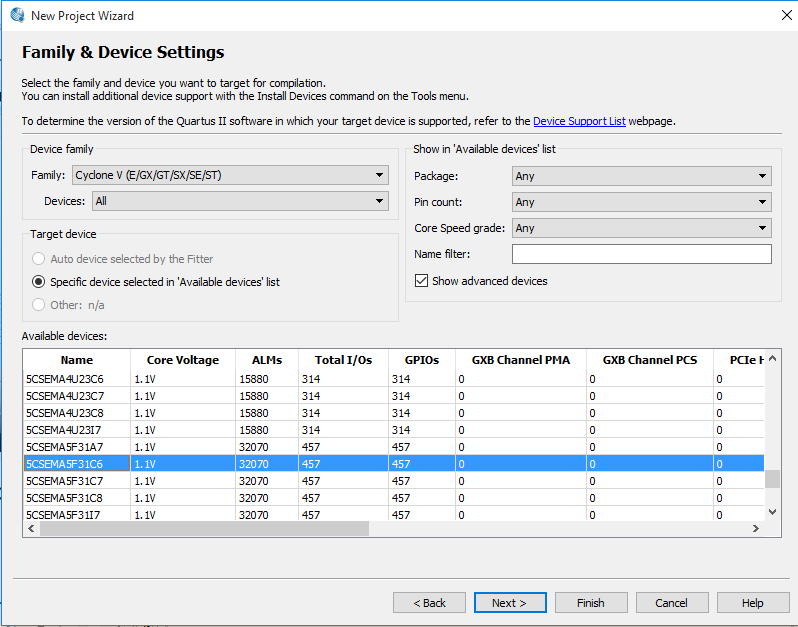
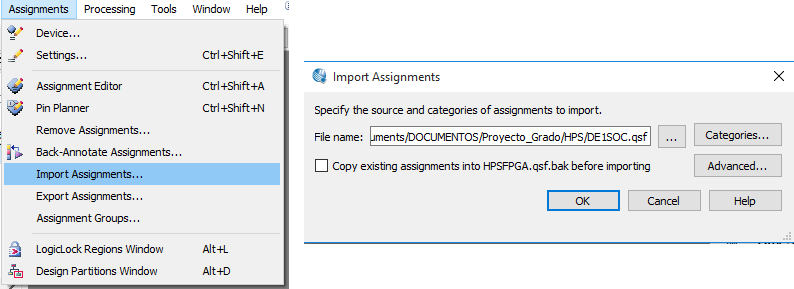
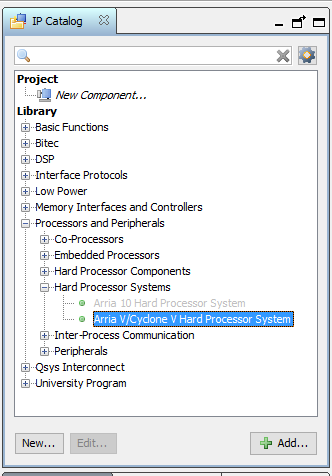
**Como crear un HPS (Hard Processor System)**

1. Creamos un nuevo proyecto en Quartus II con el nombre HPSFPGA seleccionando la familia Cyclone V y el dispositivo (5CSEMA5F31C6).
2. Creado el proyecto añadimos un archivo de la siguiente forma:
   1. Dar clic en Assignments
   2. Dar clic en Import Assignments
   3. Buscamos el archivo DE1SOC.qsf
   4. Seleccionar el archivo DE1SOC.qsf
   5. Deshabilitamos la opción copiar asignaciones existentes en HPSFPGA. Qsf
   6. Dar clic en OK.

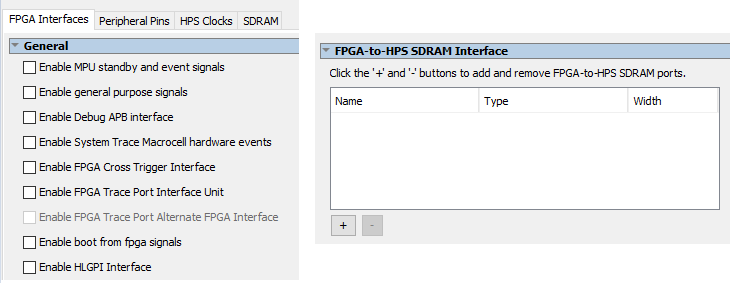
Como se muestra en la siguiente imagen:

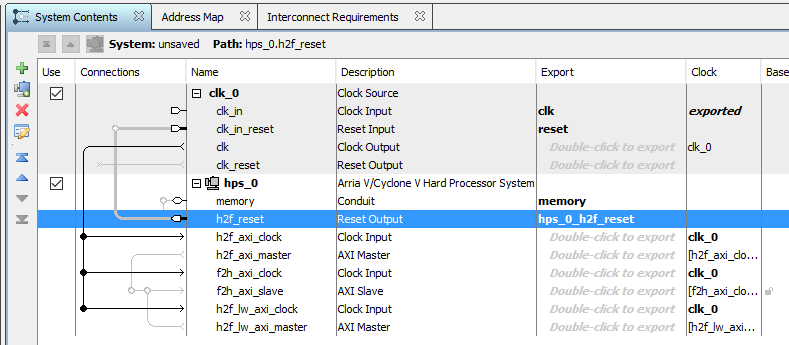


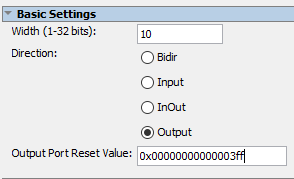
1. Abrimos la herramienta Qsys donde en catalogo IP seleccionamos Processors and Peripherals luego Hard Processor Systems y seleccionamos Cyclone V.

Configuramos el HPS:

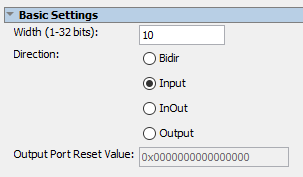
* En la pestaña FPGA Interfaces nos dirigimos a la configuración general donde deshabilitamos la espera MPU y señal de evento.
* Para este proyecto no es necesario FPGA-HPS SDRAM Interface eliminamos presionando el botón [-], luego finalizar.



En la ventana de System Contents hacemos las siguientes conexiones: el clk se une con h2f\_axi\_ clock también con f2h\_axi\_clock y f2h\_lw\_axi\_clock. Ahora exportamos h2f reset output.

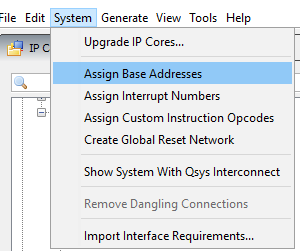
Agregamos un PIO (Parallel I/O) para LEDs y lo configuramos para que sean salidas de un ancho de 10 bits además de la configuración Output Port Reset Value: 0x00000000000003ff y finalizamos.

Hacemos las conexiones: el clk con el clk de la FPGA, el reset con clk\_reset y s1 (Avalon Memory Mapped Slave) con h2f\_lw\_master.

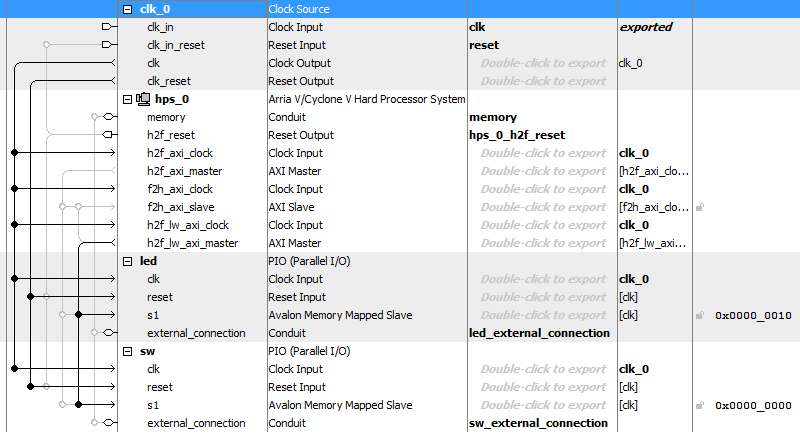
Agregamos otro PIO (Parallel I/O) para Swiches y lo configuramos para que sean entradas de un ancho de 10 bits y finalizamos.

Hacemos las mismas conexiones: el clk con el clk de la FPGA, el reset con clk\_reset y s1 (Avalon Memory Mapped Slave) con h2f\_lw\_master.

Exportamos los external\_connection del pio\_0 y del pio\_1, luego renombramos el pio\_0 como led y el pio\_1 como sw.

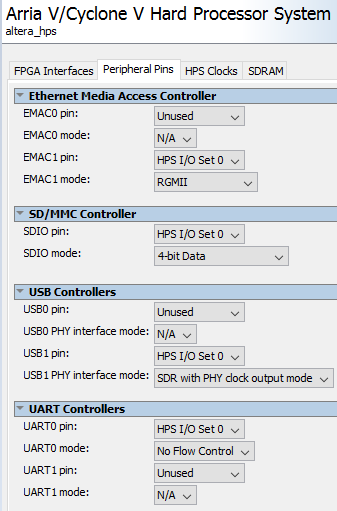
Ahora asignamos base address ingresando a System => Assign Base Addresses como lo indica la imagen.

La conexión de los PIO (Parallel I/O) junto con clk y el módulo hps es lista además las direcciones están asignadas.



Entramos a los parámetros del hps luego nos ubicamos en Peripheral Pins

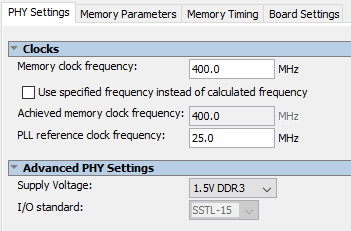
* En la parte de Ethernet Media Access Controller modificamos **EMAC1 pin** escojiendo la opción HPS I/O Set 0, además el parámetro **EMAC1 mode** se selecciona RGMII.
* En los parámetros de SD/MMC Controller escogemos para **SDIO pin:** HPS I/O Set 0 y para **SDIO mode:** 4-Bit Data.
* Para USB Controllers **USB1 pin:** HPS I/O Set 0 y **USB1 PHY interface mode:** SDR with PHY clock output mode.
* Para UART Controllers **UART0 pin:** HPS I/O Set 0 y en **UART0 mode:** No Flow Control.



Los parámetros de HPS Clock se deben dejar sin cambios.

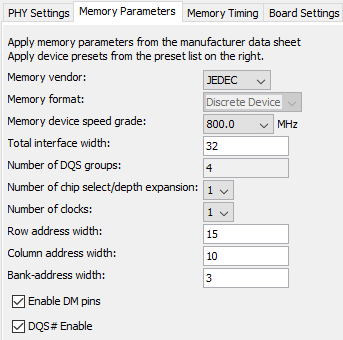
Ahora nos dirigimos a los parámetros de SDRAM para configurar la sincronización SDRAM.

* Pestaña PHY Settings 🡪 Clocks 🡪**Memory clock frequency:** 400 MHz

🡪**PLL reference clock frequency:** 25 MHz

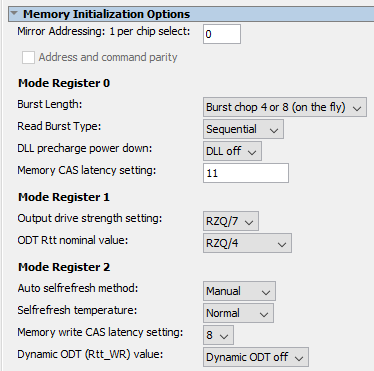
* Pestaña Memory Parameters 🡪 **Memory device speed grade:** 800 MHz

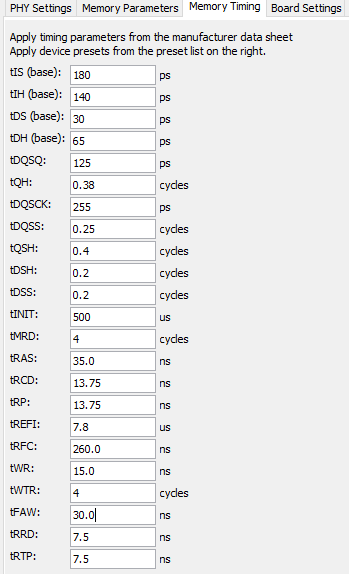
**🡪 Total interface width:** 12

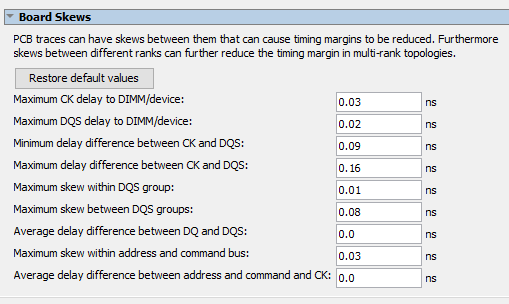
* **Row address width:** 15
* **Column address width:** 10
* **Memory Initialization Options🡪 Memory CAS latency setting :** 11

**🡪 Output drive strength setting:** RZQ/7

**🡪ODT Rtt nominal value:** RZQ/4

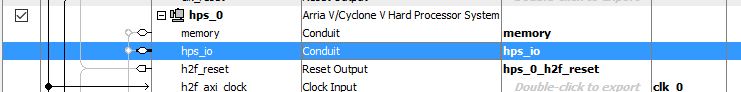
**🡪 Memory write CAS latency setting:** 8

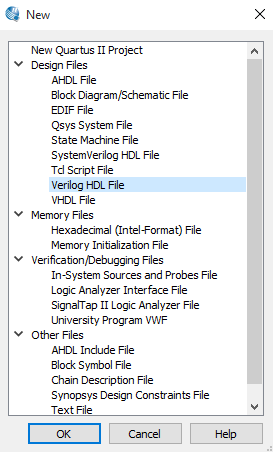
En los parámetros de Memory Timing determinamos estos valores:

Ahora en la pestaña Board Settings 🡪 Board Skews cambiamos los valores:

Una vez terminado la configuración de parámetros se da finalizar.

Luego extraemos el hps\_io para después dar clic en finalizar:



1. Continuamos en añadir el HPS- componente en el diseño de FPGA. Crear nueva Verilog HDL File.

En el nuevo archivo escribimos el siguiente código:

//=======================================================

// This code is generated by Terasic System Builder

//=======================================================

module HPSFPGA(

//////////// CLOCK //////////

input CLOCK\_50,

input CLOCK2\_50,

input CLOCK3\_50,

input CLOCK4\_50,

//////////// KEY //////////

input [3:0] KEY,

//////////// LED //////////

output [9:0] LEDR,

//////////// SW //////////

input [9:0] SW,

//////////// HPS //////////

inout HPS\_CONV\_USB\_N,

output [14:0] HPS\_DDR3\_ADDR,

output [2:0] HPS\_DDR3\_BA,

output HPS\_DDR3\_CAS\_N,

output HPS\_DDR3\_CK\_N,

output HPS\_DDR3\_CK\_P,

output HPS\_DDR3\_CKE,

output HPS\_DDR3\_CS\_N,

output [3:0] HPS\_DDR3\_DM,

inout [31:0] HPS\_DDR3\_DQ,

inout [3:0] HPS\_DDR3\_DQS\_N,

inout [3:0] HPS\_DDR3\_DQS\_P,

output HPS\_DDR3\_ODT,

output HPS\_DDR3\_RAS\_N,

output HPS\_DDR3\_RESET\_N,

input HPS\_DDR3\_RZQ,

output HPS\_DDR3\_WE\_N,

output HPS\_ENET\_GTX\_CLK,

inout HPS\_ENET\_INT\_N,

output HPS\_ENET\_MDC,

inout HPS\_ENET\_MDIO,

input HPS\_ENET\_RX\_CLK,

input [3:0] HPS\_ENET\_RX\_DATA,

input HPS\_ENET\_RX\_DV,

output [3:0] HPS\_ENET\_TX\_DATA,

output HPS\_ENET\_TX\_EN,

inout [3:0] HPS\_FLASH\_DATA,

output HPS\_FLASH\_DCLK,

output HPS\_FLASH\_NCSO,

inout [1:0] HPS\_GPIO,

inout HPS\_GSENSOR\_INT,

inout HPS\_I2C\_CONTROL,

inout HPS\_I2C1\_SCLK,

inout HPS\_I2C1\_SDAT,

inout HPS\_I2C2\_SCLK,

inout HPS\_I2C2\_SDAT,

inout HPS\_KEY,

inout HPS\_LED,

output HPS\_SD\_CLK,

inout HPS\_SD\_CMD,

inout [3:0] HPS\_SD\_DATA,

output HPS\_SPIM\_CLK,

input HPS\_SPIM\_MISO,

output HPS\_SPIM\_MOSI,

inout HPS\_SPIM\_SS,

input HPS\_UART\_RX,

output HPS\_UART\_TX,

input HPS\_USB\_CLKOUT,

inout [7:0] HPS\_USB\_DATA,

input HPS\_USB\_DIR,

input HPS\_USB\_NXT,

output HPS\_USB\_STP

);

//=======================================================

// REG/WIRE declarations

//=======================================================

wire HPS\_H2F\_RST;

//=======================================================

// Structural coding

//=======================================================

hps\_fpga u0 (

.clk\_clk (CLOCK\_50), // clk.clk

.hps\_0\_h2f\_reset\_reset\_n (HPS\_H2F\_RST), // hps\_0\_h2f\_reset.reset\_n

.hps\_io\_hps\_io\_emac1\_inst\_TX\_CLK (HPS\_ENET\_GTX\_CLK), // hps\_io.hps\_io\_emac1\_inst\_TX\_CLK

.hps\_io\_hps\_io\_emac1\_inst\_TXD0 (HPS\_ENET\_TX\_DATA[0]), // .hps\_io\_emac1\_inst\_TXD0

.hps\_io\_hps\_io\_emac1\_inst\_TXD1 (HPS\_ENET\_TX\_DATA[1]), // .hps\_io\_emac1\_inst\_TXD1

.hps\_io\_hps\_io\_emac1\_inst\_TXD2 (HPS\_ENET\_TX\_DATA[2]), // .hps\_io\_emac1\_inst\_TXD2

.hps\_io\_hps\_io\_emac1\_inst\_TXD3 (HPS\_ENET\_TX\_DATA[3]), // .hps\_io\_emac1\_inst\_TXD3

.hps\_io\_hps\_io\_emac1\_inst\_RXD0 (HPS\_ENET\_RX\_DATA[0]), // .hps\_io\_emac1\_inst\_RXD0

.hps\_io\_hps\_io\_emac1\_inst\_MDIO (HPS\_ENET\_MDIO), // .hps\_io\_emac1\_inst\_MDIO

.hps\_io\_hps\_io\_emac1\_inst\_MDC (HPS\_ENET\_MDC), // .hps\_io\_emac1\_inst\_MDC

.hps\_io\_hps\_io\_emac1\_inst\_RX\_CTL (HPS\_ENET\_RX\_DV), // .hps\_io\_emac1\_inst\_RX\_CTL

.hps\_io\_hps\_io\_emac1\_inst\_TX\_CTL (HPS\_ENET\_TX\_EN), // .hps\_io\_emac1\_inst\_TX\_CTL

.hps\_io\_hps\_io\_emac1\_inst\_RX\_CLK (HPS\_ENET\_RX\_CLK), // .hps\_io\_emac1\_inst\_RX\_CLK

.hps\_io\_hps\_io\_emac1\_inst\_RXD1 (HPS\_ENET\_RX\_DATA[1]), // .hps\_io\_emac1\_inst\_RXD1

.hps\_io\_hps\_io\_emac1\_inst\_RXD2 (HPS\_ENET\_RX\_DATA[2]), // .hps\_io\_emac1\_inst\_RXD2

.hps\_io\_hps\_io\_emac1\_inst\_RXD3 (HPS\_ENET\_RX\_DATA[3]), // .hps\_io\_emac1\_inst\_RXD3

.hps\_io\_hps\_io\_sdio\_inst\_CMD (HPS\_SD\_CMD), // .hps\_io\_sdio\_inst\_CMD

.hps\_io\_hps\_io\_sdio\_inst\_D0 (HPS\_SD\_DATA[0]), // .hps\_io\_sdio\_inst\_D0

.hps\_io\_hps\_io\_sdio\_inst\_D1 (HPS\_SD\_DATA[1]), // .hps\_io\_sdio\_inst\_D1

.hps\_io\_hps\_io\_sdio\_inst\_CLK (HPS\_SD\_CLK), // .hps\_io\_sdio\_inst\_CLK

.hps\_io\_hps\_io\_sdio\_inst\_D2 (HPS\_SD\_DATA[2]), // .hps\_io\_sdio\_inst\_D2

.hps\_io\_hps\_io\_sdio\_inst\_D3 (HPS\_SD\_DATA[3]), // .hps\_io\_sdio\_inst\_D3

.hps\_io\_hps\_io\_usb1\_inst\_D0 (HPS\_USB\_DATA[0]), // .hps\_io\_usb1\_inst\_D0

.hps\_io\_hps\_io\_usb1\_inst\_D1 (HPS\_USB\_DATA[1]), // .hps\_io\_usb1\_inst\_D1

.hps\_io\_hps\_io\_usb1\_inst\_D2 (HPS\_USB\_DATA[2]), // .hps\_io\_usb1\_inst\_D2

.hps\_io\_hps\_io\_usb1\_inst\_D3 (HPS\_USB\_DATA[3]), // .hps\_io\_usb1\_inst\_D3

.hps\_io\_hps\_io\_usb1\_inst\_D4 (HPS\_USB\_DATA[4]), // .hps\_io\_usb1\_inst\_D4

.hps\_io\_hps\_io\_usb1\_inst\_D5 (HPS\_USB\_DATA[5]), // .hps\_io\_usb1\_inst\_D5

.hps\_io\_hps\_io\_usb1\_inst\_D6 (HPS\_USB\_DATA[6]), // .hps\_io\_usb1\_inst\_D6

.hps\_io\_hps\_io\_usb1\_inst\_D7 (HPS\_USB\_DATA[7]), // .hps\_io\_usb1\_inst\_D7

.hps\_io\_hps\_io\_usb1\_inst\_CLK (HPS\_USB\_CLKOUT), // .hps\_io\_usb1\_inst\_CLK

.hps\_io\_hps\_io\_usb1\_inst\_STP (HPS\_USB\_STP), // .hps\_io\_usb1\_inst\_STP

.hps\_io\_hps\_io\_usb1\_inst\_DIR (HPS\_USB\_DIR), // .hps\_io\_usb1\_inst\_DIR

.hps\_io\_hps\_io\_usb1\_inst\_NXT (HPS\_USB\_NXT), // .hps\_io\_usb1\_inst\_NXT

.hps\_io\_hps\_io\_uart0\_inst\_RX (HPS\_UART\_RX), // .hps\_io\_uart0\_inst\_RX

.hps\_io\_hps\_io\_uart0\_inst\_TX (HPS\_UART\_TX), // .hps\_io\_uart0\_inst\_TX

.led\_external\_connection\_export (LEDR), // led\_external\_connection.export

.memory\_mem\_a (HPS\_DDR3\_ADDR), // memory.mem\_a

.memory\_mem\_ba (HPS\_DDR3\_BA), // .mem\_ba

.memory\_mem\_ck (HPS\_DDR3\_CK\_P), // .mem\_ck

.memory\_mem\_ck\_n (HPS\_DDR3\_CK\_N), // .mem\_ck\_n

.memory\_mem\_cke (HPS\_DDR3\_CKE), // .mem\_cke

.memory\_mem\_cs\_n (HPS\_DDR3\_CS\_N), // .mem\_cs\_n

.memory\_mem\_ras\_n (HPS\_DDR3\_RAS\_N), // .mem\_ras\_n

.memory\_mem\_cas\_n (HPS\_DDR3\_CAS\_N), // .mem\_cas\_n

.memory\_mem\_we\_n (HPS\_DDR3\_WE\_N), // .mem\_we\_n

.memory\_mem\_reset\_n (HPS\_DDR3\_RESET\_N), // .mem\_reset\_n

.memory\_mem\_dq (HPS\_DDR3\_DQ), // .mem\_dq

.memory\_mem\_dqs (HPS\_DDR3\_DQS\_P), // .mem\_dqs

.memory\_mem\_dqs\_n (HPS\_DDR3\_DQS\_N), // .mem\_dqs\_n

.memory\_mem\_odt (HPS\_DDR3\_ODT), // .mem\_odt

.memory\_mem\_dm (HPS\_DDR3\_DM), // .mem\_dm

.memory\_oct\_rzqin (HPS\_DDR3\_RZQ), // .oct\_rzqin

.reset\_reset\_n (1'b1), // reset.reset\_n

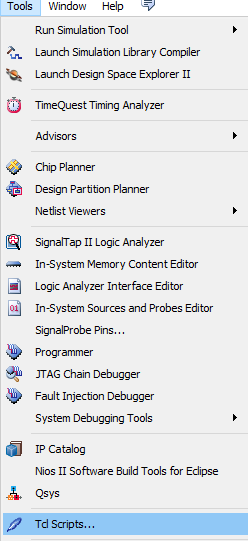
.sw\_external\_connection\_export (SW) // sw\_external\_connection.export

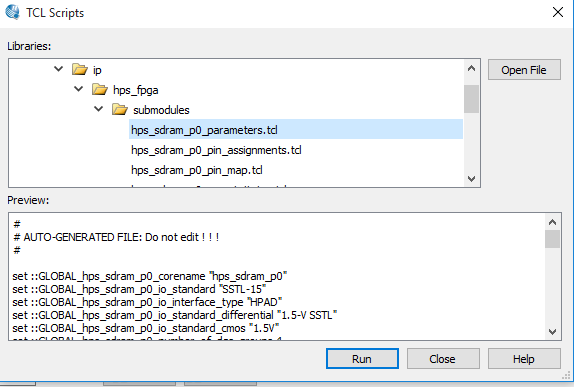
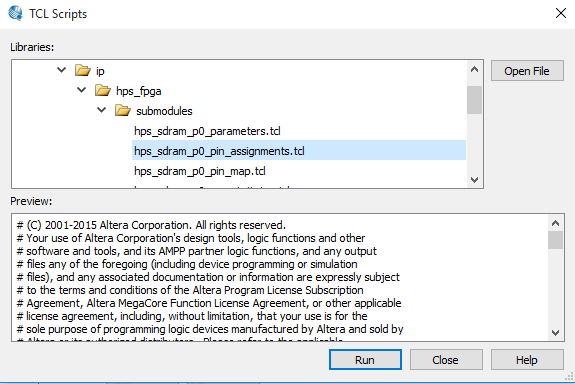
);

endmodule

1. Ahora compilamos el código para que nos genere unos archivos .tcl

Nos dirigimos a tolos 🡪Tcl scripts para agregar los Tcl, escogemos hps\_sdram\_p0\_parameters damos clic en Run y también a hps\_sdram\_p0\_pin\_assignments como se muestra en la figuras.

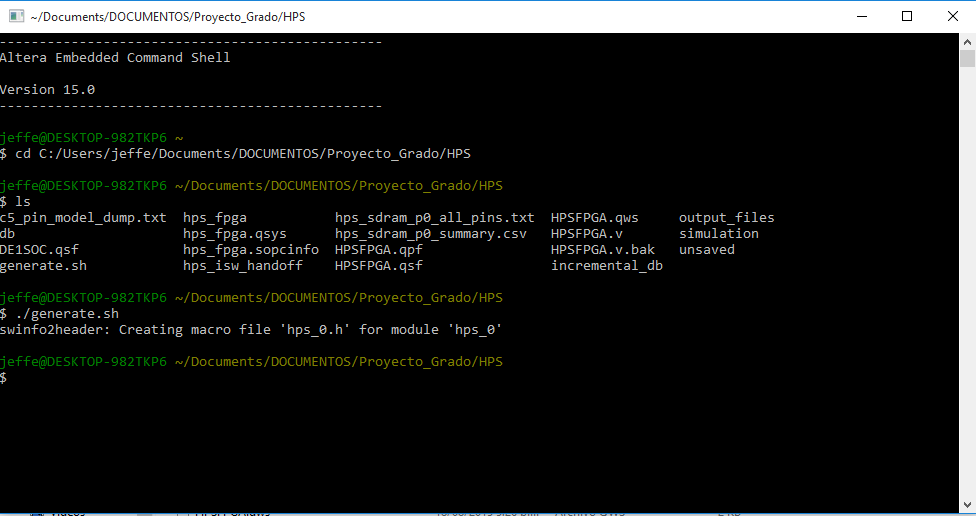




Luego compilamos sin errores.

1. Abrimos el SoC EDS Command Shell,

* Agregamos un archivo llamado generate.sh y lo pegamos en la carpeta del proyecto.
* Le damos la ruta donde se encuentra el proyecto luego se le escribe ls y por último se busca el archivo ./generate.sh



El crea un archivo hps\_0.h que se ubicara en la carpeta del proyecto.