



4 DE SEPTIEMBRE DE 2019

DISEÑO E IMPLEMENTACIÓN DE UN
SISTEMA DE GESTIÓN BASE DE DATOS
USANDO FPGA TERCASIC DE10 STANDARD
MANUAL TÉCNICO
DISEÑO DE SISTEMAS DIGITALES

JOCELYN MIRANDA Y LINO ONTANO
ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL
Facultad de Ingeniería en Electricidad y Computación

OBJETIVOS

I. General

Realizar consultas desde la FPGA hacia el servidor de base de datos implementado en la parte del Hard Processor System de la tarjeta de desarrollo DE10 Standard.

II. Específicos

- Diseñar la arquitectura digital del sistema.
- Realizar la conexión entre las componentes a utilizar en el archivo Qsys.
- Configurar el servidor de base de datos en el HPS.
- Establecer el direccionamiento en el sistema operativo Linux.
- Crear archivos ejecutables para realizar las consultas a la base de datos.

DESCRIPCIÓN DEL PROYECTO

El sistema constará de diseñar e implementar un sistema que realice monitoreo de consultas a una base de datos. El servidor de la base de datos se la implementará en el Hard-Processor System (HPS) de la FPGA DE-10 Standard, la cuál será manejada vía Putty SSH conectada a una red local. La FPGA será la encargada de realizar las consultas al HPS el cual responderá y la FPGA mostrará el resultado por pantalla por medio de su puerto VGA. El proyecto se lo desarrollará en tres etapas: La primera etapa constará de montar el servidor de la base de datos en la HPS y poder acceder a ella vía Putty SSH, la imagen del sistema operativo será descargada desde la página de Terasic 1 y donde se levantará el servicio de base de datos MySQL. La segunda etapa será poder mostrar al servidor por pantalla utilizando la FPGA, donde el procesador NIOS manejará las respuestas del servidor para ser mostradas y procesadas. Y la última etapa será de poder, desde la FPGA, solicitar consultas al HPS y recibir las mismas respuestas, es decir una comunicación bidireccional.

DISEÑO DE LA ARQUITECTURA

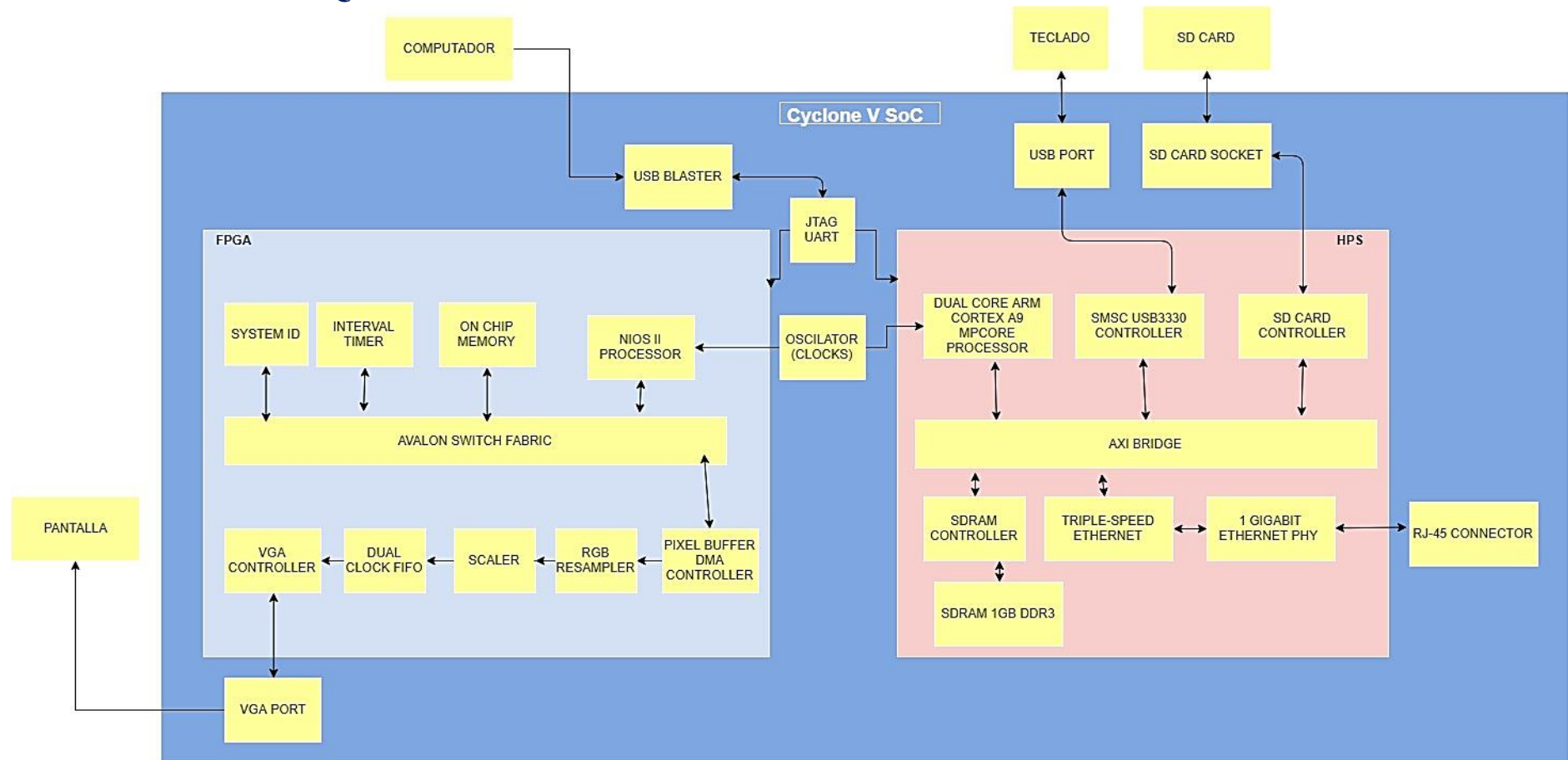


Ilustración 1 - Arquitectura

I. Descripción de cada bloque

AVALON: Permite la comunicación entre los diferentes bloques presentes en el sistema.

SDRAM CONTROLLER: Permite al procesador almacenar datos para su manipulación.

SDRAM 1GB DDR3: Dada la cantidad de peticiones que el servidor tendrá, se consideró este bloque ya que ejecuta instrucciones con mayor rapidez, esto permite que mejore la tasa de transferencia de datos.

SD CARD CONTROLLER: Recibe las señales dadas por el convertidor analógico/digital las cuales son procesadas en el interior del Core. El Core genera la resolución para la pantalla. Guardará los datos que reciba de los sensores en una parte de la memoria.

SD CARD SOCKET: Puerto para el intercambio del flujo de información relacionado con el servidor Linux.

SD CARD: Tarjeta Micro SD booteable con el sistema Linux.

TRIPLE- SPEED ETHERNET: Contiene múltiples velocidades semejantes a PHY.

1GB ETHERNET PHY: Interfaz Ethernet por la cual se conectará el dispositivo a un enrutador para su manejo por PUTTY.

RJ-45 CONNECTOR: Para la conexión a la red.

DUAL CORE ARM CORTEX A9 MPCORE PROCESSOR: Se encargará de la parte del servidor de base de datos.

JTAG UART: Permite la programación del procesador y la creación del sistema embebido en la tarjeta de desarrollo.

USB BLASTER: Puerto USB de la tarjeta FPGA.

VGA CONTROLLER: Sincroniza la salida de las tramas para correcta reproducción a través de una pantalla conectada por VGA.

VGA PORT: Para la conexión de la pantalla con la tarjeta de video

NIOS II PROCESSOR: Se encarga de preparar las tramas para la salida por medio del protocolo VGA.

ON CHIP MEMORY: Va de la mano con el Procesador NIOS II. El núcleo de memoria FIFO en el chip es un componente configurable utilizado para almacenar datos y proporcionar control de flujo en un sistema SOPC Builder. El FIFO puede operar con un solo reloj o con relojes separados para los puertos de entrada y salida. El FIFO en chip. El núcleo de memoria no es compatible con ráfaga de lectura o escritura.

INTERNAL TIMER: Controla los intervalos de tiempo de todo el sistema.

SYSTEM ID: Se encarga de verificar que un programa ejecutable sea compilado con la imagen del Hardware actual configurado en la tarjeta de desarrollo FPGA.

DESARROLLO

Instalación del Sistema operativo Linux en una MicroSD

1. Descargar la imagen Linux LXDE Desktop de la tarjeta DE10 Standard desde la página de Terasic¹.

Linux BSP (Board Support Package): MicroSD Card Image





Title	Version	Size(KB)	Date Added	Download
Linux Console (Kernel 4.5)	1.2 		2018-03-15	
Linux LXDE Desktop (Kernel 4.5)	1.2 		2017-06-06	

Ilustración 2 - Terasic - Linux

2. Descargue e instale el software *Win32 Disk Imager*.
3. Conecte la *MICROSD* al ordenador.
4. Abra el software *Win32 Disk Imager*. Seleccione el archivo de la imagen descargada y el dispositivo donde deseamos escribir. Clic en *Write*.

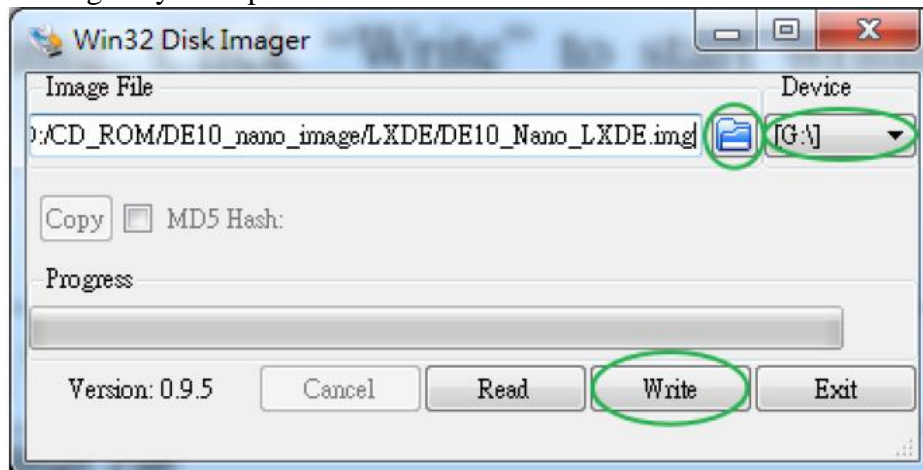


Ilustración 3 - Grabar la imagen en una MICROSD

¹ <https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&No=1081&PartNo=4>

Configuración del Servidor de base de datos en Linux

1. Conectamos la MicroSD y el cable Ethernet a la tarjeta de desarrollo DE10 Standard.
2. Después de arrancar el sistema operativo, utilizar el usuario por defecto para iniciar sesión en el mismo. *User: root, sin contraseña.*
3. Para configurar la red y poder tener acceso a Internet. Modificamos el archivo de interfaces de red por medio del siguiente comando:
④ *vi /etc/network/interfaces*
4. Copie el siguiente código en el archivo modificando la IP, máscara de subred y Gateway.

```
#include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
auto lo
iface lo inet loopback

#auto eth0
wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

iface eth0 inet static
address 200.126.14.134
netmask 255.255.255.128
gateway 200.126.14.129

#auto wlan0
#iface wlan0 inet dhcp
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Ilustración 4 - Archivo interfaces

5. Para tener acceso a Internet se configuran las DNS. Utilice el siguiente comando:
④ *vi /etc/resolv.conf*
6. Escribir lo siguiente en el archivo resolv.conf, guarde y cierre el editor de texto vi presionando ESC wq!.

```
nameserver 8.8.8.8
nameserver 192.168.1.17
nameserver 192.168.1.19
root@DE10-Standard:/etc#
```

Ilustración 5 - resolv.conf

7. Ingresar las siguientes líneas de comando para crear una ruta por defecto.
④ *ifconfig <interfaz> <dirección IP> netmask <Máscara de subred>*
route add default gw <Gateway>
8. Reiniciamos el servicio de networking para hacer efectivos los cambios realizados.
④ *systemctl restart networking.service*
9. Realizamos un ping para probar conectividad a Internet
④ *ping www.espol.edu.ec*

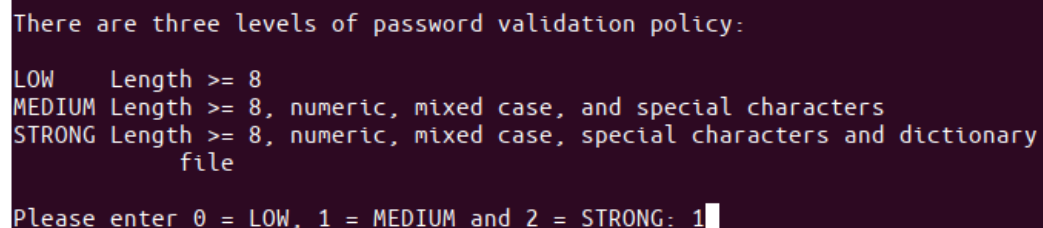
10. Para configurar el servidor de base de datos MySQL instalamos los paquetes necesarios con el siguiente comando:

⌘ `sudo apt-get install mysql-server mysql-client libmysqlclient-dev`

11. Cada vez que instala una copia nueva de MySQL, hay algunas configuraciones predeterminadas que debe cambiar para mejorar la seguridad de su instalación de MySQL. Esto incluye la eliminación de usuarios de prueba, bases de datos de prueba y permisos para inicio de sesión remoto por parte de un usuario root. Ejecutar el siguiente comando:

⌘ `sudo mysql_secure_installation`

12. Mientras ejecuta este comando, lo primero que se le pedirá que haga es configurar el complemento Validar contraseña. Esto le permite establecer una contraseña segura para root en función de la seguridad de la contraseña que desea elegir. Ingrese *Y* para ejecutar *Validate Password Plugin* y recibirá el siguiente mensaje:



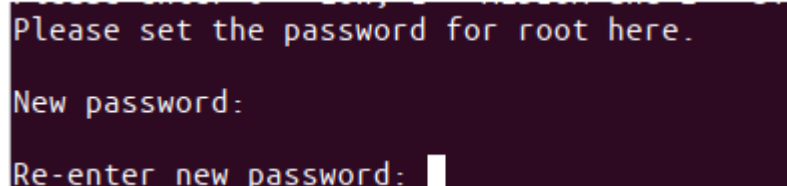
```
There are three levels of password validation policy:

LOW      Length >= 8
MEDIUM  Length >= 8, numeric, mixed case, and special characters
STRONG Length >= 8, numeric, mixed case, special characters and dictionary
         file

Please enter 0 = LOW, 1 = MEDIUM and 2 = STRONG: 1
```

Ilustración 6 - Configurar complemento para validar la contraseña

13. El sistema le pedirá la nueva contraseña de *root*. Ingrese y vuelva a ingresar la contraseña en las siguientes indicaciones.



```
Please set the password for root here.

New password:

Re-enter new password:
```

Ilustración 7 - Ingrese contraseña

14. Revisar si el servicio de MySQL está funcionando correctamente.

⌘ `systemctl status mysql.service`

15. Una vez completa la instalación ingresar a MySQL con su usuario y contraseña creados.

⌘ `mysql -u root -p`

16. Crear la base de datos SISTEMA_RIEGO

⌘ `create Database SISTEMA_RIEGO;`

17. Ejecute el siguiente comando para realizar modificaciones en la base de datos creada.

⌘ `use SISTEMA_RIEGO;`

18. Crear la tabla microcontrolador

```
CREATE TABLE microcontrolador(id INT NOT NULL AUTO_INCREMENT,nombre VARCHAR(50),bateria float,firmware VARCHAR(50),num_sensores int,description varchar(250),dtm VARCHAR(30),fabricante VARCHAR(30),num_procesadores int,num_periféricos int,PRIMARY KEY (id)) ENGINE=InnoDB;
```

19. Crear la tabla ultimoRegistro

```
CREATE TABLE ultimoRegistro (id INT NOT NULL AUTO_INCREMENT, dispositivo VARCHAR(50),valor int, PRIMARY KEY (id)) ENGINE=InnoDB;
```

20. Agregar registros a las tablas creadas por medio del siguiente comando:

```
INSERT INTO microcontrolador VALUES (1,'AVRMini',26.8,'v1.2',3, 'Desarrollado por Pascal Stang','2010-11-25 11:11:11','Stang',2,5);
```

```
INSERT INTO microcontrolador VALUES (2,'Wiring',46.8,'v2.2',4, 'Proyecto abierto iniciado por Hernando Barragán','2011-11-25 19:11:11','Instituto Avric',3,4);
```

```
INSERT INTO microcontrolador VALUES (3,'Arduino',96.8,'v3.2',3, 'Plataforma de código abierto','2012-11-25 18:11:11','Arduino',1,6);
```

```
INSERT INTO microcontrolador VALUES (4,'Basic Stamp',26.8,'v1.0',5, 'Ejecuta programas en lenguaje PBASIC','2013-11-25 10:11:11','Stang',3,5);
```

```
INSERT INTO microcontrolador VALUES (5,'Raspberry Pi 3',16.8,'v3.0',6, 'Capaz de correr distribución de linux','2014-11-25 11:11:11','Stang',3,8);
```

```
INSERT INTO microcontrolador VALUES (6,'Raspberry Pi 4',88.7,'v5.0',2, 'Capaz de correr distribución de linux','2015-11-25 12:11:11','Stang',4,5);
```

```
INSERT INTO microcontrolador VALUES (7,'Beaglebone',11.9,'v1.0',3, 'Capaz de soportar ambientes no favorables','2016-11-25 13:11:11','Stang',5,4);
```

```
INSERT INTO microcontrolador VALUES (8,'FPGA de10-standard',1.1,'v1.0',3, 'Desarrollado por Intel','2017-11-25 14:11:11','Intel',6,4);
```

```
INSERT INTO microcontrolador VALUES (9,'FPGA de0-nano',97.6,'v2.0',4, 'Desarrollado por Intel','2018-11-25 15:11:11','Intel',2,5);
```

```
INSERT INTO microcontrolador VALUES (10,'FPGA de10-nano',12.3,'v3.0',5, 'Desarrollado por Intel','2019-11-25 16:00:00','Intel',3,3);
```

```
INSERT INTO microcontrolador VALUES (11,'FPGA de2-115',78.9,'v4.0',3, 'Desarrollado por Intel','2020-11-25 17:00:00','Intel',2,2);
```

```
INSERT INTO ultimoRegistro VALUES (1,'sensor1',23);
```

```
INSERT INTO ultimoRegistro VALUES (2,'sensor2',48);
```

```
INSERT INTO ultimoRegistro VALUES (3,'sensor3',26);
```


21. Salir de Mysql

⌕ `exit;`

22. Crear una carpeta llamada “HPS” en el directorio `/home/root`

⌕ `mkdir <carpeta>;`

23. Crear un archivo llamado main.c

⌕ `vi <archivo>;`

24. Copie el siguiente código en su archivo main.c

```
#include <stdlib.h>
#include <stdio.h>
#include <mysql.h>
#include <string.h>
int main() {
    MYSQL *conn;
    MYSQL_RES *res;
    MYSQL_ROW row;

    char *server = "localhost";
    char *user = "root";
    char *password = "root123"; /* Poner la contraseña */
    char *database = "SISTEMA_RIEGO";
    conn = mysql_init(NULL);
    char *respuesta;
    int sensor1;
    int sensor2;
    int sensor3;
    int i=0;

    /* Conectarse a la base de datos */
    if (!mysql_real_connect(conn, server, user, password, database, 0, NULL, 0)) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1);
    }
    /* enviar la consulta SQL */
    if (mysql_query(conn, "select * from ultimoRegistro")) {
        fprintf(stderr, "%s\n", mysql_error(conn));
        exit(1);
    }
    res = mysql_use_result(conn);

    // printf("Ultimo valor sensor1: ");

    while ((row = mysql_fetch_row(res)) != NULL) {
        respuesta = row[2];
```

```

        //sensor1= atoi(respuesta);
        if(i==0){ sensor1=atoi(respuesta); }
        if(i==1){ sensor2=atoi(respuesta); }
        if(i==2){ sensor3=atoi(respuesta); }
        i+=1;
    }
    printf("%d \n", sensor1);
    //printf("Ultimo valor sensor2: ");
    printf("%d \n", sensor2);
    //printf("Ultimo valor sensor3: ");
    printf("%d \n", sensor3);

    /* liberar los recursos y cerrar la conexion */
    mysql_free_result(res);
    mysql_close(conn);
}

```

25. Crear el archivo ejecutable llamado *main20* del programa main.c

```
⌘ gcc main.c $(mysql_config --libs) -o <name> $(mysql_config --oflags)
```

26. Asignarle todos los permisos al archivo generado *main20*

```
⌘ chmod +x main20
```

27. Crear un archivo Shell que llamará al archivo ejecutable y generará un archivo de texto con el resultado de la consulta, este posteriormente será leído.

```
⌘ vi mostrar.sh
```

28. Copie el siguiente código en el archivo mostrar.sh

```
#!/bin/bash
./main20 > consulta.txt
```

Configurar Arquitectura

1. Utilizamos el proyecto por defecto que tiene configurado la tarjeta DE10 Standard “ControlPanel”.
2. Abrir el archivo DE10_Standard_FB.qpf alojado en la carpeta SocFPGA/ControlPanel/Quartus de los archivos del CD.

c5_pin_model_dump.txt	03/09/2019 13H41	Documento de tex...	5 KB
CLOCK_DIV_50.vhd	08/06/2018 10H01	Virtual Hard Disk	4 KB
contador_hw.tcl	02/09/2019 15H46	Altium Script Doc...	5 KB
DE10_Standard_FB.bsf	26/04/2017 20H22	Archivo BSF	25 KB
DE10_Standard_FB.qpf	02/02/2017 09H42	Archivo QPF	1 KB
DE10_Standard_FB.qsf	03/09/2019 13H35	Archivo QSF	73 KB

Ilustración 8 - Archivo qpf Control Panel

3. Creamos un archivo VHDL para el contador de segundos llamado *bloque*, File -> New -> VHDL File.

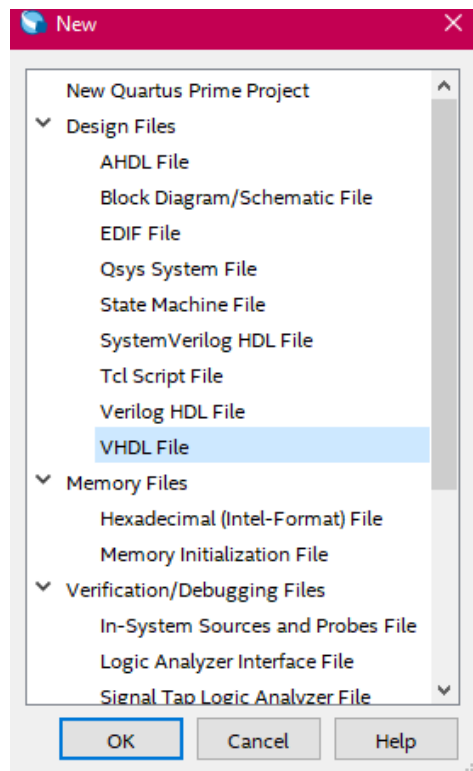


Ilustración 9 - Crear archivo VHDL

4. Copiamos el siguiente código en el mismo.

```
#####  
-- Contador.vhd  
-- This component describes a simple counter UP-DOWN with an Avalon-MM slave  
-- interface. The Input Signals can be written to registers 0 and 1, and the outputs of the  
-- counter can be read back from registers 2 and 3.  
#####  
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.std_logic_UNSIGNED.all;  
use ieee.std_logic_Arith.all;
```

ENTITY bloque IS

PORT(

```
-- Avalon Clock interface
clk1,clk2 : in std_logic; --clk1 for Avalon and clk2 for Counter
-- Avalon Reset interface
reset : in std_logic;
-- Avalon-MM Slave interface
address : in std_logic_vector(1 downto 0);
read : in std_logic;
write : in std_logic;
readdata : out std_logic_vector(31 downto 0);
writedata : in std_logic_vector(31 downto 0);
counter_conduit: out std_logic_vector(9 downto 0)
);
```

END bloque;

ARCHITECTURE sol OF bloque IS

```
--Address for different registers
constant REG_INPUT_1_OFST : std_logic_vector(1 downto 0) := "00";--Address
for Cargar, habilcnt and descendente
constant REG_INPUT_2_OFST : std_logic_vector(1 downto 0) := "01";--Address
for dato_ent
constant REG_OUTPUT_1_OFST : std_logic_vector(1 downto 0) := "10";--
Address for Q
constant REG_OUTPUT_2_OFST : std_logic_vector(1 downto 0) := "11";--
Address for ct_term
signal reg_input_1 : unsigned(writedata'range); --creating register 1 for Inputs ->
Cargar, habilcnt and descendente
signal reg_input_2 : unsigned(writedata'range); --creating register 2 for input ->
dato_ent
signal reg_output_1 : std_logic_vector(31 downto 0); --creating register 3 for
Output -> Q
signal reg_output_2 : unsigned(readdata'range); --creating register 4 for Output ->
ct_term
SIGNAL cargar, habilcnt, descendente, ct_term : STD_LOGIC; -- Input Signals
for the Counter Process
SIGNAL dato_ent,Q : STD_LOGIC_VECTOR (9 downto 0); -- Input and Output
Signals for the Counter Process
SIGNAL conteo: STD_LOGIC_VECTOR(9 downto 0); -- define a 4 bits Bus
```

BEGIN

```
-- Avalon-MM slave write
process(clk1, reset)
begin
    if reset = '1' then
        reg_input_1 <= (others => '0');
        reg_input_2 <= (others => '0');
        elsif rising_edge(clk1) then
            if write = '1' then
```

```

                                case address is
                                    when REG_INPUT_1_OFST =>
                                        reg_input_1 <= unsigned(writedata);
                                    when REG_INPUT_2_OFST =>
                                        reg_input_2 <= unsigned(writedata);
                                    -- RESULT register is read-only
                                    when REG_OUTPUT_1_OFST
=> null;
                                        when REG_OUTPUT_2_OFST
=> null;
                                    -- Remaining addresses in register
map are unused.
                                        when others => null;
                                end case;
                            end if;
                        end if;
                    end process;

-- Avalon-MM slave read
process(clk1, reset)
begin
    reg_output_1(9 downto 0)<=Q;reg_output_2(0)<=ct_term;
    if rising_edge(clk1) then
        if read = '1' then
            case address is
                when REG_INPUT_1_OFST =>
                    readdata <= std_logic_vector(reg_input_1); --assign
readdata<=Cargar&habilcnt&descendente
                when REG_INPUT_2_OFST =>
                    readdata <= std_logic_vector(reg_input_2); --assign readdata<=data_ent
                when REG_OUTPUT_1_OFST =>
                    readdata <= reg_output_1; --assign readdata<=Q
                when REG_OUTPUT_2_OFST =>
                    readdata <= std_logic_vector(reg_output_2);--assign readdata<=ct_term

                -- Remaining addresses in register map are
unmapped => return 0.
                when others => readdata <= (others =>
'0');
            end case;
        end if;
    end if;
end process;

--Counter Process
PROCESS(clk2,reset,descendente)
BEGIN

    cargar<=reg_input_1(2);habilcnt<=reg_input_1(1);descendente<=reg_input_1(2);
    dato_ent<=conv_std_logic_vector(reg_input_2,10);
    if reset='1' then conteo<="0000000000"; -- borrar asÃ-ncrona

```

```

        elsif (clk2'event and clk2='1') then -- flanco ascendente?
            if cargar='1' then conteo<=dato_ent; --carga en paralelo
            elsif habilcnt='1' then -- habilitado?
                if descendente='0' then conteo<=conteo+1; --
incremento
                else conteo<=conteo-1; --decremento
                end if;
            end if;
        end if;
        if (((conteo="0000000000" and descendente='1')) OR
((conteo="1111111111" and descendente='0'))) AND habilcnt='1'
            then ct_term<='1';
            else ct_term<='0';
            end if;
        q<=conteo; --transfer the content from register to output
    END PROCESS;

    counter_conduit<=Q; --transfer the content from register Q to output
Conduit

END sol;

```

4. Creamos un archivo VHDL para el divisor de frecuencia llamado *CLOCK_50*, *File* -> *New* -> *VHDL File* (Ilustración 9).

5. Copiamos el siguiente código.

```

LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.all;
USE IEEE.STD_LOGIC_ARITH.all;
USE IEEE.STD_LOGIC_UNSIGNED.all;

ENTITY CLOCK_DIV_50 IS
    PORT
    ( CLOCK_50MHz :IN STD_LOGIC;
      CLOCK_1MHz :OUT STD_LOGIC;
      CLOCK_100KHz :OUT STD_LOGIC;
      CLOCK_10KHz :OUT STD_LOGIC;
      CLOCK_1KHz :OUT STD_LOGIC;
      CLOCK_100Hz :OUT STD_LOGIC;
      CLOCK_10Hz :OUT STD_LOGIC;
      CLOCK_1Hz :OUT STD_LOGIC);
END CLOCK_DIV_50;

ARCHITECTURE a OF CLOCK_DIV_50 IS
    SIGNAL count_1Mhz: STD_LOGIC_VECTOR(5 DOWNT0 0);
    SIGNAL count_100Khz, count_10Khz, count_1Khz: STD_LOGIC_VECTOR(2
DOWNT0 0);
    SIGNAL count_100hz, count_10hz, count_1hz: STD_LOGIC_VECTOR(2 DOWNT0
0);
    SIGNAL clock_1Mhz_int, clock_100Khz_int, clock_10Khz_int, clock_1Khz_int:
STD_LOGIC;
    SIGNAL clock_100hz_int, clock_10hz_int, clock_1hz_int: STD_LOGIC;

```

```

BEGIN
  PROCESS
  BEGIN
    -- Divide by 50
    WAIT UNTIL clock_50Mhz'EVENT and clock_50Mhz = '1'; -- 24 Mhz
    IF count_1Mhz < 49 THEN
      count_1Mhz <= count_1Mhz + 1;
    ELSE
      count_1Mhz <= "000000";
    END IF;
    IF count_1Mhz < 4 THEN
      clock_1Mhz_int <= '0';
    ELSE
      clock_1Mhz_int <= '1';
    END IF;
    -- Ripple clocks are used in this code to save prescalar hardware
    -- Sync all clock prescalar outputs back to master clock signal
    clock_1Mhz <= clock_1Mhz_int;
    clock_100Khz <= clock_100Khz_int;
    clock_10Khz <= clock_10Khz_int;
    clock_1Khz <= clock_1Khz_int;
    clock_100hz <= clock_100hz_int;
    clock_10hz <= clock_10hz_int;
    clock_1hz <= clock_1hz_int;
  END PROCESS;
  -- Divide by 10
  PROCESS
  BEGIN
    WAIT UNTIL clock_1Mhz_int'EVENT and clock_1Mhz_int = '1';
    IF count_100Khz /= 4 THEN
      count_100Khz <= count_100Khz + 1;
    ELSE
      count_100Khz <= "000";
      clock_100Khz_int <= NOT clock_100Khz_int;
    END IF;
  END PROCESS;
  -- Divide by 10
  PROCESS
  BEGIN
    WAIT UNTIL clock_100Khz_int'EVENT and clock_100Khz_int = '1';
    IF count_10Khz /= 4 THEN
      count_10Khz <= count_10Khz + 1;
    ELSE
      count_10Khz <= "000";
      clock_10Khz_int <= NOT clock_10Khz_int;
    END IF;
  END PROCESS;
  -- Divide by 10
  PROCESS
  BEGIN

```

```

    WAIT UNTIL clock_10Khz_int'EVENT and clock_10Khz_int = '1';
    IF count_1Khz /= 4 THEN
        count_1Khz <= count_1Khz + 1;
    ELSE
        count_1Khz <= "000";
        clock_1Khz_int <= NOT clock_1Khz_int;
    END IF;
END PROCESS;
-- Divide by 10
PROCESS
BEGIN
    WAIT UNTIL clock_1Khz_int'EVENT and clock_1Khz_int = '1';
    IF count_100hz /= 4 THEN
        count_100hz <= count_100hz + 1;
    ELSE
        count_100hz <= "000";
        clock_100hz_int <= NOT clock_100hz_int;
    END IF;
END PROCESS;
-- Divide by 10
PROCESS
BEGIN
    WAIT UNTIL clock_100hz_int'EVENT and clock_100hz_int = '1';
    IF count_10hz /= 4 THEN
        count_10hz <= count_10hz + 1;
    ELSE
        count_10hz <= "000";
        clock_10hz_int <= NOT clock_10hz_int;
    END IF;
END PROCESS;
-- Divide by 10
PROCESS
BEGIN
    WAIT UNTIL clock_10hz_int'EVENT and clock_10hz_int = '1';
    IF count_1hz /= 4 THEN
        count_1hz <= count_1hz + 1;
    ELSE
        count_1hz <= "000";
        clock_1hz_int <= NOT clock_1hz_int;
    END IF;
END PROCESS;
END a;

```

6. Damos clic derecho en el archivo VHDL del contador y seleccionamos la opción *Set as Top-Level Entity*, Ilustración 10.

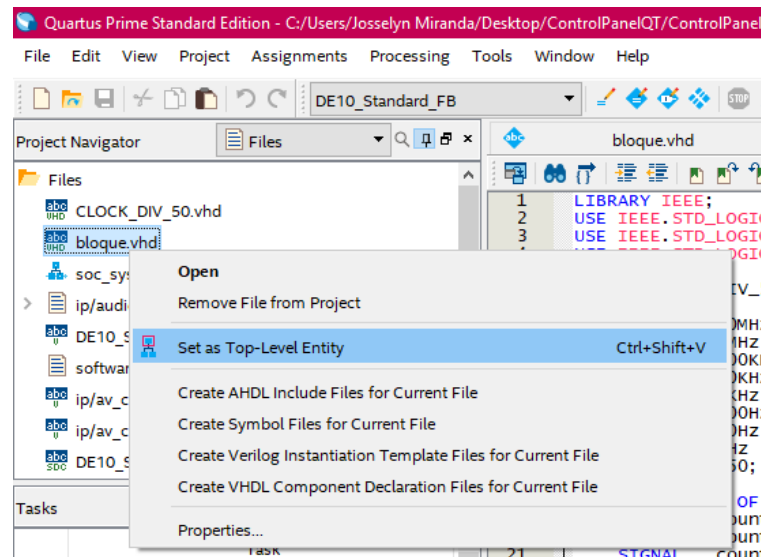



Ilustración 10 - Asignar alta prioridad

7. Damos clic en  para empezar la compilación del archivo.
8. Repetimos el paso 6 y 7 pero esta vez para el archivo VHDL del divisor de frecuencia.
9. Luego de la compilación exitosa (100%). Seleccionamos *Files* en el *Project Navigator*. Damos doble clic en el archivo *soc_system.qsys* donde realizaremos la conexión de los bloques de nuestra arquitectura.

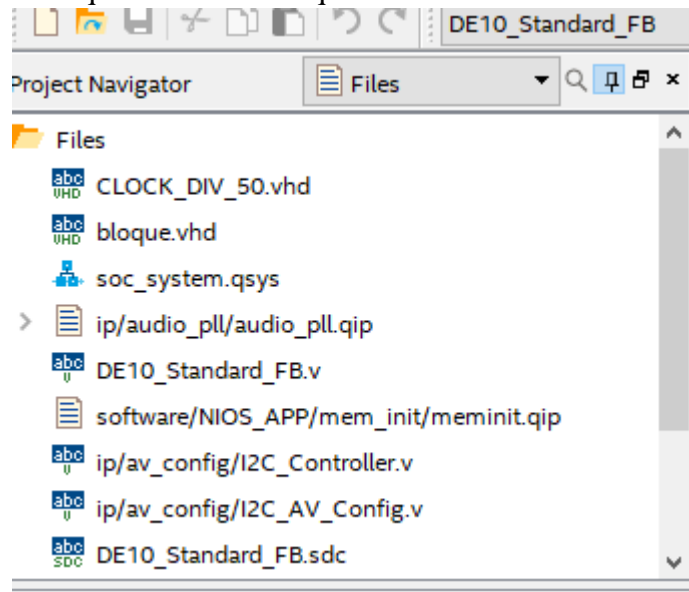


Ilustración 11 - Project navigator

10. Dentro de la ventana Qsys observamos todos los componentes utilizados por el proyecto base Control Panel. Sin embargo, nuestra arquitectura no utiliza todo ello, por lo que procedemos a eliminar los siguientes componentes: vga_controller, Terasic_ALSA, tv_decoder, vga_stream y el resto de componentes relacionados a VGA.
11. Creamos las componentes del *CONTADOR* y *CLOCK50*. Doble clic en *New Component* del catálogo IP.

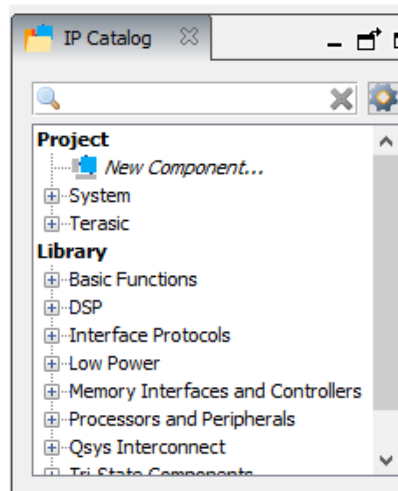


Ilustración 12 - Crear componentes

12. Seleccionamos *Add File* y escogemos el archivo del bloque *contador.vhd*. Luego haga clic en *Analyze Synthesis Files*. Clic en *Finish*.

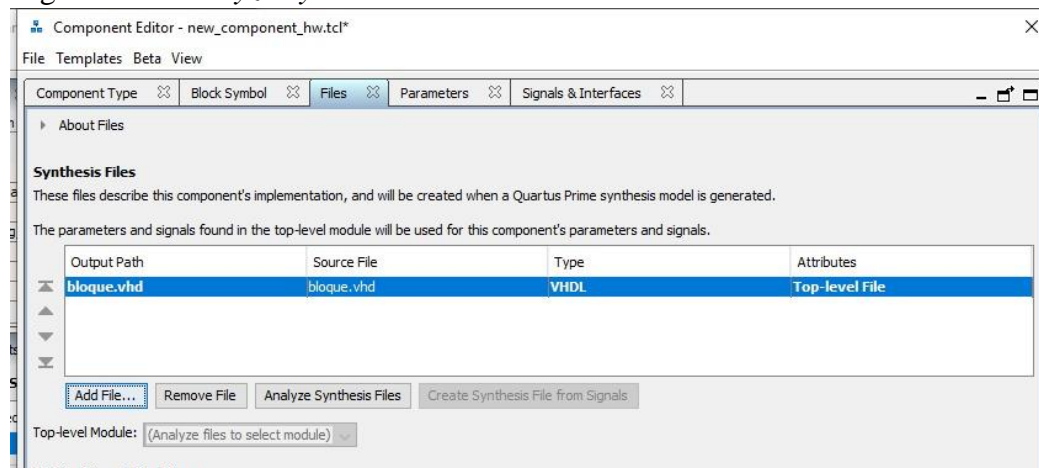
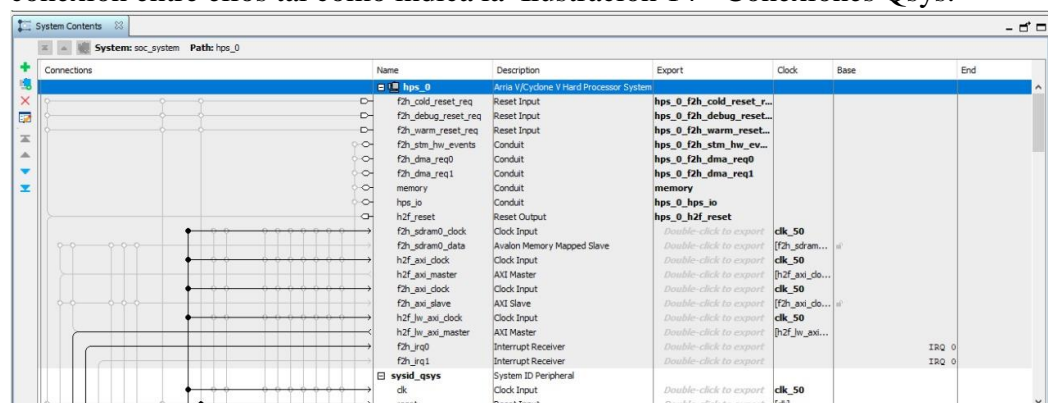


Ilustración 13 - Add file vhd

13. Repita los pasos anteriores para crear el componente *CLOCK_50* usando el archivo VHDL respectivo.
14. Agregamos los bloques para VGA, para agregarlos utilizamos el buscador de IP Catalog.
15. Una vez agregados todos los bloques faltantes a Qsys empezar a realizar la conexión entre ellos tal como indica la Ilustración 14 - Conexiones Qsys.



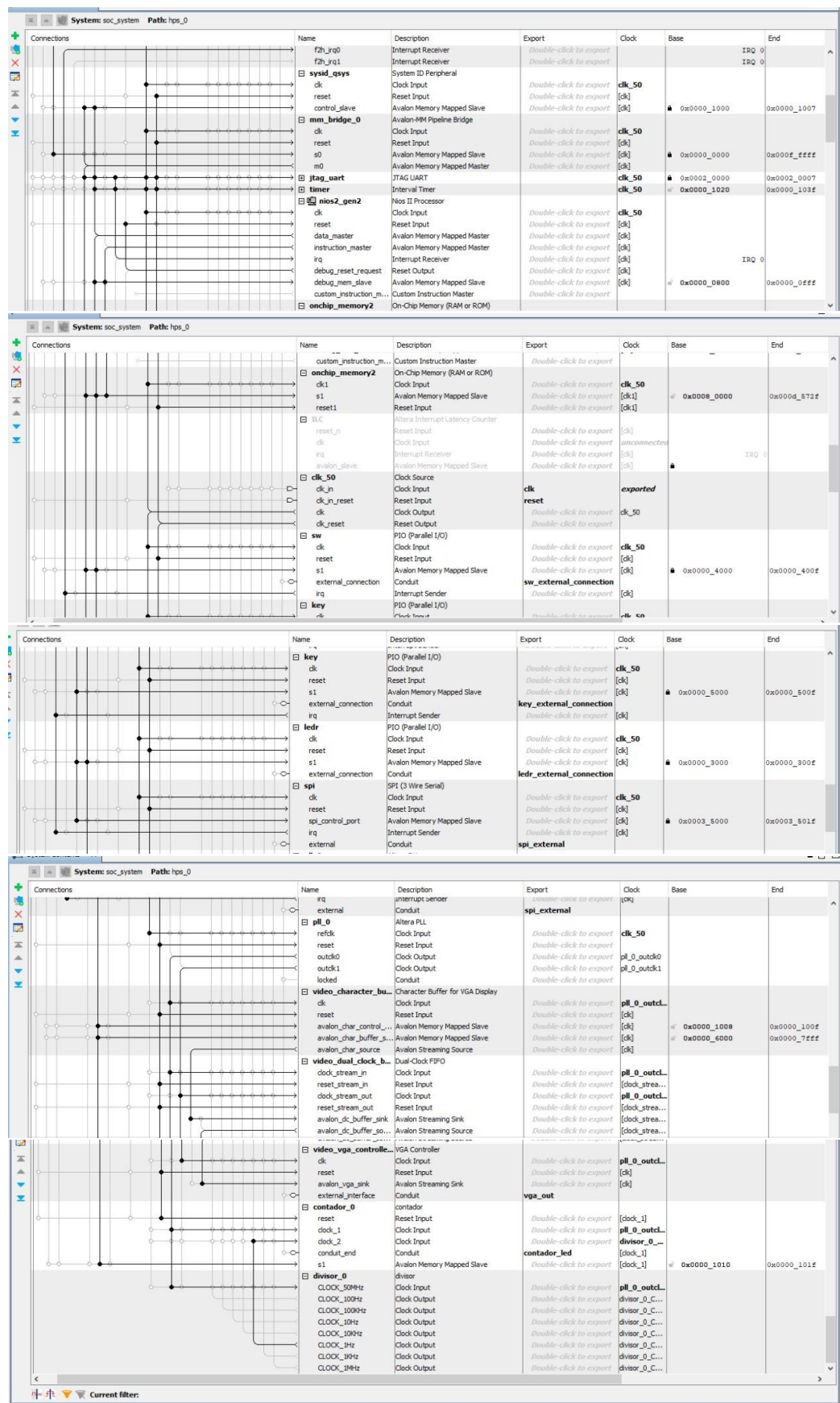


Ilustración 14 - Conexiones Qsys

16. Configuramos el Vector Reset y Exception en el *nios2_gen2*.

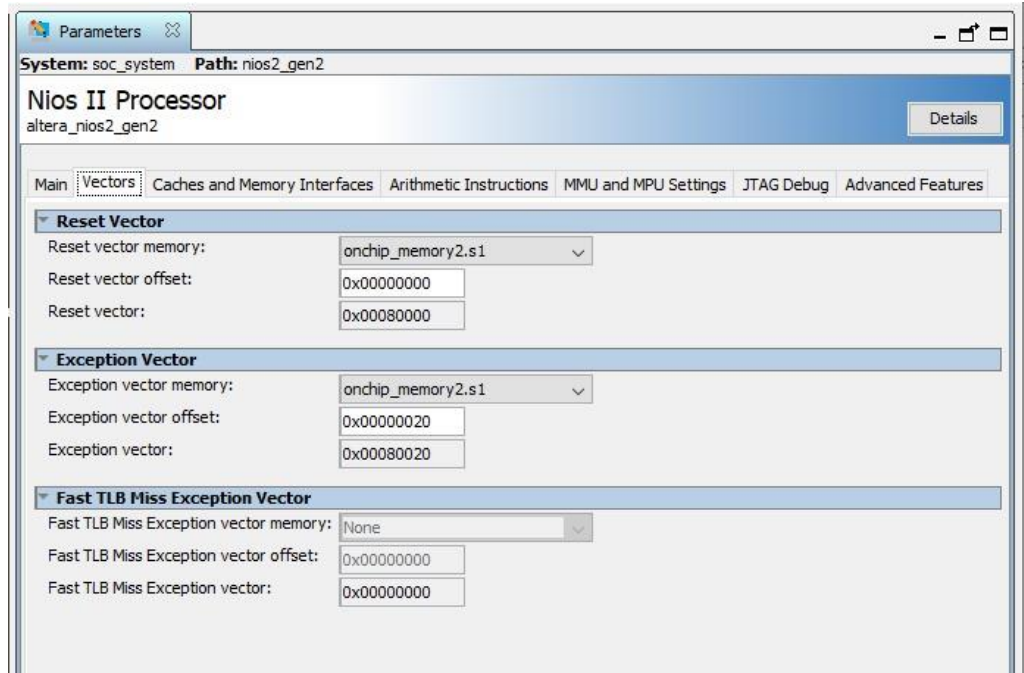


Ilustración 15 - Reset and Exception Vectors

17. Después de realizar las conexiones entre componentes asignamos Base Address desde la pestaña System ->Assign Base Address. Como indica la Ilustración 16.

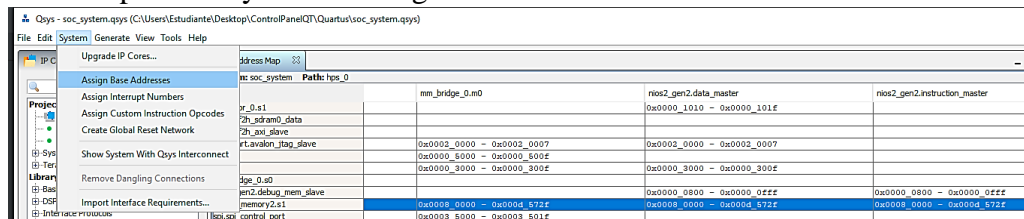


Ilustración 16 - Asignar direcciones

18. Presionar el botón *Generar HDL* para actualizar el *soc_system.qsys* con los cambios realizados.

19. Ver la instanciación del componente de *soc_system u0*. Seleccionar la opción *Generate -> Show Instantiation Template*.

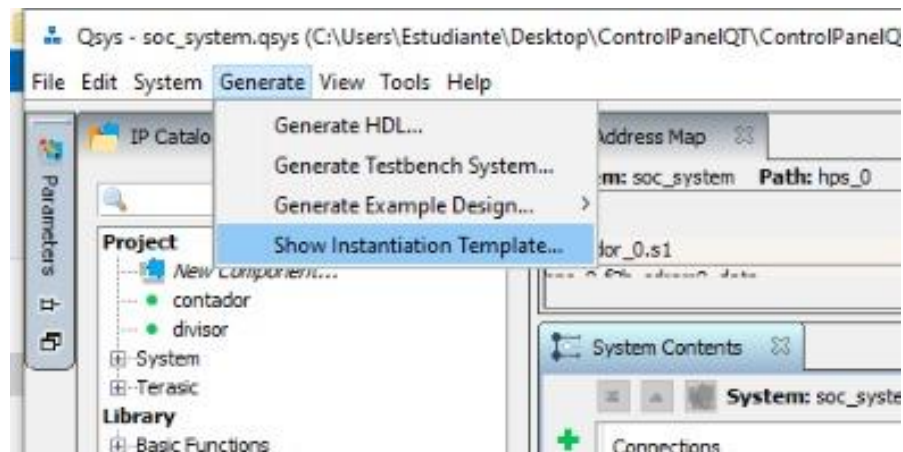


Ilustración 17 - Instanciación

20. Clic en *Copy* para copiar el texto generado.

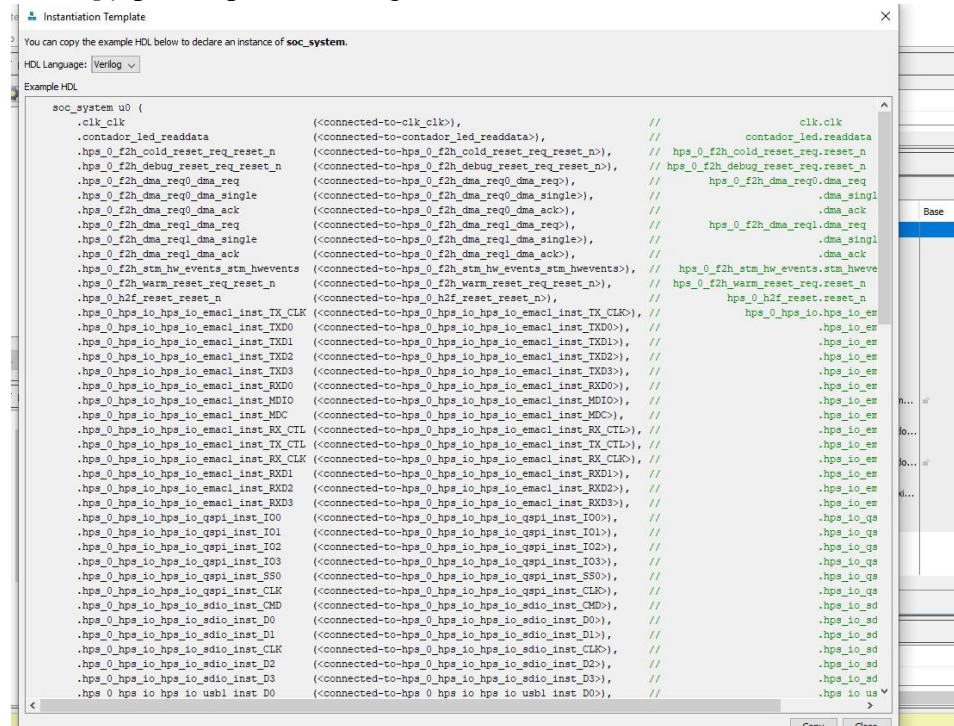


Ilustración 18 - Instantiation Template

21. En el archivo *DE10_Standard_FB.v* reemplace la componente *soc_system u0* por el texto generado en el paso anterior y complete las entradas y salidas que recibe dicho componente. Debe quedar lo más semejante al código adjunto en el enlace escrito al pie de esta página².

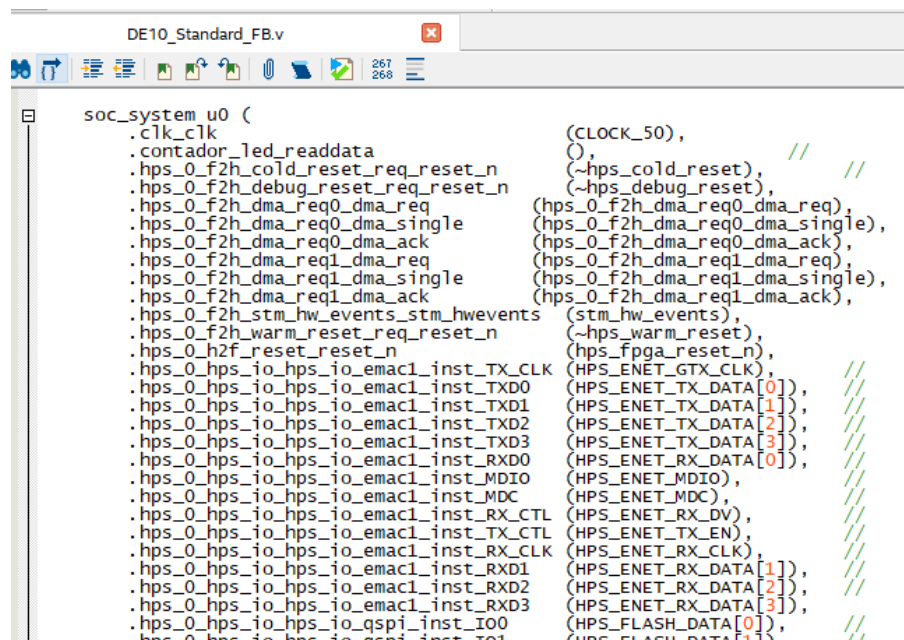


Ilustración 19 - Entradas y salidas de la componente *soc_system u0*

² https://drive.google.com/file/d/1AbAcdrImQLI_LZUMD9LCBaUVxVFCaM0z/view?usp=sharing

22. Compilación del proyecto. Repita el paso 6 y 7 para el archivo *DE10_Standard_FB.v*.
23. Cree una carpeta dentro del directorio */Quartus* y llámela *WORKSPACE* como indica la imagen.

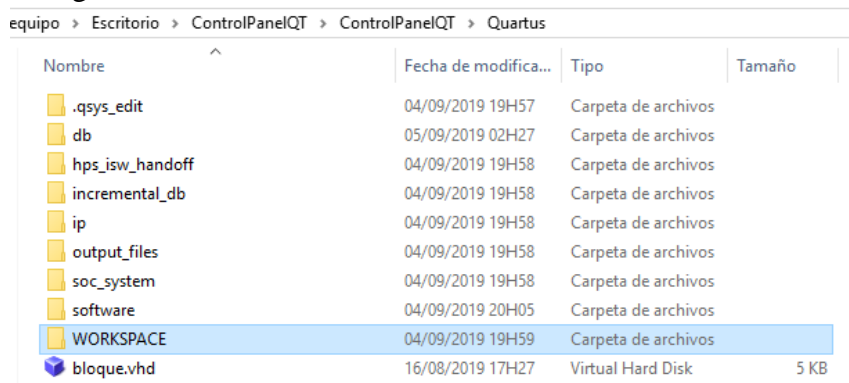


Ilustración 20 - Espacio de trabajo WORKSPACE

24. Para programar el microprocesador. Abra *Eclipse* desde el menú de *Quartus*, *Tools* -> *Nios II Software Build Tools for Eclipse*.

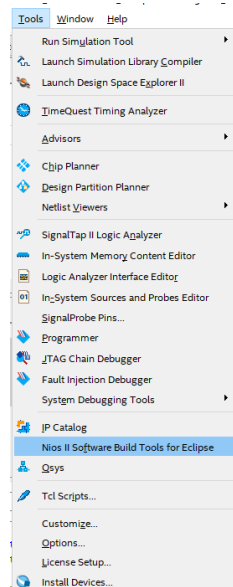


Ilustración 21 - Seleccionar herramienta Eclipse

25. Lo primero que pedirá Eclipse será ubicar en donde va a estar alojado el proyecto que vayamos a crear, seleccione la carpeta *WORKSPACE* que se encuentra dentro de la carpeta del proyecto, como se encuentra en la Ilustración 22. Haga clic en OK.

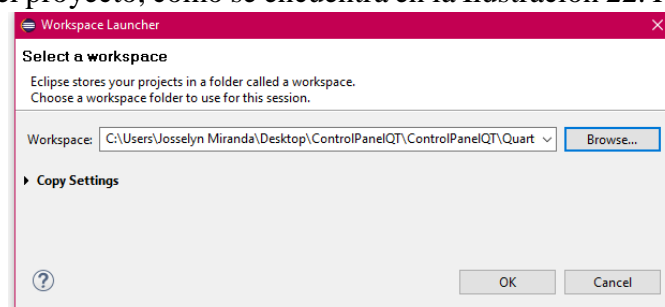


Ilustración 22 - Ubicación del proyecto en Eclipse

26. Se abrirá el entorno de programación de *Eclipse*. Se procederá a crear un nuevo proyecto, para ello seleccione *File*→*New*→*Nios II Application and BSP from Template* como se visualiza en la Ilustración 23.

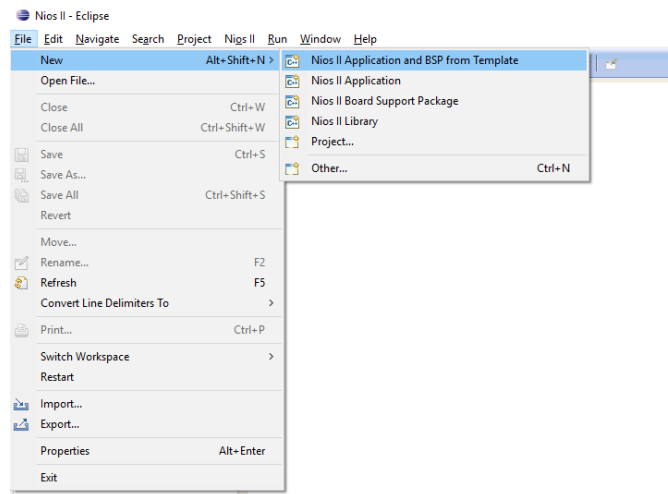


Ilustración 23 - Seleccione una nueva aplicación para NIOS II

27. En la opción *SOPC Information File name*, proceda a buscar el archivo *soc_system.sopcinfo* que tendrá toda la información de la computadora embebida, como se visualiza en Ilustración 24.

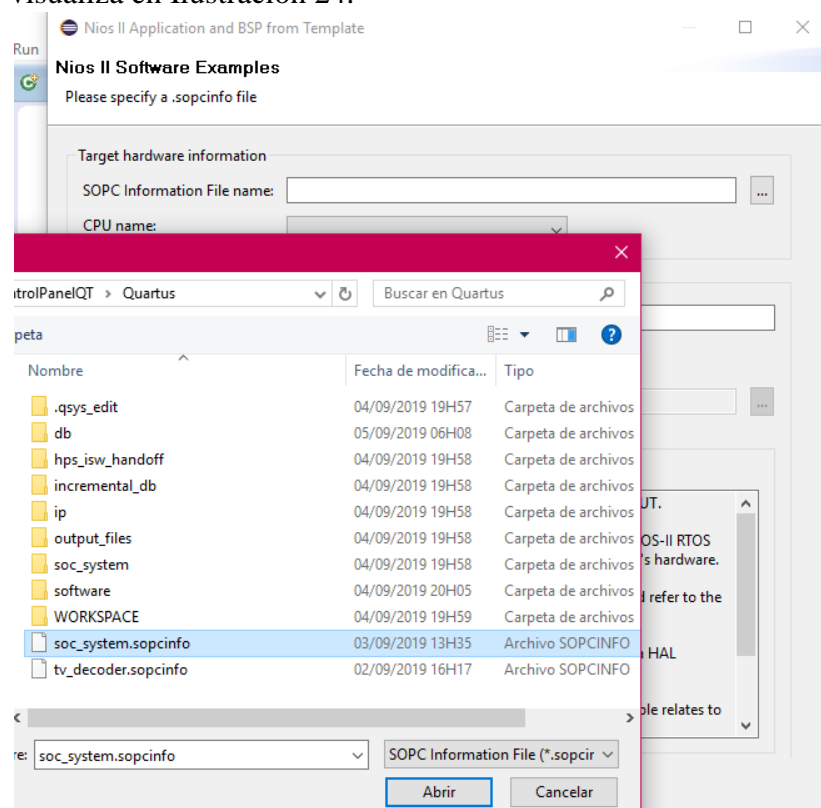


Ilustración 24 - Seleccionar *soc_system.sopcinfo*

28. En la opción *CPU name* se puede utilizar el núcleo disponible en la computadora embebida. Se utilizará el único núcleo *nios2_gen2*.
29. En *Project name* se elegirá el nombre del proyecto, utilice el nombre *FINAL* de preferencia.

30. En *Project Template*, seleccione la opción *Blank Project* para crear nuestro proyecto vacío. Haga clic en *Finish*.

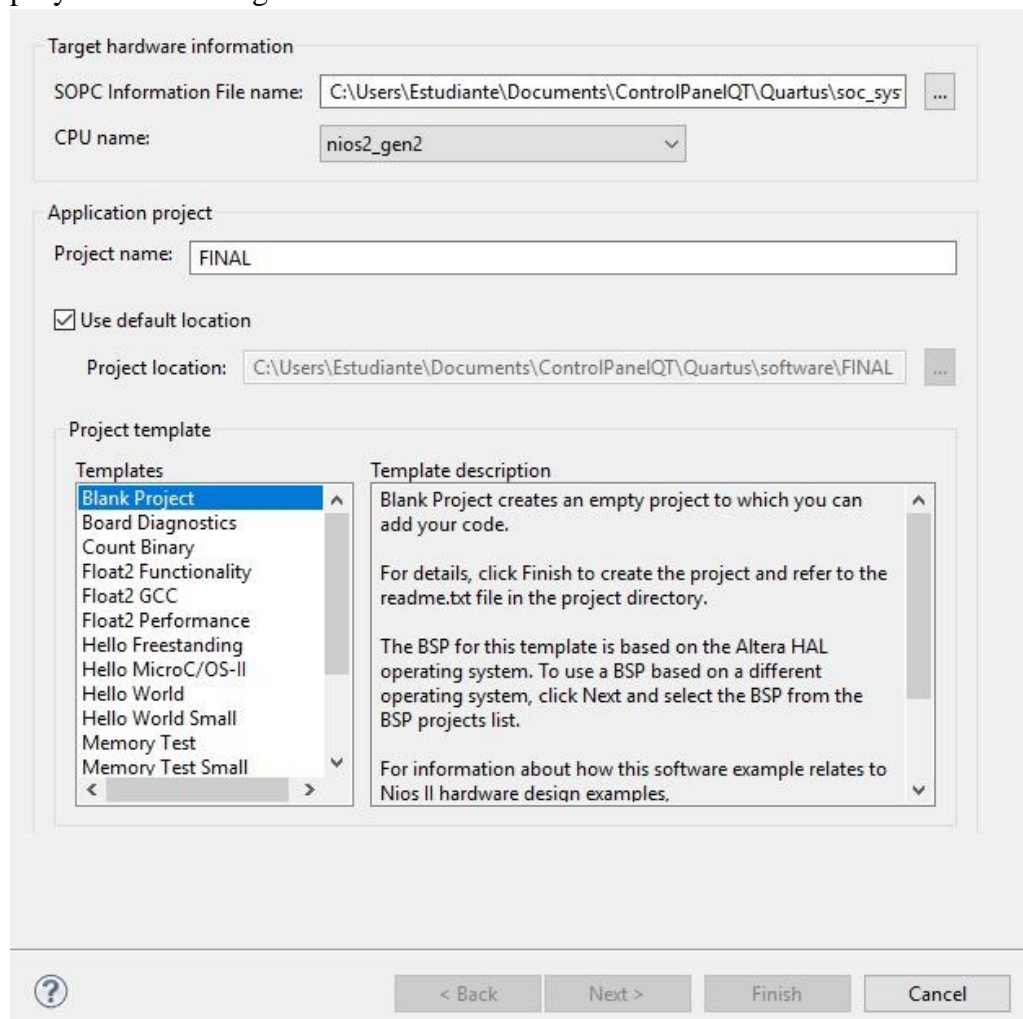


Ilustración 25 - Información para crear proyecto

En la sub-ventana de Project Explorer podrán visualizar dos carpetas: FINAL y FINAL_bsp. En la primera se podrán crear todos los archivos de programación en lenguaje C/C++ para nuestro microprocesador, y la segunda carpeta contiene todos los componentes del hardware, como se visualiza en la Ilustración 25.

31. Se procederá a crear un archivo en lenguaje C, para ello haga clic derecho en la carpeta FINAL y seleccione *New* → *File* como se muestra en la Ilustración 26.

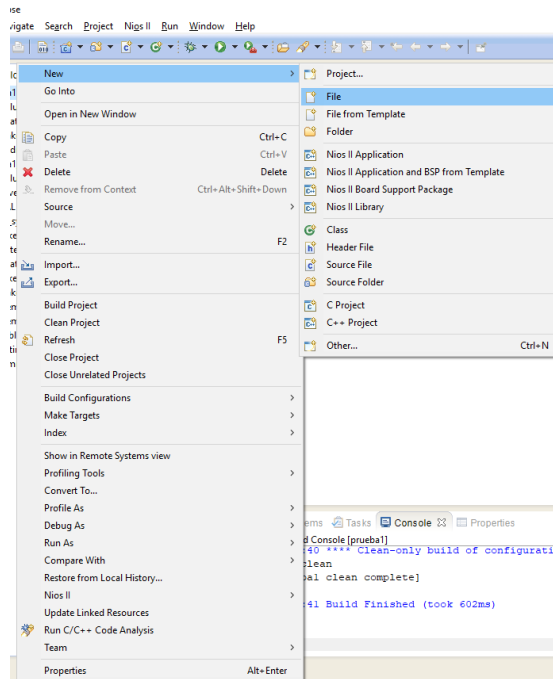


Ilustración 26 - Crear archivo main.c en Eclipse

32. En la ventana *New File*, coloque el nombre *main.c* y haga clic en *Finish* como se muestra en la Ilustración 27.

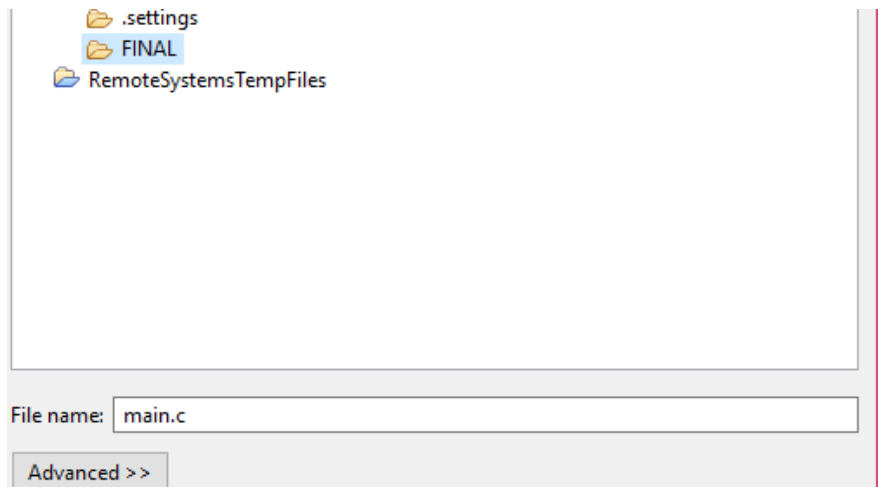


Ilustración 27 - Ventana de la creación de un archivo en lenguaje C

33. Doble clic en el archivo *main.c* creado y copie el siguiente código:

```
#include <stdio.h>
#include <stdlib.h>
#include "io.h"
#include "altera_up_avalon_video_character_buffer_with_dma.h"
```

```
int main (void)
{
    alt_up_char_buffer_dev *dev1;
    alt_up_char_buffer_init(dev1);
```

```

dev1 =
alt_up_char_buffer_open_dev("/dev/video_character_buffer_with_dma_0");
alt_up_char_buffer_clear(dev1);

volatile int * SW_switch_ptr = (int *) 0x00004000;
volatile int * contador = (int *) 0x00001010; // contador
int * memoria = (int *) 0x00080000; //memoria
int SW_value;
char read[50];
char input[100];
char time[100];
int prueba;
int consulta_val;
int _consulta = 25;
IOWR(memoria,0x000a0000,_consulta);

while (1){
    //printf("El resultado es: %d\n",*(contador+2));

    SW_value = *(SW_switch_ptr);
    int band = 0;
    *(contador+1)=0;
    *(contador)=4;
    *(contador)=2;
    alt_up_char_buffer_string(dev1, "Utilice una combinación de
switch (SW0,SW1,SW2) para realizar consulta. ", 0,5);
    alt_up_char_buffer_string(dev1, "Utilice SW9 para mostrar la
consulta realizada. ", 0,6);
    if(SW_value&512){
        alt_up_char_buffer_clear(dev1);
        consulta_val = IORD(memoria,0x000a0000);
        prueba = IORD(memoria,0x00090000);//consultamos el
numero de consulta;
        printf("%d\n",prueba);
        sprintf(read, "%d", consulta_val);
        alt_up_char_buffer_string(dev1, "Mostrando consulta:",
0,1);
        alt_up_char_buffer_string(dev1, "TABLA: sensor_temp:",
0,3);
        alt_up_char_buffer_string(dev1, "Temperatura (ultimo
registro):", 0,4);
        alt_up_char_buffer_string(dev1, read, 50,4);
        alt_up_char_buffer_string(dev1, "En el segundo: ", 0,47);
        sprintf(time, "%d", *(contador+2));
        alt_up_char_buffer_string(dev1, time, 17,47);
        band = 1;
    }
}

```

```

/*
char * read = IORD(memoria,0x00080000);//a
alt_up_char_buffer_clear(dev1);
sprintf(time, "%d", *(contador+2));
alt_up_char_buffer_string(dev1, "Mostrando consulta:",
0,1);

alt_up_char_buffer_string(dev1, read, 0,2);
alt_up_char_buffer_string(dev1, "En el segundo: ", 0,4);
alt_up_char_buffer_string(dev1, time, 17,4);
band = 1;*/
}
if(band==1)
{
usleep(4000000);
SW_value = *(SW_switch_ptr);
while(SW_value&512){
alt_up_char_buffer_string(dev1,
"*****", 0,49);
alt_up_char_buffer_string(dev1, "Regrese el switch a
la posicion original para seguir consultado. ", 0,50);
SW_value = *(SW_switch_ptr);
printf("%d\n",SW_value);
usleep(90000);
}
alt_up_char_buffer_clear(dev1);
band =0;
}
}
}

```

Este código permite leer en memoria y presentarle al usuario por VGA la respuesta de la consulta a la base de datos configurada en el HPS.

34. Luego proceda a hacer clic derecho en la carpeta *FINAL* y seleccione la opción *Build Project* como se muestra en la Ilustración 28.

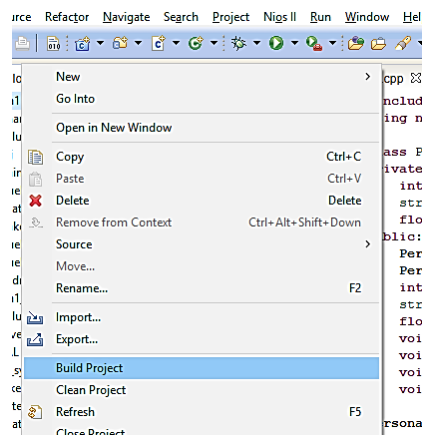


Ilustración 28 - Compilación del proyecto

35. Generar el archivo Hex que utilizará el On-ChipMemory para correr el script cuando inicie el sistema. Clic derecho en el proyecto Make Targets -> Rebuild Last Target.

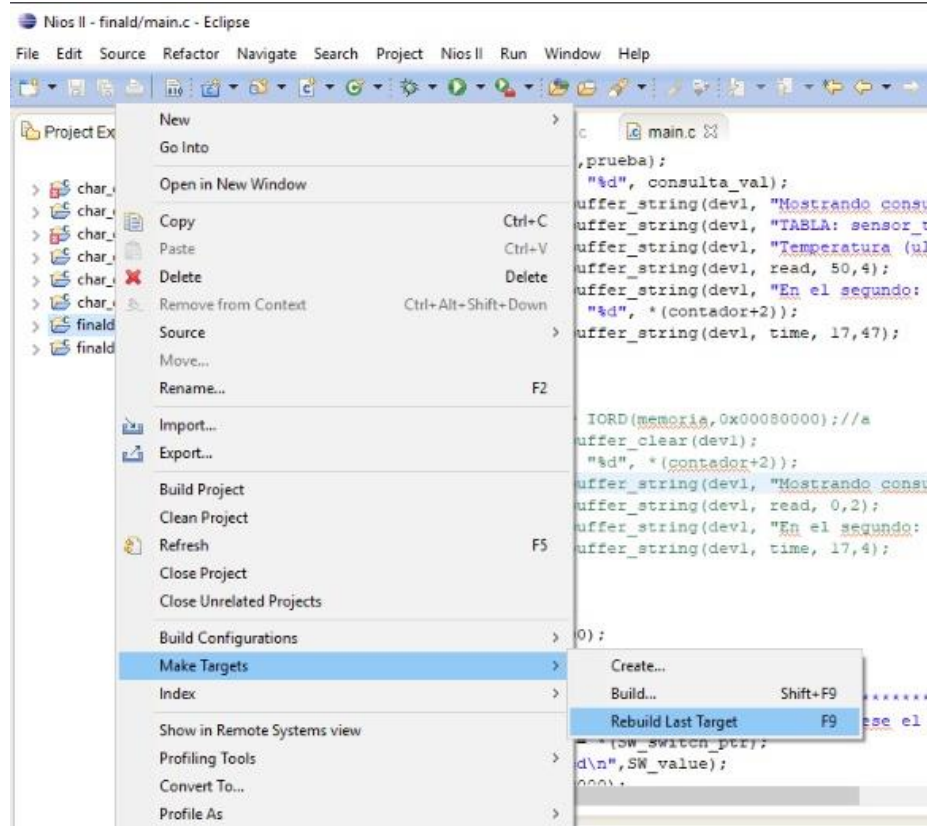


Ilustración 29 - Generar archivo .

36. Seleccionar mem_init_generate. Clic en **Build**

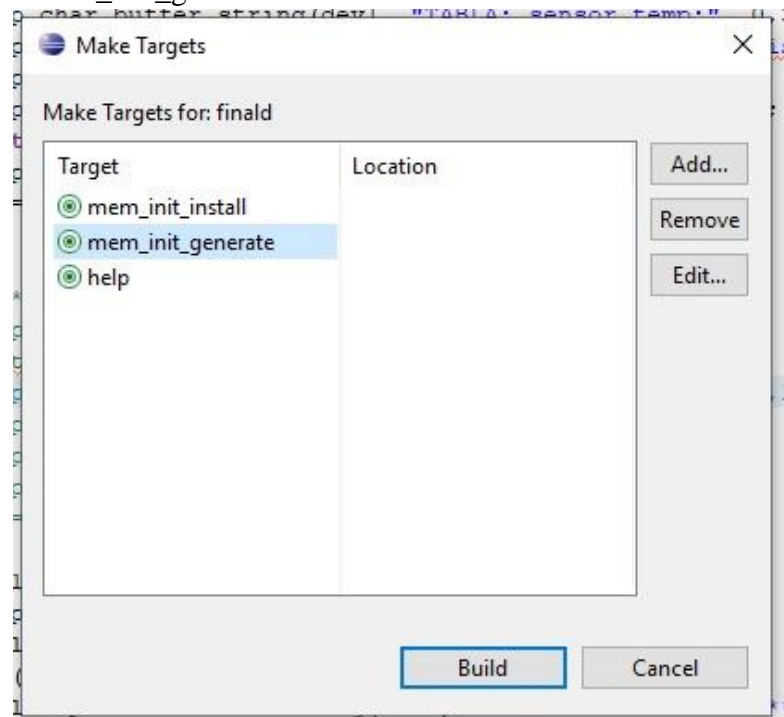


Ilustración 30 - Make Targets

37. Agregar el archivo generado en el paso anterior al `on_chip_memory2` por medio del software *Qsys*. Doble clic en el componente *on_chip_memory*, en la opción *Memory Initialization*-> *Use created initialization file* seleccionar el archivo .hex alojado en la ruta por defecto */Quartus/software/FINAL/mem_init*.

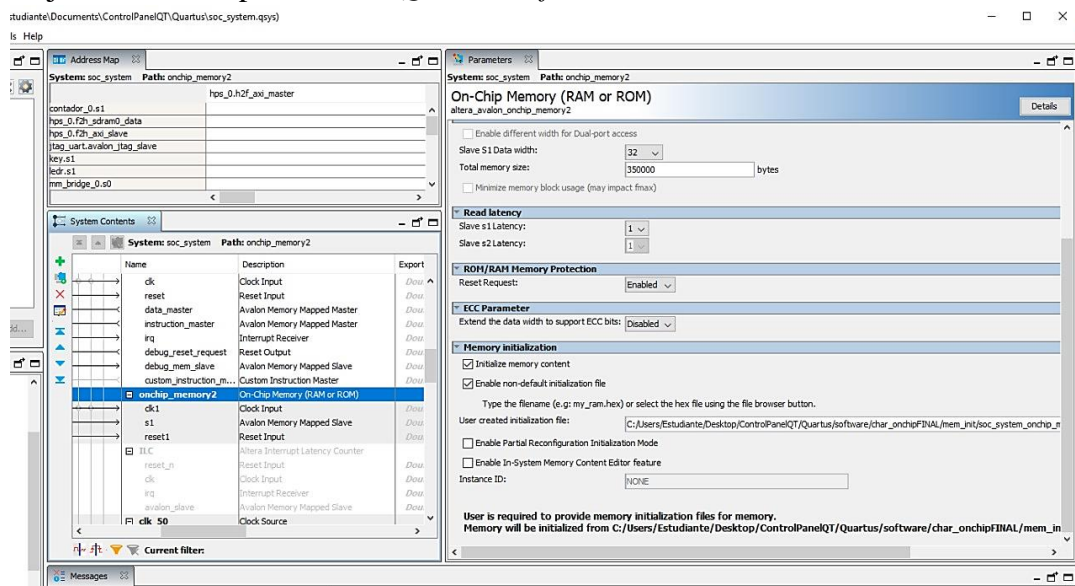


Ilustración 31 - Añadir archivo al NIOS II

38. Repita el paso 16.
39. Repita el paso 7. Una vez que la compilación llegue al 100% se genera un archivo *soc_system.dtb* en el directorio */Quartus* y otro archivo *soc_system.rbf* en el directorio */Quartus/output_files*. Conecte la MicroSD a la computadora donde está trabajando y copie estos dos archivos en el directorio */lib/firmware* de la *MicroSD*.

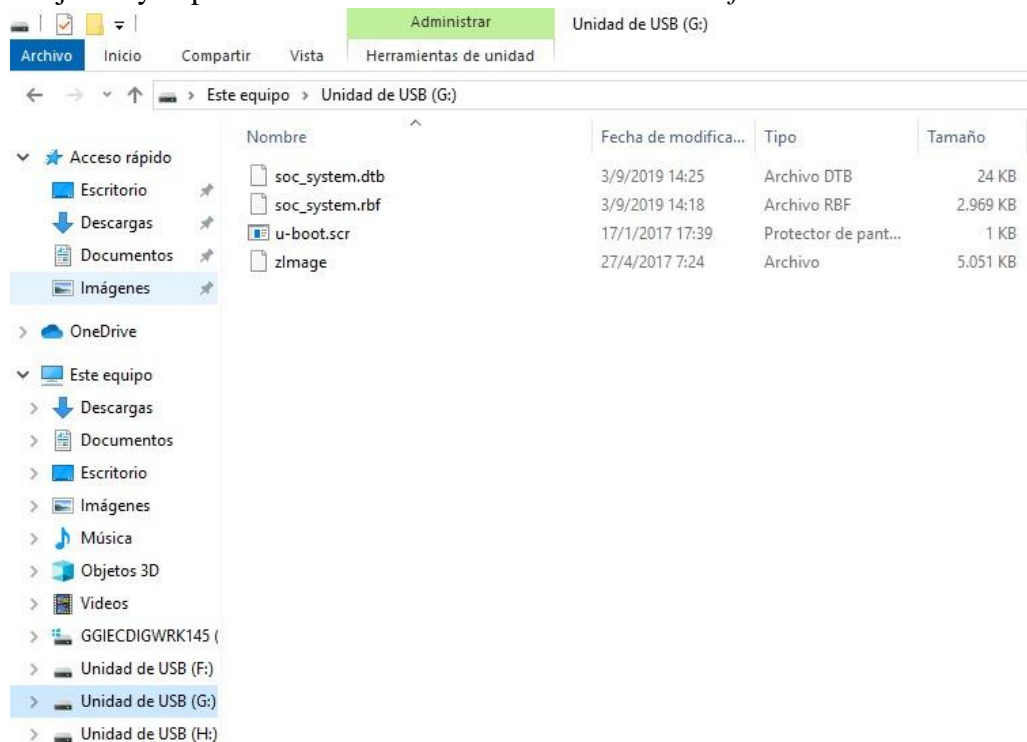


Ilustración 32 - Actualización de archivos de configuración FPGA en Linux

Estos archivos son necesarios para que el Sistema operativo Linux configure la parte de la *FPGA* cuando el sistema arranque, el archivo *dtbo* corresponde a la superposición *FPGA* device tree y *rbf* corresponde al flujo de bits de configuración de la *FPGA*.

40. Conecte la *MICROSD* a la tarjeta de desarrollo DE10 Standard.

41. Cambie el Swtich MSEL[4:0] a 01010.

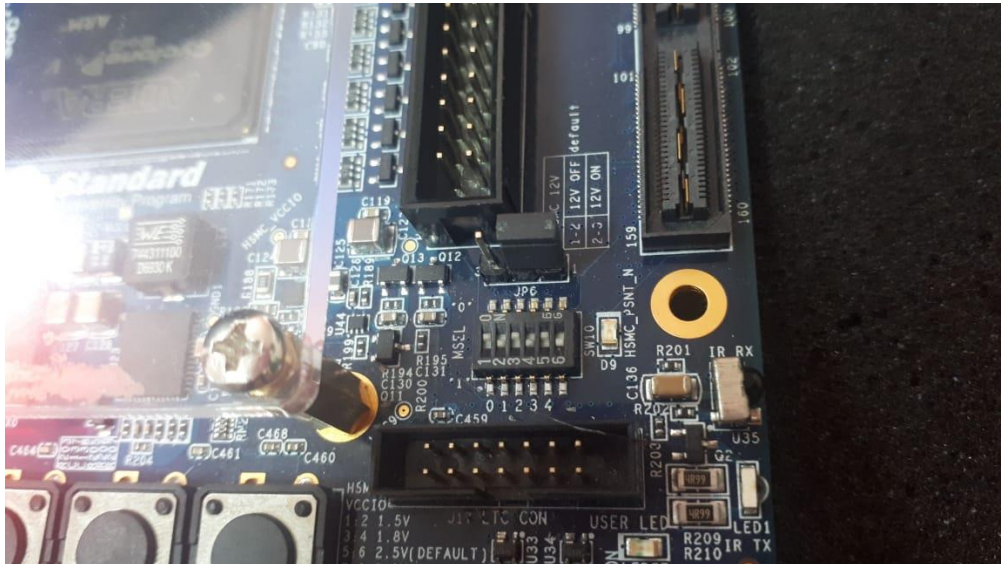


Ilustración 33- MSEL

42. Proceda a conectar la tarjeta de desarrollo DE10 Standard a la fuente de alimentación.

43. Editamos el archivo Makefile que tiene el proyecto original. Reemplazamos el código por el siguiente:

```
#
TARGET = HPS_FPGA_LED

#
ALT_DEVICE_FAMILY ?= soc_cv_av
SOCEDS_ROOT ?= $(SOCEDS_DEST_ROOT)
HWLIBS_ROOT = $(SOCEDS_ROOT)/ip/altera/hps/altera_hps/hwlib
CROSS_COMPILE = arm-linux-gnueabihf-
CFLAGS      = -g -Wall -D$(ALT_DEVICE_FAMILY) -
I$(HWLIBS_ROOT)/include/$(ALT_DEVICE_FAMILY) -
I$(HWLIBS_ROOT)/include/
LDFLAGS = -g -Wall
CC = $(CROSS_COMPILE)gcc
ARCH= arm

build: $(TARGET)
$(TARGET): main.o
    $(CC) $(LDFLAGS) $^ -o $$@
%.o : %.c
    $(CC) $(CFLAGS) -c $< -o $$@
```

```
.PHONY: clean
```

```
clean:
```

```
rm -f $(TARGET) *.a *.o *~
```

44. Creamos un archivo llamado `generate_hps_qsys_header.sh` para generar el encabezado de la arquitectura. Ubicado en el mismo directorio del `soc_system.sopcinfo`.

```
#!/bin/sh
```

```
sopc-create-header-files \
```

```
"/soc_system.sopcinfo" \
```

```
--single hps_0.h \
```

```
--module hps_0
```

45. Creamos un archivo `main.c` en el mismo directorio del Makefile

```
#include <stdio.h>
```

```
#include <string.h>
```

```
#include <stdlib.h>
```

```
#include <unistd.h>
```

```
#include <fcntl.h>
```

```
#include <sys/mman.h>
```

```
#include "hwlib.h"
```

```
#include "socal/socal.h"
```

```
#include "socal/hps.h"
```

```
#include "socal/alt_gpio.h"
```

```
#include "hps_0.h"
```

```
#define HW_REGS_BASE ( ALT_STM_OFST )
```

```
#define HW_REGS_SPAN ( 0x04000000 )
```

```
#define HW_REGS_MASK ( HW_REGS_SPAN - 1 )
```

```
#define ON_CHIP_WR ( 0x00090000 )
```

```
int leerConsulta(int numero);
```

```
void* virtual_base;
```

```
void* sw_addr;
```

```
void* on_chip;
```

```
void* prueba;
```

```
int fd;
```

```
int sw_value;
```

```
char * onchip_value;
```

```
char * texto ;
```

```
int consulta;
```

```
int valor;
```

```
int main (){
```

```
fd=open("/dev/mem",(O_RDWR|O_SYNC));
```



```

virtual_base=mmap(NULL,HW_REGS_SPAN,(PROT_READ|PROT_WRITE),
MAP_SHARED,fd,HW_REGS_BASE);
if( virtual_base == MAP_FAILED ) {
    printf( "ERROR: mmap() failed...\n" );
    close( fd );
    return( 1 );
}
sw_addr=virtual_base + ( ( unsigned long )( ALT_LWFPGASLVS_OFST +
SW_BASE ) & ( unsigned long)( HW_REGS_MASK ) );
on_chip=virtual_base + ( ( unsigned long )( ALT_LWFPGASLVS_OFST +
ONCHIP_MEMORY2_BASE ) & ( unsigned long)( HW_REGS_MASK ) );
prueba=virtual_base + ( ( unsigned long )( ALT_LWFPGASLVS_OFST +
ON_CHIP_WR ) & ( unsigned long)( HW_REGS_MASK ) );
/* on_chip = 0;
//printf("%p\n",&on_chip);

while(1){
sw_value=*(uint32_t *)sw_addr;
usleep(1000000);
printf("%u\n",sw_value);

if(sw_value==1){
    consulta = 1;
    system("./mostrar.sh");
    valor = leerConsulta(consulta);
    //printf("%p\n",&on_chip);
    printf("%d\n",valor);
    if (valor == -1){
        valor = 0; //Valido error de archivo
    }
    *(uint32_t *)on_chip=valor;
    *(uint32_t *)prueba=12;

    printf("Entro en el SW 1\n");
}
if(sw_value==2){
    consulta = 2;
    system("./mostrar.sh");
    valor = leerConsulta(consulta);
    //printf("%p\n",&on_chip);
    printf("%d\n",valor);
    if (valor == -1){
        valor = 0; //Valido error de archivo
    }

    *(uint32_t *)on_chip=valor;
    printf("Entro en el SW 2\n");
}
}

```



```

}
if(sw_value==4){
    consulta = 3;
    system("./mostrar.sh");
    valor = leerConsulta(consulta);
    //printf("%p\n",&on_chip);
    printf("%d\n",valor);
    if (valor == -1){
        valor = 0; //Valido error de archivo
    }
    *(uint32_t *)on_chip=valor;
    printf("Entro en el SW 3\n");
}
/*
if(sw_value==2){
    *(char *) on_chip = (char)"Hola Mundox";
    onchip_value=*(char *)on_chip;
    printf("%s\n",onchip_value);
    printf("Entro en el SW 2\n");
}
*/
}
return 0;
}

int leerConsulta(int numero){
    FILE *file; //puntero del archivo
    int value=-1;
    char cadena [100];
    char texto [20];
    int contador = 0;
    file = fopen ("consulta.txt","r"); //nombre del archivo a leer
    if (file==NULL){
        printf("\nError de apertura del archivo.");
    }
    else{

        while(feof(file)==0) //revisa lineas de archivo
        {
            fgets(cadena,100,file);          //extrae la cadena
            //lee lineas de archivo hasta donde haya salto de linea

            if(contador + 1 == numero){
                strcpy(texto, cadena);
                value=atoi(cadena);
                break;
            }
        }
    }
}

```

```

        contador = contador + 1;

    }
    if(value == -1){ //validación si no se inicializa
        strcpy(texto, "404 Not Found");
        value = 0;
    }

    /*
    printf("Caracter: %s\n", texto);
    printf("Numero: %d\n", value);*/
}
fclose(file); //cerrar el archivo y retornar valor
return value;
}

```

46. Abrir el programa SoC EDS Command Shell. Ubicarnos en el directorio de nuestro proyecto y ejecutar las siguientes líneas:

```

chmod +x generate_hps_qsys_header.sh
make

```

47. Si nuestro ordenador está en red con la FPGA podemos utilizar el siguiente comando para pasarnos el archivo ejecutable que se generó, Ilustración 40.

```

scp <nombre del archivo> root@<IP del HPS>:<directorío destino>
scp HPS_FPGA_LED root@200.126.14.146:/home/root/HPS

```

48. Accedemos al sistema operativo Linux por SSH utilizando el software PUTTY, le asignamos todos los privilegios al archivo ejecutable y lo ejecutamos, Ilustración 41.

```

chmod +x HPS_FPGA_LED
./HPS_FPGA_LED

```

Interacción con el usuario

Una vez que se esté ejecutando el archivo HPS_FPGA_LED generado en el último paso de Configurar Arquitectura podemos interactuar con los tres primeros y el último switch de la tarjeta DE10 Standard, estos permiten realizar diferentes consultas a la base de datos. Además podemos usar la Botonera del HPS para reiniciar el sistema operativo LXDE.

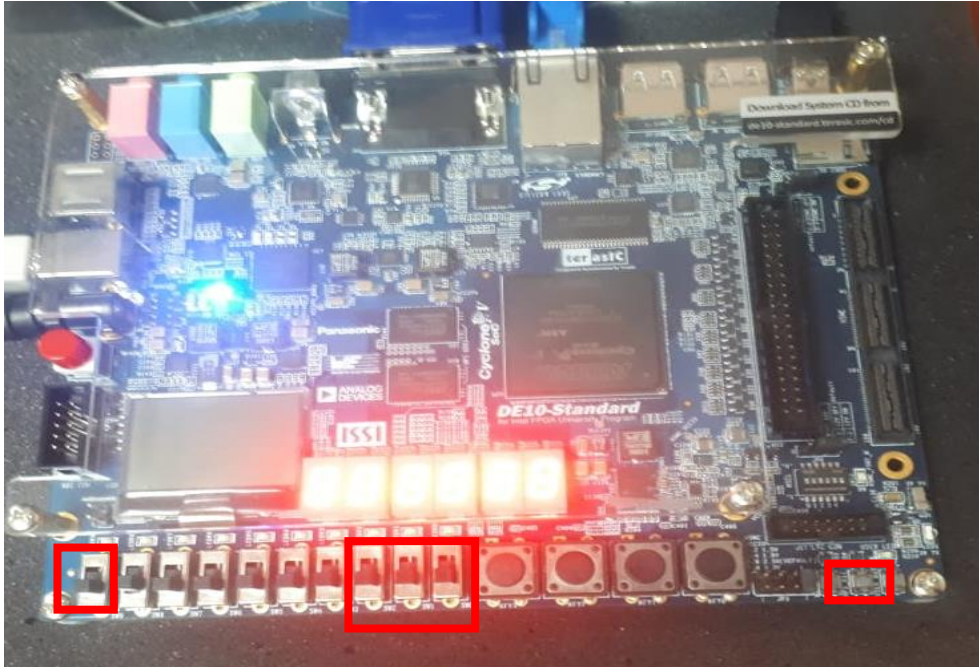


Ilustración 34 - Periféricos usados

Al activar el SW0 se realiza la consulta al campo *valor* del primer dispositivo de la tabla ultimoRegistro y al activar el SW9 se muestra en pantalla el resultado de la consulta.



Ilustración 35 - SW0

Al activar el SW1 se realiza la consulta al campo *valor* del segundo dispositivo de la tabla ultimoRegistro y al activar el ultimo SW se muestra en pantalla el resultado de la consulta.



Ilustración 36 - SW1

Al activar el SW2 se realiza la consulta al campo *valor* del tercer dispositivo de la tabla ultimoRegistro y al activar el ultimo SW se muestra en pantalla el resultado de la consulta.



Ilustración 37- SW2

ANEXOS

ENLACE EN DRIVE

Carpeta compartida con acceso a lectura, en este enlace se encuentra alojado el Documento IEEE y el presente documento (Manual Técnico).

<https://drive.google.com/drive/folders/1JTLbT8e3Wnj9QqhATpKR9N5PE9A0YCYby?usp=sharing>

ENLACE EN GITHUB

https://github.com/jocammir/Sistema_gestion_base_de_datos_FPGA_HPS_DE10Standard.git

Dado que es un repositorio privado debe solicitar permisos al siguiente correo electrónico: jocammir@espol.edu.ec

FOTOS

```
mysql> select * from ultimoRegistro;
+----+-----+-----+
| id | dispositivo | valor |
+----+-----+-----+
| 1 | sensor1    | 99    |
| 2 | sensor2    | 50    |
| 3 | sensor3    | 43    |
+----+-----+-----+
3 rows in set (3.61 sec)

mysql> update ultimoRegistro set valor=100 where id=1;
Query OK, 1 row affected (0.06 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from ultimoRegistro;
+----+-----+-----+
| id | dispositivo | valor |
+----+-----+-----+
| 1 | sensor1    | 100   |
| 2 | sensor2    | 50    |
| 3 | sensor3    | 43    |
+----+-----+-----+
3 rows in set (0.00 sec)

mysql> select * from microcontrolador;
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | nombre      | bateria | firmware | num_sensores | description | dtm | fabricante | num_procesadores | num_periferos |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | AVRMini     | 56.8    | v1.2     | 3 | Desarrollado por Pascal Stang | 2010-11-25 11:11:11 | Stang | 2 | 5 |
| 2 | Wiring      | 46.8    | v2.2     | 4 | Proyecto abierto iniciado por Hernando Barragn | 2011-11-25 16:11:11 | Instituto Avric | 3 | 4 |
| 3 | Arduino     | 96.8    | v3.2     | 3 | Plataforma de cdigo abierto | 2012-11-25 10:11:11 | Arduino | 1 | 6 |
| 4 | Basic Stamp | 26.8    | v1.0     | 5 | Ejecuta programas en lenguaje PBASIC | 2013-11-25 10:11:11 | Stang | 3 | 5 |
| 5 | Raspberry Pi 3 | 16.8    | v3.0     | 6 | Capaz de correr distribucin de linux | 2014-11-25 11:11:11 | Stang | 3 | 5 |
| 6 | Raspberry Pi 4 | 88.7    | v5.0     | 2 | Capaz de correr distribucin de linux | 2015-11-25 12:11:11 | Stang | 4 | 5 |
| 7 | BeagleBone   | 11.9    | v1.0     | 3 | Capaz de soportar ambientes no favorables | 2016-11-25 13:11:11 | Stang | 5 | 4 |
| 8 | FPGA de10-standard | 1.1    | v1.0     | 3 | Desarrollado por Intel | 2017-11-25 14:11:11 | Intel | 6 | 4 |
| 9 | FPGA de10-nano | 97.6    | v2.0     | 4 | Desarrollado por Intel | 2018-11-25 15:11:11 | Intel | 2 | 5 |
| 10 | FPGA de10-nano | 12.3    | v3.0     | 5 | Desarrollado por Intel | 2019-11-25 16:00:00 | Intel | 3 | 3 |
| 11 | FPGA de10-115 | 78.9    | v4.0     | 3 | Desarrollado por Intel | 2020-11-25 17:00:00 | Intel | 2 | 2 |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
11 rows in set (0.00 sec)

mysql> update microcontrolador set valor=0 where id=10;
ERROR 1054 (42S22): Unknown column 'valor' in 'field list'
mysql> update microcontrolador set num_procesadores=8 where id=10;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Ilustración 38 - Consulta a las tablas de la base de datos desde Linux

```
COM5 - PuTTY
/* Conectarse a la base de datos */
root@RIO-Standard:~/Comex_mysql/mysqlComexFinal# mysql -u root -p
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 16
Server version: 5.7.26-Debian (Ubuntu)

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> use SISTEMA_RIEGO;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_SISTEMA_RIEGO |
+-----+
| microcontrolador        |
| sensor_temp              |
| sensor_temp2             |
| sensor_temp3             |
| ultimoRegistro           |
+-----+
5 rows in set (0.00 sec)
```

Ilustración 39 - Tablas de la base de datos

```

~/Documents/ControlPanelQT/Quartus
contador_hw.tcl      generate_hps_qsys_header.sh      Makefile      tv_decoder.qsys
db                  hps_0.h                        Makefile2     tv_decoder.sopcinfo
DE10_Standard_FB.bsf hps_common_board_info.xml      output_files  u-boot.scr
DE10_Standard_FB.qpf HPS_FPGA_LED                  soc_system    WORKSPACE
DE10_Standard_FB.qsf hps_isw_handoff               soc_system.dtb
DE10_Standard_FB.sdc hps_sdram_p0_summary.csv       soc_system.dts

GGIECDIGWRK145+Estudiante@GGIECDIGWRK145 ~/Documents/ControlPanelQT/Quartus
$ rm HPS_FPGA_LED

GGIECDIGWRK145+Estudiante@GGIECDIGWRK145 ~/Documents/ControlPanelQT/Quartus
$ make
arm-linux-gnueabihf-gcc -g -Wall -Dsoc_cv_av -IC:/intelFPGA/17.0/embedded/ip/altera/hps/altera_hps/hwlib/include/soc_cv_av -IC:/intelFPGA/17.0/embedded/ip/altera/hps/altera_hps/hwlib/include/ -c main.c -o main.o
arm-linux-gnueabihf-gcc -g -Wall main.o -o HPS_FPGA_LED

GGIECDIGWRK145+Estudiante@GGIECDIGWRK145 ~/Documents/ControlPanelQT/Quartus
$ scp HPS_FPGA_LED root@200.126.14.146:/home/root/HPS
Could not create directory '/home/Estudiante/.ssh'.
The authenticity of host '200.126.14.146 (200.126.14.146)' can't be established.
ECDSA key fingerprint is SHA256:YAVGTDiDj5Pwbx1o4bkeYZtfVcVyKJiITZwZVRnIJP4.
Are you sure you want to continue connecting (yes/no)? yes
Failed to add the host to the list of known hosts (/home/Estudiante/.ssh/known_hosts).
HPS_FPGA_LED 100% 11KB 11.1KB/s 00:00

```

Ilustración 40 - SoC EDS Command Shell

```

root@DE10-Standard:~/HPS# ./HPS_FPGA_LED
4
43
Entro en el SW 3
4
43
Entro en el SW 3
4
43
Entro en el SW 3

```

Ilustración 41 - Ejecución del archivo HPS_FPGA_LED

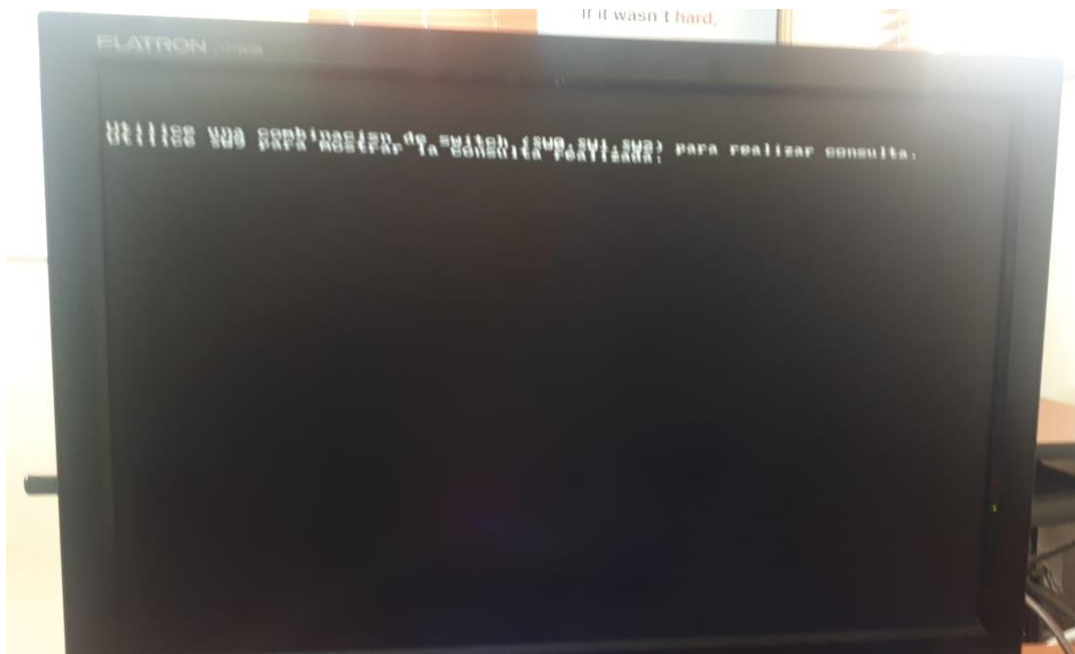


Ilustración 42 - Vista principal del programa



Ilustración 43 - DE10Standard ejecutándose exitosamente