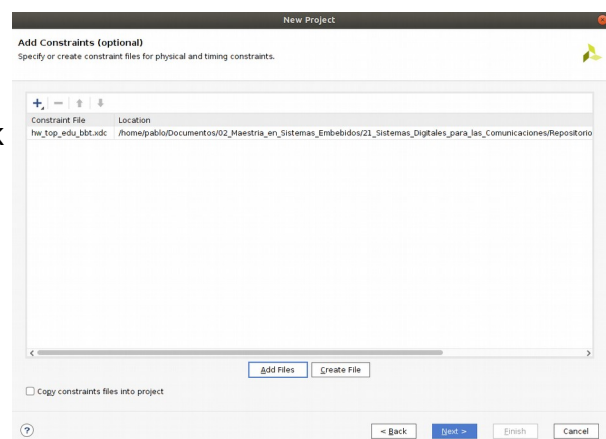
y hacer click en **Next**.

Verificar que el **Target**

sea VHDL

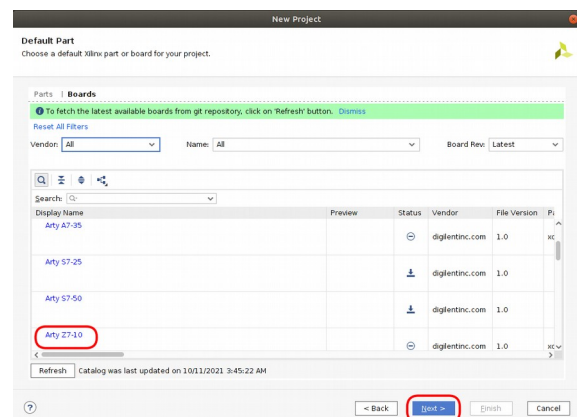


8) Ir al boton **"Add Files"** y agregar el constraints **"hw_top_edu_bbt.xdc"**, y hacer click en **Next**.



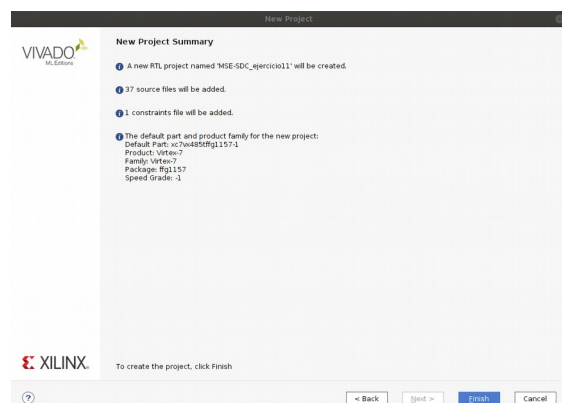
9) Agregamos la placa **"Arty Z7-10"**

y hacer click en **Next**.

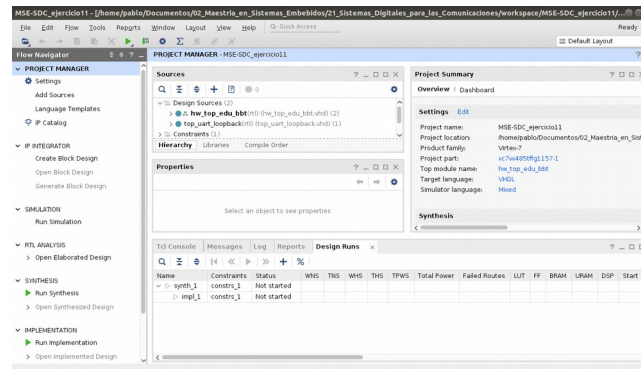


10) El sistema queda:

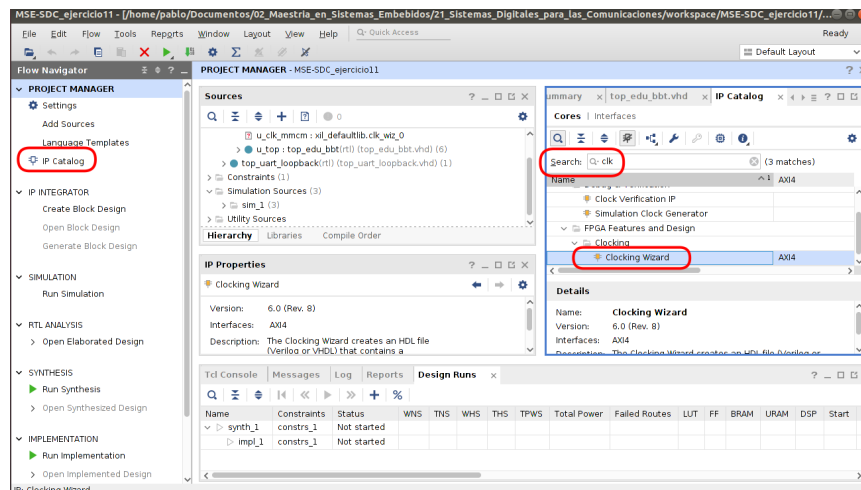
y hacer click en **Next**.



11) Se explica en el video **"Clase5_SDC.mp4"** a las 2 hs y 21 minutos.

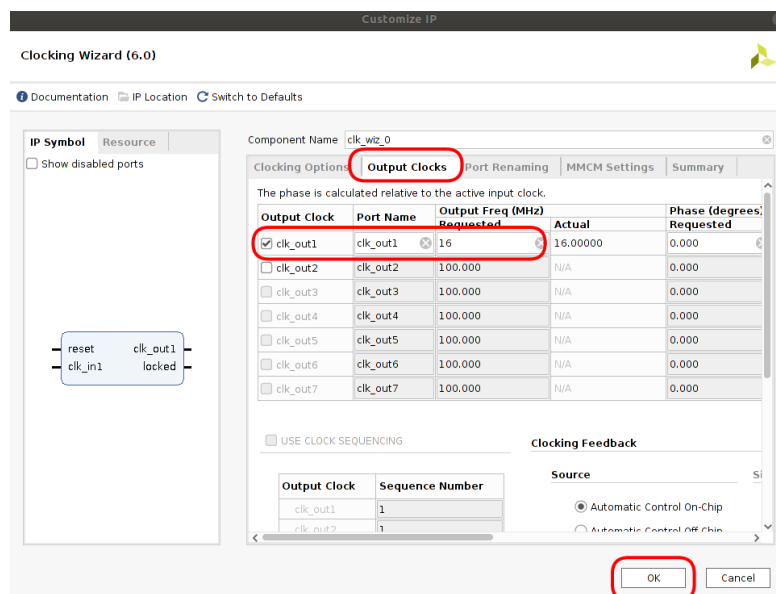


12) Se va a **"IP Catalog"**, en **"Search"** se escribe **clk**, y se elije **"Clocking Wizard"**.



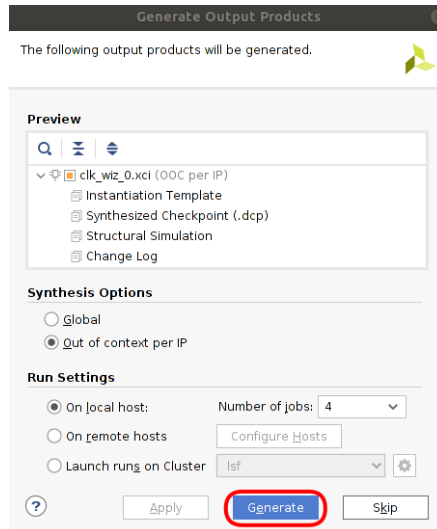
Se hace doble click.

13) Se va ala solapa **"Output Clocks"** y se pone **16MHz** en **clk_out1**, presionando al boton de **OK**.

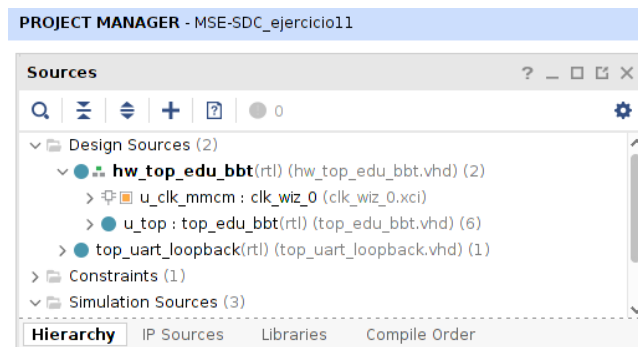


Importante para la Arty Z7: En la solapa **Clock Options** cambiar la frecuencia del clock primario de 100MHz a 125MHz. Si se usa la **Arty A35** dejarlo en 100MHz.

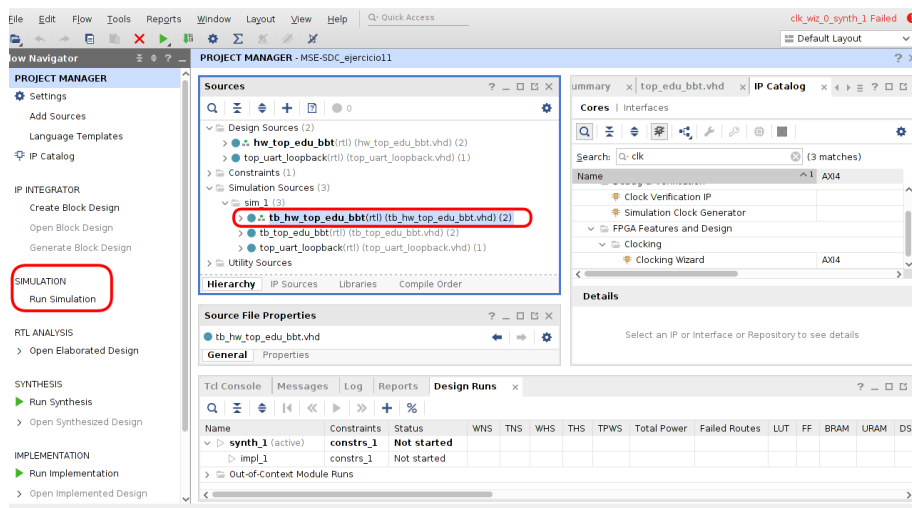
14) Aparece el siguiente cartel, y presionar el boton de **"Generate"**.



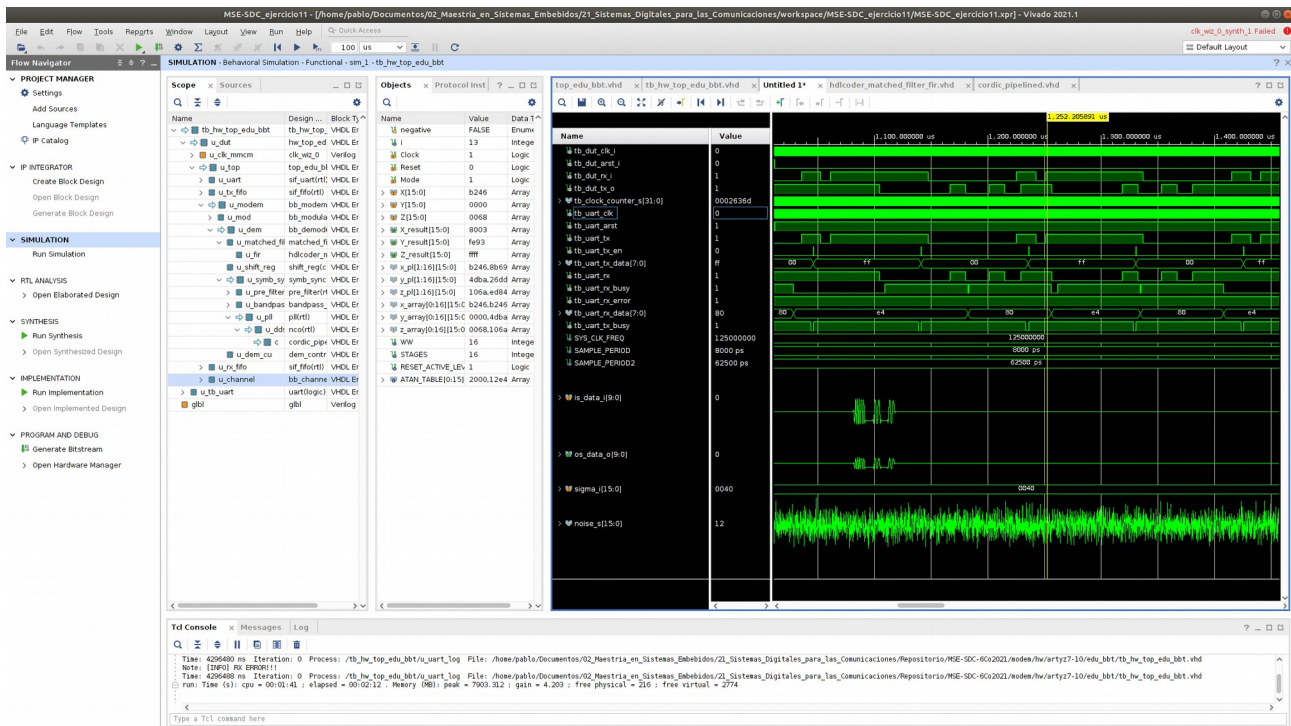
15) Verificar que en la pantalla de Source quede así:



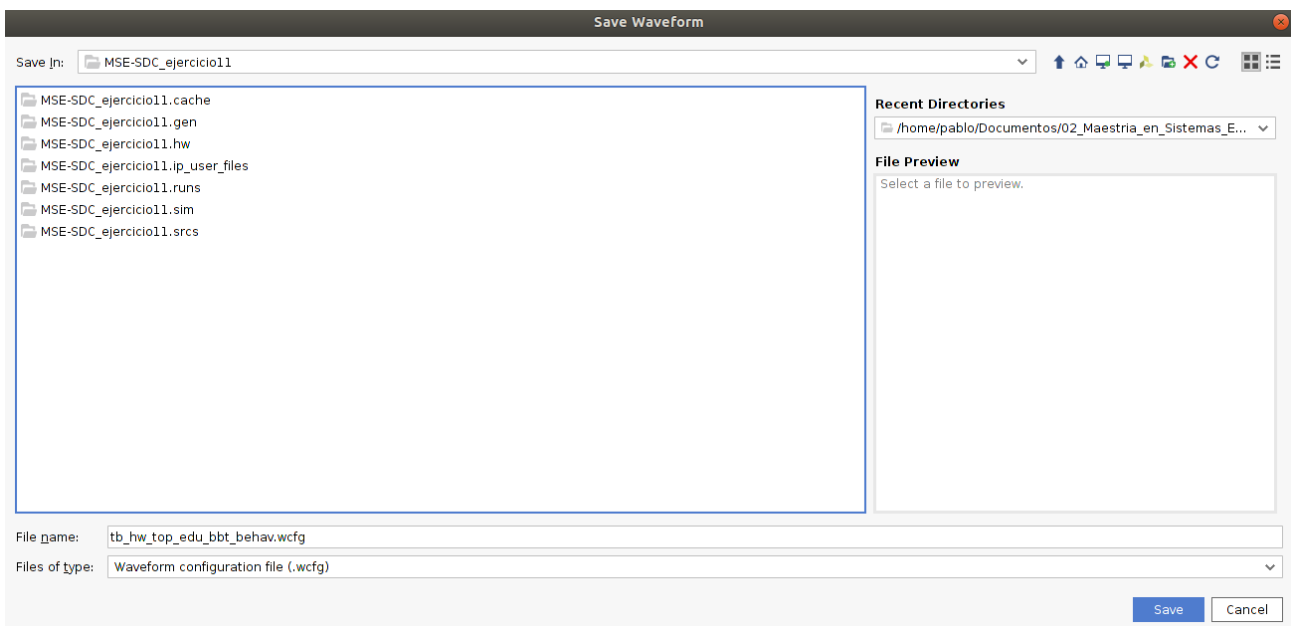
16) Para simular se puede ir a **"Simulation Source"** y posicionarse en el archivo **"tb_hw_top_edu_bbt.vhd"**, ir al menu a la izquierda **"SIMULATION"**, y luego presionar **"Run Simulation"**.



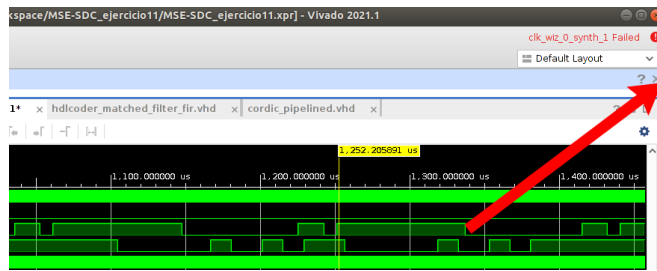
17) Agrego por ejemplo las señales del canal, y simulo de nuevo. Importante: para ver la señales analógicas, posicionaese en la señal con el boton derecho del mouse seleccioner el menú **"Waveform Style"** y luego seleccionar **"Analog"**. IR nuevamente al menú y seleccionar **"Radix"** y seleccionar **"Signed Decimal"**.



18) Si se presiona **"CTRL+S"** se puede guardar las señales



19) Cerramos la simulación con la "x" de la esquina superior derecha.



20) Generar el Bitstream

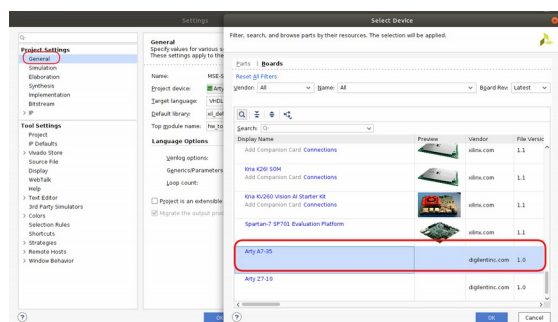
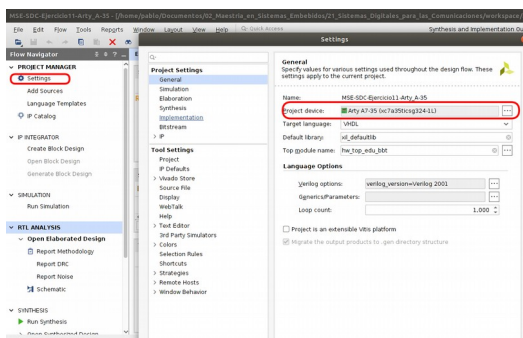
a. Seleccionar el directorio de trabajo:

**/home/pablo/Documents/02_Maestria_en_Sistemas_Embebidos/
21_Sistemas_Digitales_para_las_Comunicaciones/workspace**

21) Como tengo otra placa la ARTY A-35, copio el proyecto.

Para eso se debe ir al menú "File", luego a "Project" y "Save as", escribir en **Project Name: "MSE-SDC-Ejercicio11-Arty_A-35"**, y en **Project Location: "/home/pablo/Documents/02_Maestria_en_Sistemas_Embebidos/21_Sistemas_Digitales_para_las_Comunicaciones/workspace"**

22) Voy a "Settings" y cambio la placa:



22) Mirando el Constraints original lo adapto para la nueva placa, a continuación se muestran las líneas utilizadas:

```
## Clock signal

set_property -dict { PACKAGE_PIN E3      IOSTANDARD LVCMOS33 } [get_ports { clk_i }]; #IO_L12P_T1_MRCC_35
Sch=gclk[100]


## LEDs

set_property -dict { PACKAGE_PIN H5      IOSTANDARD LVCMOS33 } [get_ports { led_o[0] }]; #IO_L24N_T3_35 Sch=led[4]
set_property -dict { PACKAGE_PIN J5      IOSTANDARD LVCMOS33 } [get_ports { led_o[1] }]; #IO_25_35 Sch=led[5]
set_property -dict { PACKAGE_PIN T9      IOSTANDARD LVCMOS33 } [get_ports { led_o[2] }]; #IO_L24P_T3_A01_D17_14
Sch=led[6]

set_property -dict { PACKAGE_PIN T10     IOSTANDARD LVCMOS33 } [get_ports { led_o[3] }]; #IO_L24N_T3_A00_D16_14
Sch=led[7]


## Buttons

set_property -dict { PACKAGE_PIN D9      IOSTANDARD LVCMOS33 } [get_ports { arst_i }]; #IO_L6N_T0_VREF_16 Sch=btn[0]


## USB-UART Interface

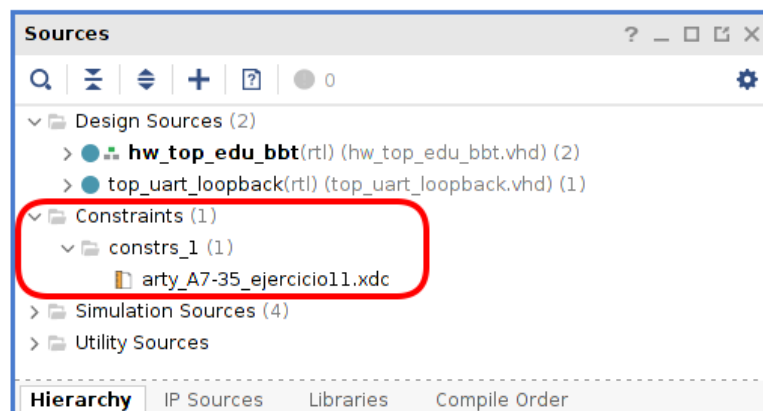
set_property -dict { PACKAGE_PIN D10     IOSTANDARD LVCMOS33 } [get_ports { tx_o }]; #IO_L19N_T3_VREF_16
Sch=uart_rxd_out

set_property -dict { PACKAGE_PIN A9      IOSTANDARD LVCMOS33 } [get_ports { rx_i }]; #IO_L14N_T2_SRCC_16
Sch=uart_txd_in
```

se lo guarda en un archivo **"arty_A7-35_ejercicio11.xdc"**.

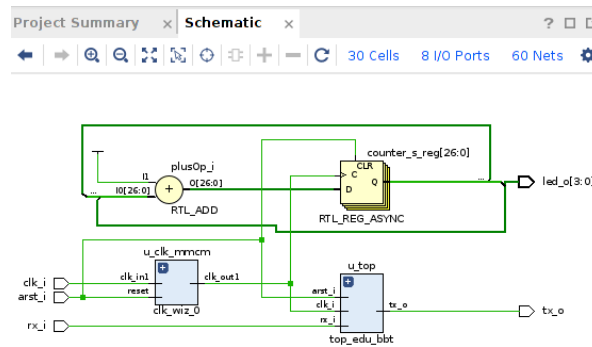
23) A continuación descargar el viejo Constraints **"hw_top_edu_bbt.xdc"**, y cargar el nuevo archivo Constraints **"arty_A7-35_ejercicio11.xdc"**.

El **"Source"** debe quedar:



24) Verificar todo, eso implica:

- Correr la simulación
- Correr el análisis RTL, que debe dar:

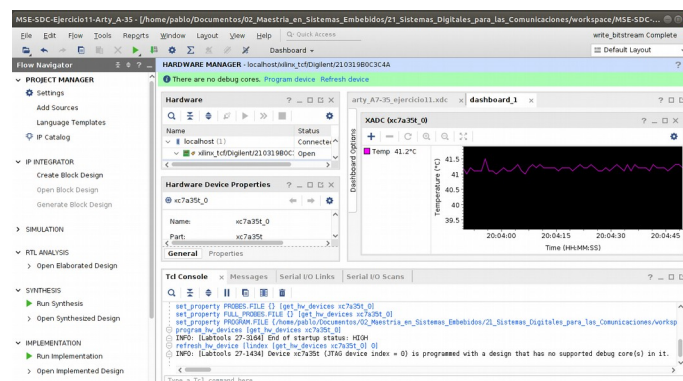


- Sintetizar, y verificar que no haya errores.
- Correr la implementación, y verificar los recursos.
- Generar el Bitstream.
- Programar la placa Artty A-35.

Importante: cuando se hace todo esto abrir el “Monitor de Sistema” de Ubuntu, y verificar la **Memoria de Intercambio**, si esta se llena el sistema operativo se tilda.

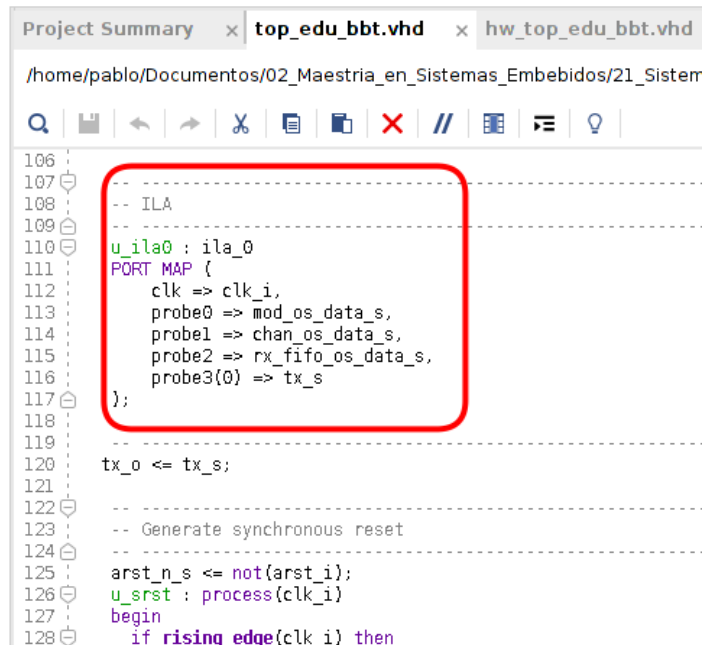
Para programar la placa ir al directorio `"/workspace/MSE-SDC-Ejercicio11-Arty_A-35/MSE-SDC-Ejercicio11-Arty_A-35.runs/impl_1"`, y bucar el archivo `"hw_top_edu_bbt.bit"`.

25) Abrir una terminal como por ejemplo GtkTerm configurar en **115200** baudios y mandar algunos caracteres.



Configurar el ILA (Integrated Logic Analyzer)

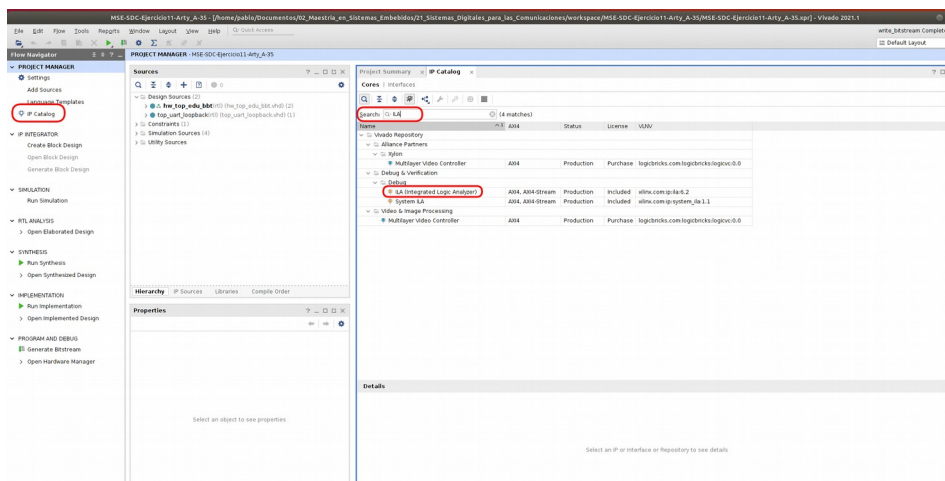
En el archivo "top_edu_bbt.vhd", ya se tiene un ILA precargado:



```
106
107
108 -- ILA
109
110 u_ila0 : ila_0
111 PORT MAP (
112     clk => clk_i,
113     probe0 => mod_os_data_s,
114     probe1 => chan_os_data_s,
115     probe2 => rx_fifo_os_data_s,
116     probe3(0) => tx_s
117 );
118
119
120 tx_o <= tx_s;
121
122
123 -- Generate synchronous reset
124
125 arst_n_s <= not(arst_i);
126 u_srst : process(clk_i)
127 begin
128     if rising_edge(clk_i) then
```

Faltaría agregar el en el **IP Catalog** el módulo de referencia, respetando la señales declaradas:

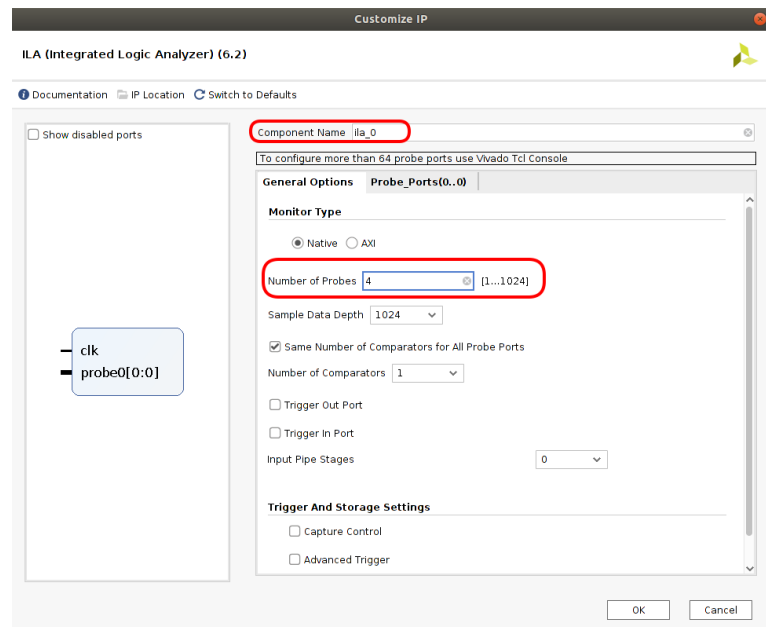
1) Ir al manú izquierdo de la aplicación a la opción "IP Catalog", poner en el vbuscador la palabra "ILA", y agregar el módulo:



2) Al hacer doble click sobre se abre la siguiente pantalla:

ILA (Integrated Logic Analyzer)

Verificar que el nombre del módulo sea **ila_0** , y que el número de pruebas sea **4**.

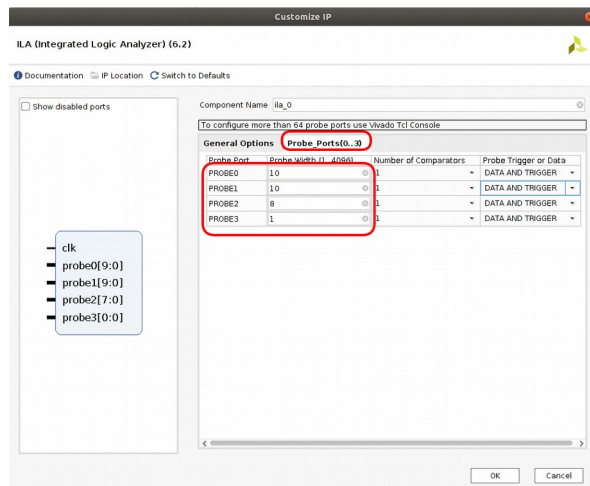


3) Pasar a la siguiente solapa **"Probe_Ports"**, y haer coincidir cada punto de prueba con la longitud de la variable del archivo **"top_edu_bbt.vhd"**.

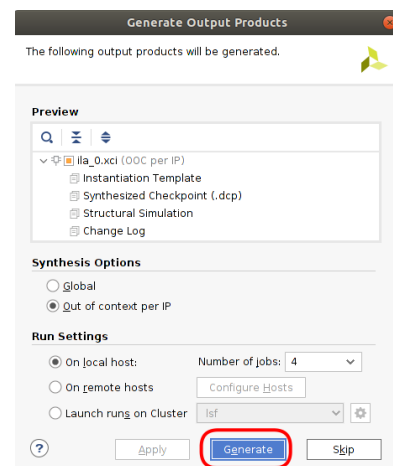
```

93 COMPONENT ila_0
94 PORT (
95     clk : IN STD_LOGIC;
96     probe0 : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
97     probe1 : IN STD_LOGIC_VECTOR(9 DOWNTO 0);
98     probe2 : IN STD_LOGIC_VECTOR(7 DOWNTO 0);
99     probe3 : IN STD_LOGIC_VECTOR(0 DOWNTO 0);
100 );
101 END COMPONENT ;
102
103
104 begin
105
106
107
108 -- ILA
109
110 u_ila0 : ila_0
111 PORT MAP (
112     clk => clk_i,
113     probe0 => mod_os_data_s,
114     probe1 => chan_os_data_s,
115     probe2 => rx_fifo_os_data_s,
116     probe3(0) => tx_s
117 );
118
119
120

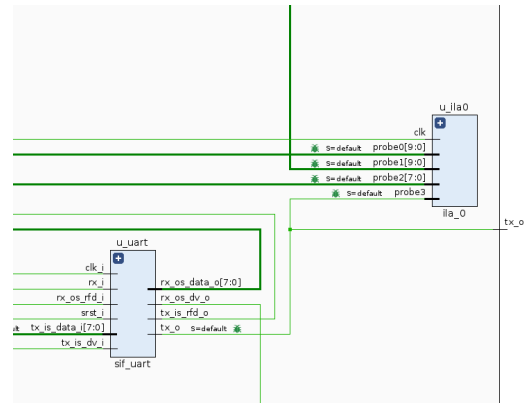
```



4) Al hacer click en el boton **"OK"**, se abre el siguiente cuadro y clickear en el boton **"Generate"**.



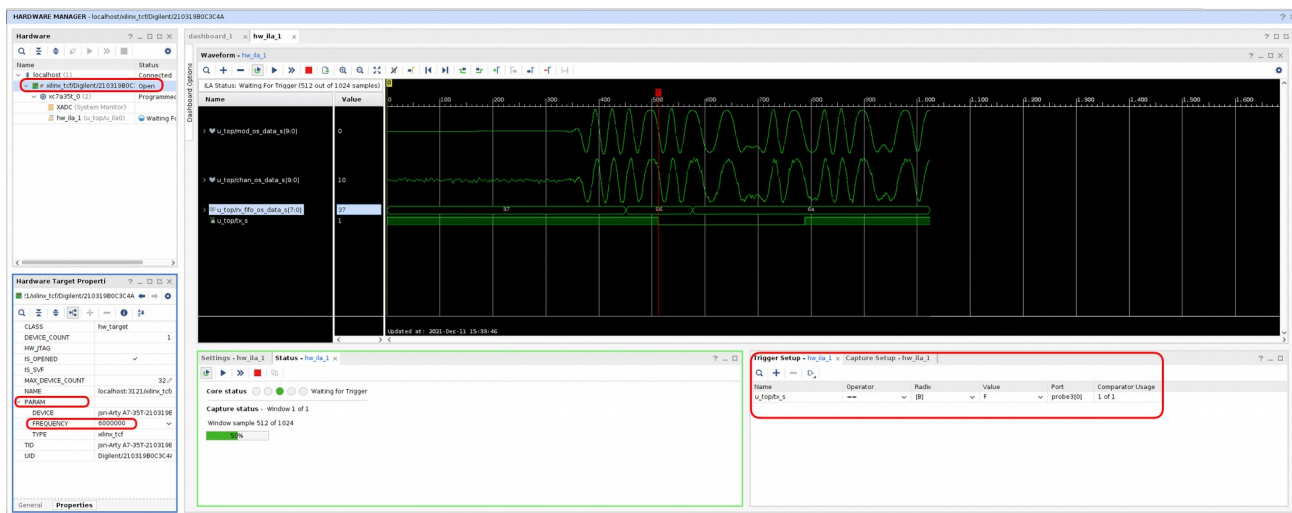
5) Si se corre el análisis RTL, verificar que el módulo ILA este conectado:



6) Generar el Bitstream, programar la FPGA y verificar que el ILA anda funcionando.

a. Acordarse de cambiar la frecuencia de refresco, que debe ser menor que la mitad de los 16MHz. En este caso se puso 6MHz.

b. Configurar el disparo, que para este caso se lo configuró con flanco descendente de **tx_s**.



Configurar el VIO (Virtual Input/Output)

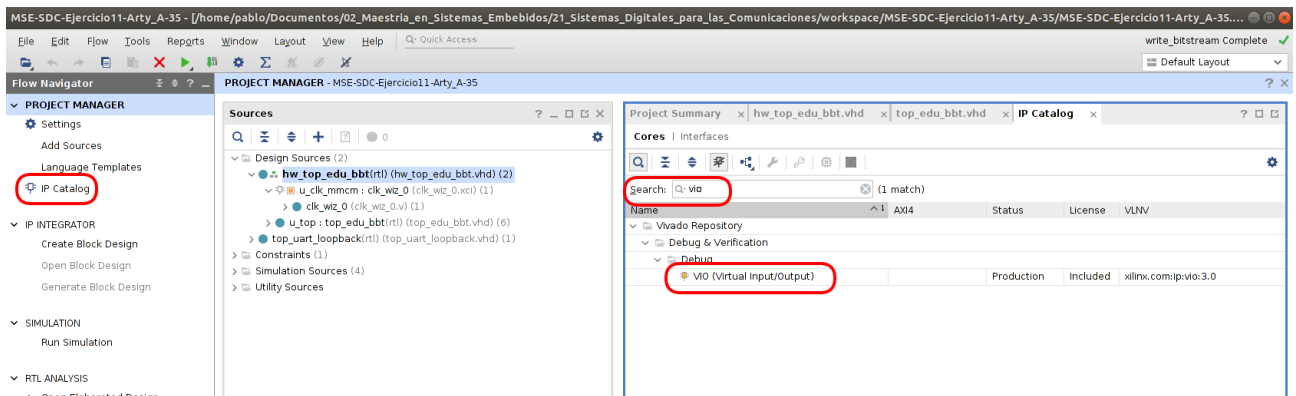
A continuación se va a agregar un VIO que maneje en principio la configuraci3n del modem de las siguientes variables:

sigma_i ; nm1_pre_i ; nm1_bytes_i

ambas estan definidas en el archivo "top_edu_bbt.vhd", y actualmente se coargan con unas constantes definidas en el mismo archivo con nombres:

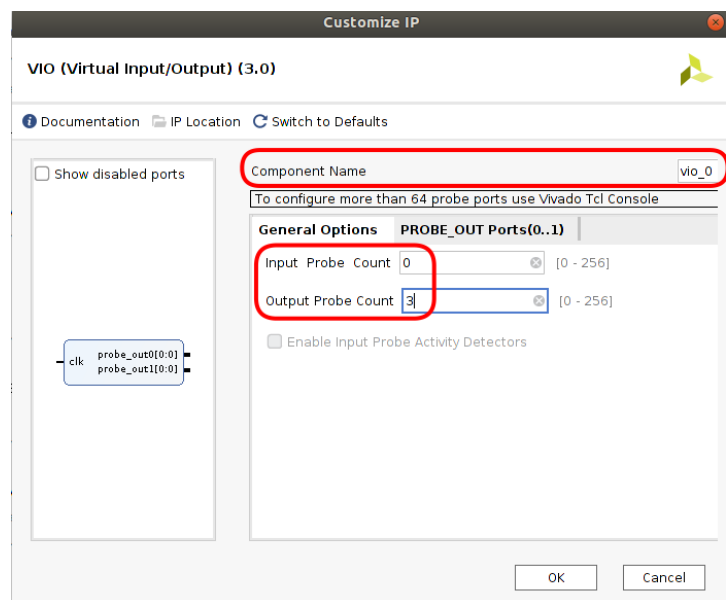
sigma_c ; nm1_pre_c ; nm1_bytes_c

1) Agregamos el VIO, para eso nos dirigimos a "IP Catalog" y en el buscador ponemos "VIO".



2) Haciendo doble click en **VIO (Virtual Input/Output)** se habre la siguiente pantalla:

Verificar que el nombre del componente sea **vio_0**, y setear la cntdad de entradas en **cer0** y las salidas en **tres**.

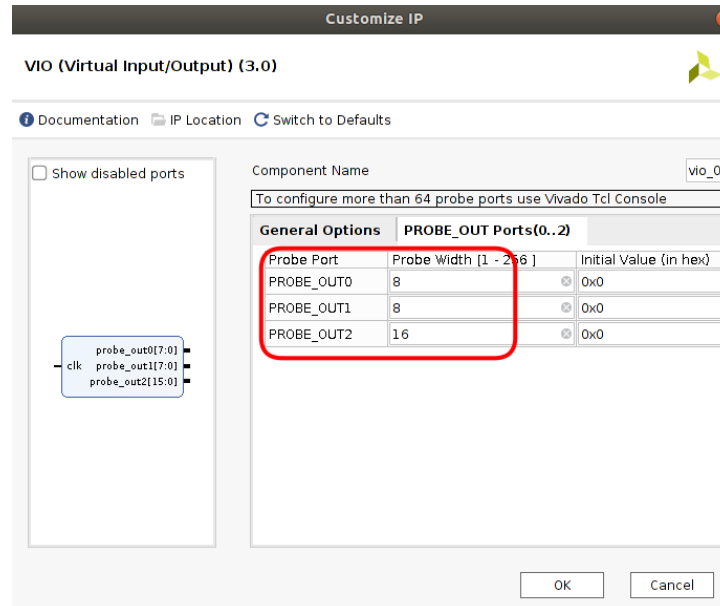


3) Ir a la solapa "PROBE_OUT Ports", y seleccionar la longitud. La longitud se pueden sacar del archivo anteriormente mencionado:

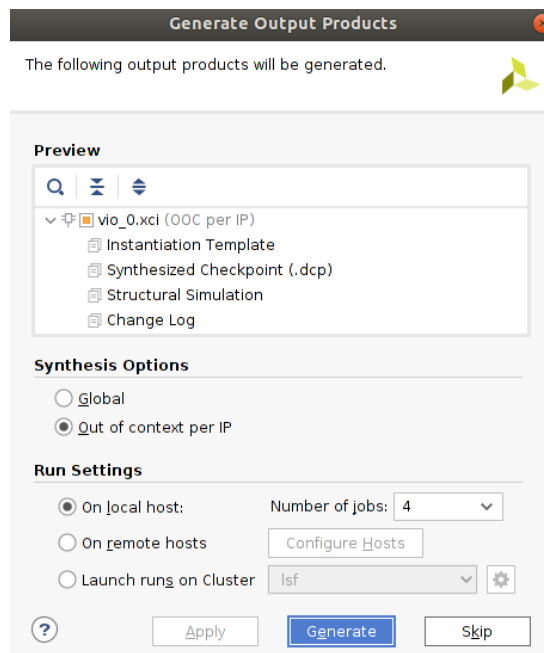
```

75  -- Modem config
76  constant nml_bytes_c : std_logic_vector( 7 downto 0) := X"03";
77  constant nml_pre_c   : std_logic_vector( 7 downto 0) := X"07"; -- Original era 07
78  constant nml_sfd_c   : std_logic_vector( 7 downto 0) := X"03"; -- Original era 03
79  constant det_th_c    : std_logic_vector(15 downto 0) := X"0040";
80  constant pll_kp_c    : std_logic_vector(15 downto 0) := X"A000";
81  constant pll_ki_c    : std_logic_vector(15 downto 0) := X"9000";
82  -- Channel config
83  constant sigma_c     : std_logic_vector(15 downto 0) := X"0040"; -- Q16.12
84

```



4) Al presionar el boton de "OK", aparece la siguiente pantalla que se deberá presionar el boton "Generate"



Al presionar OK en la nueva ventana emergente. En este momento comienza la síntesis del bloque que se acaba de configurar.

5) Agregamos a la arquitectura del archivo “**top_edu_bbt.vhd**” las señales para poder conectar la salida del VIO, y al mismo tiempo asignarlas, tambien ponemos sus valores iciales,y agregamos el componente VIO:

```

Project Summary x IP Catalog x vio_0.vhd x top_edu_bbt.vhd * x vio_0.v x
/home/pablo/Documentos/02_Maestria_en_Sistemas_Embebidos/21_Sistemas_Digitales_para_las_Comunicaciones/Repositorio/MSE- x

sigma_c Next Previous Highlight Match Case Whole Words Reached bottom of the page.

72 signal chan_os_dv_s : std_logic;
73 signal chan_os_rfd_s : std_logic;
74
75 -- Modem config
76 constant nml_bytes_c : std_logic_vector( 7 downto 0) := X"03";
77 constant nml_pre_c : std_logic_vector( 7 downto 0) := X"07"; -- Original era 07
78 constant nml_sfd_c : std_logic_vector( 7 downto 0) := X"03"; -- Original era 03
79 constant det_th_c : std_logic_vector(15 downto 0) := X"0040";
80 constant pll_kp_c : std_logic_vector(15 downto 0) := X"A000";
81 constant pll_ki_c : std_logic_vector(15 downto 0) := X"9000";
82 -- Channel config
83 constant sigma_c : std_logic_vector(15 downto 0) := X"0040"; -- Q16.12
84
85 -----
86
87 -----
88
89 -- DEBUG SIGNALS
90 -----
91 -- ILA
92 signal tx_s : std_logic;
93 -- ILA component
94 COMPONENT ila_0
95 PORT (
96     clk : IN STD_LOGIC;
97     probe0 : IN STD_LOGIC_VECTOR(9 DOWNT0 0);
98     probe1 : IN STD_LOGIC_VECTOR(9 DOWNT0 0);
99     probe2 : IN STD_LOGIC_VECTOR(7 DOWNT0 0);
100     probe3 : IN STD_LOGIC_VECTOR(0 DOWNT0 0)
101 );
102 END COMPONENT ;
103
104 -- VIO Signals
105 signal nml_bytes_sig : std_logic_vector( 7 downto 0) := nml_bytes_c;
106 signal nml_pre_sig : std_logic_vector( 7 downto 0) := nml_pre_c; -- Original era 07
107 signal sigma_sig : std_logic_vector(15 downto 0) := sigma_c; -- Q16.12
108 -- VIO component
109 COMPONENT vio_0
110 PORT (
111     CLK : IN STD_LOGIC;
112     probe_out0 : OUT STD_LOGIC_VECTOR(7 DOWNT0 0) ;
113     probe_out1 : OUT STD_LOGIC_VECTOR(7 DOWNT0 0) ;
114     probe_out2 : OUT STD_LOGIC_VECTOR(15 DOWNT0 0)
115 );
116 END COMPONENT ;
117

```

6) Instanciamos el VIO:

```

120 begin
121
122     -- ILA
123
124     u_ila0 : ila_0
125     PORT MAP (
126         clk => clk_i,
127         probe0 => mod_os_data_s,
128         probe1 => chan_os_data_s,
129         probe2 => rx_fifo_os_data_s,
130         probe3(0) => tx_s
131     );
132
133     -- VIO
134
135     u_vio : vio_0
136     PORT MAP (
137         clk => clk_i,
138         probe_out0 => nml_bytes_sig,
139         probe_out1 => nml_pre_sig,
140         probe_out2 => sigma_sig
141     );
142
143     tx_o <= tx_s;
144
145

```

7) En todo lugar que figure las constantes **"sigma_c ; nm1_pre_c ; nm1_bytes_c"** debemos reemplazarlas por **"sigma_sig ; nm1_pre_sig ; nm1_bytes_sig"** :

Para sigma:

```
335 | -----
336 | -- Channel
337 | -----
338 | u_channel : bb_channel
339 | port map
340 | {
341 |     -- clk, en, rst
342 |     clk_i      => clk_i,
343 |     en_i       => '1',
344 |     srst_i     => srst_s,
345 |     -- Input Stream
346 |     is_data_i  => mod_os_data_s,
347 |     is_dv_i    => mod_os_dv_s,
348 |     is_rfd_o   => mod_os_rfd_s,
349 |     -- Output Stream
350 |     os_data_o  => chan_os_data_s,
351 |     os_dv_o    => chan_os_dv_s,
352 |     os_rfd_i   => chan_os_rfd_s,
353 |     -- Control
354 |     sigma_i    => sigma_sig
355 | },
356 | -----
```

Para nm1_pre y nm1_bytes:

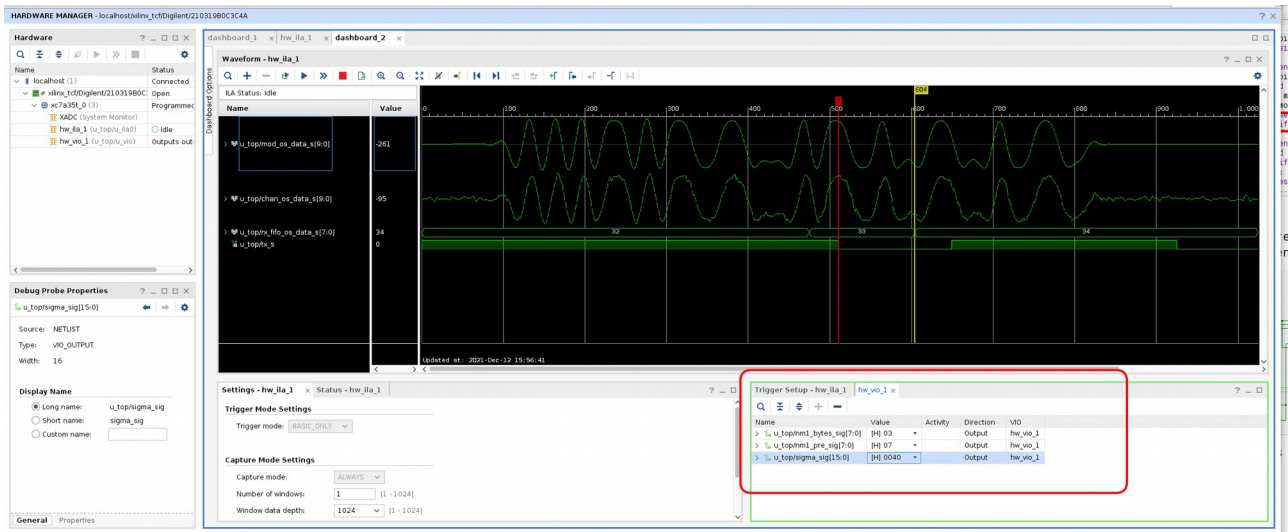
```
260 | -----
261 | -- Modem
262 | -----
263 | -- I want to keep the modem_is... signals, in case new blocks are added
264 | -- between the tx_fifo and the modem.
265 | modem_is_data_s  <= tx_fifo_os_data_s;
266 | modem_is_dv_s    <= tx_fifo_os_dv_s;
267 | tx_fifo_os_rfd_s <= modem_is_rfd_s;
268 | -- Modem module
269 | u_modem : bb_modem
270 | port map
271 | {
272 |     -- clk, en, rst
273 |     clk_i      => clk_i,
274 |     en_i       => '1',
275 |     srst_i     => srst_s,
276 |     -- Input Stream
277 |     is_data_i  => modem_is_data_s,
278 |     is_dv_i    => modem_is_dv_s,
279 |     is_rfd_o   => modem_is_rfd_s,
280 |     -- Output Stream
281 |     os_data_o  => modem_os_data_s,
282 |     os_dv_o    => modem_os_dv_s,
283 |     os_rfd_i   => modem_os_rfd_s,
284 |     -- DAC Stream
285 |     dac_os_data_o => mod_os_data_s,
286 |     dac_os_dv_o  => mod_os_dv_s,
287 |     dac_os_rfd_i => mod_os_rfd_s,
288 |     -- ADC Stream
289 |     adc_is_data_i => chan_os_data_s,
290 |     adc_is_dv_i  => chan_os_dv_s,
291 |     adc_is_rfd_o => chan_os_rfd_s,
292 |     -- Config
293 |     nm1_bytes_i => nm1_bytes_sig,
294 |     nm1_pre_i   => nm1_pre_sig,
295 |     nm1_srd_i   => nm1_srd_c,
296 |     det_th_i    => det_th_c,
297 |     pll_kp_i    => pll_kp_c,
298 |     pll_ki_i    => pll_ki_c,
299 |     -- Control
300 |     send_i      => modem_send_s,
301 |     -- State
302 |     tx_rdy_o    => modem_tx_rdy_s,
303 |     rx_ovf_o    => modem_rx_ovf_s
304 | };
```

```

222  -----
223  -- SEND CONTROL: send_s signal logic
224  -----
225  u_send_logic : process(clk_i)
226  begin
227      if rising_edge(clk_i) then
228          if srst_s = '1' then
229              pipe_data_counter_s <= (others => '0');
230              modem_send_s <= '0';
231              -- modem_tx_rdy_d10_s <= (others => '0');
232          else
233              if uart_os_dv_s = '1' and
234                 uart_os_rfd_s = '1' and
235                 modem_is_dv_s = '1' and
236                 modem_is_rfd_s = '1'
237              then
238                  pipe_data_counter_s <= pipe_data_counter_s;
239              elsif modem_is_dv_s = '1' and
240                    modem_is_rfd_s = '1'
241              then
242                  pipe_data_counter_s <= std_logic_vector(unsigned(pipe_data_counter_s)-1);
243              elsif uart_os_dv_s = '1' and
244                    uart_os_rfd_s = '1'
245              then
246                  pipe_data_counter_s <= std_logic_vector(unsigned(pipe_data_counter_s)+1);
247              end if;
248              if modem_send_s = '1' then
249                  modem_send_s <= '0';
250              else
251                  if unsigned(pipe_data_counter_s) > unsigned(nml_bytes_sig) and modem_tx_rdy_s = '1' then
252                      modem_send_s <= '1';
253                  end if;
254              end if;
255          end if;
256      end if;
257  end process;
258  -----

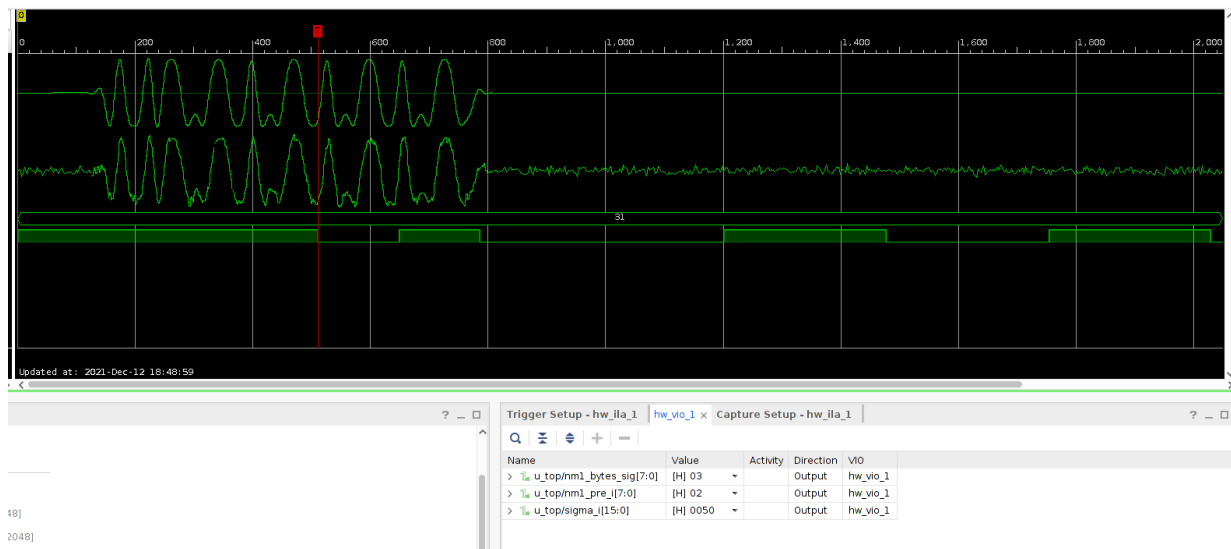
```

9) A continuación se muestra el VIO en funcionamiento

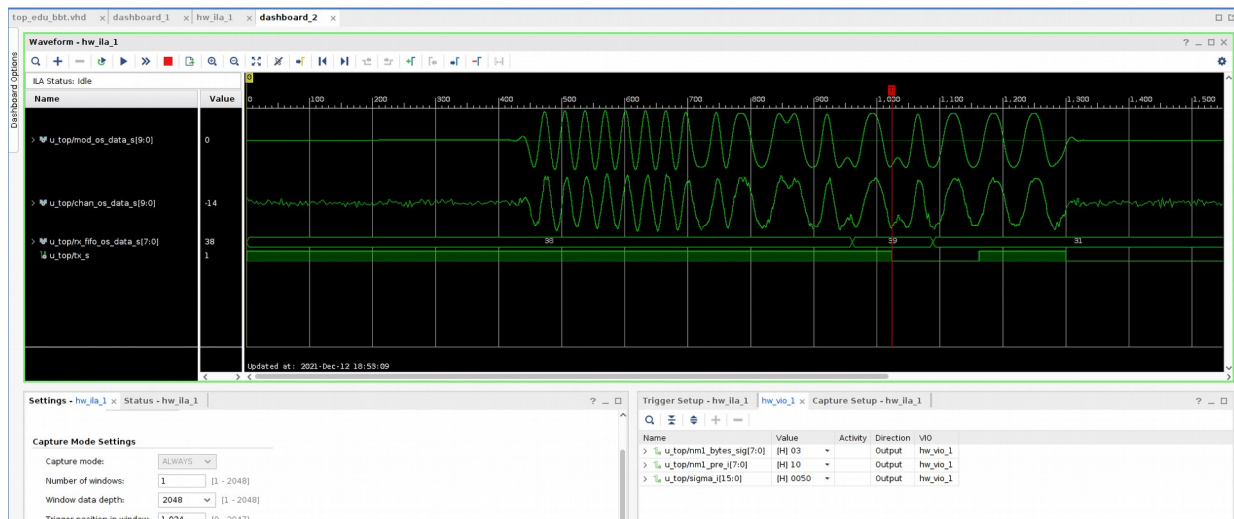


Pruebas

a) Preámbulo de 3, Bytes=4, y Ruido=0x50



b) Preámbulo de 16, Bytes=4, y Ruido=0x50



c) Preámbulo de 16, Bytes=4, y Ruido=0x100

