DISEÑO DE PROCESADORES DEDICADOS

Práctica 2
"Síntesis Lógica"
Tarjeta Spartan 3E

Dr. Juan Carlos Herrera Lozada
jlozada@ipn.mx

Instituto Politécnico Nacional
Centro de Innovación y Desarrollo Tecnológico en Cómputo
CIDETEC

Marzo 2013

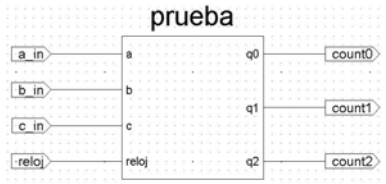
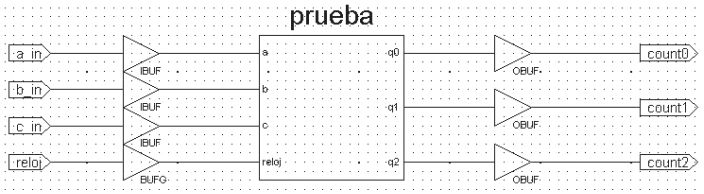
Campo 1: **Datos Personales.**

Campo 2: **Objetivos**.

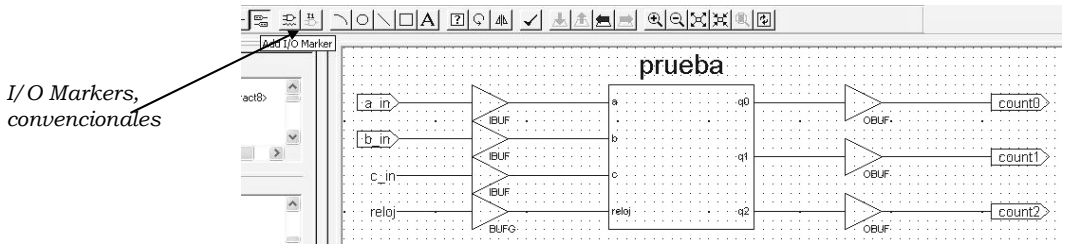
- Síntesis lógica y programación de la tarjeta de desarrollo DIGILENT SPARTAN 3E-STARTER, XC3S500E.

Campo 3: **Desarrollo de la Práctica**.

Antes de comenzar a utilizar la tarjeta, debes adecuar tus diagramas esquemáticos para la implementación física; esto es, independientemente del número de macros de tu esquemático, las variables en el nivel jerárquico más alto (entradas y salidas hacia el exterior) deben tener buffers y pines físicos (PADS). Dentro de una macro no puede haber PADS, sino símbolos a conectores de jerarquía baja. Lo anterior es importante debido a que el simulador no te indicará un error, pero las herramientas de síntesis sí lo harán. Observa la siguiente conversión.

Para simulación y/o para generar una macro	*Para implementación física*
	

Cada Pin debe incluir su respectivo buffer, por ejemplo, *IBUF* es un buffer sólo para entradas y *OBUF* es un buffer sólo para salidas, ambos los obtienes de la biblioteca de *símbolos* dentro de la edición esquemática en la categoría *IO*. Si analizas las categorías indicadas podrás observar que es posible considerar buffers de entrada o de salida múltiples. De momento, sólo utilizaremos individuales.
BUFG es un buffer global que es exclusivo para señales de reloj y lo obtienes en la categoría *BUFFER*.
Los PADs son los mismos *I/O Markers* que utilizas para las terminales en cualquier diagrama, disponibles en el menú principal de la edición esquemática. Es importante etiquetar las variables para que exista una referencia real al momento de asignar pines físicos.



El diseño con el que comenzaremos a utilizar la tarjeta (Digilent Spartan 3E-Starter, XC3S500E) es un contador ascendente de 8 bits, que envía datos hacia los 8 LEDs de propósito general, rotulados descriptivamente como *LD7, LD6, …, LD0*. Tienes que descargar el manual de usuario de la tarjeta en cuestión y el código en VHDL para la práctica 2, ambos materiales disponibles en línea (página del curso). Si fuera el caso que se utilizara la tarjeta Spartan 3A/3AN 3S700A, los pasos aquí descritos son similares.

3.1 Generando el símbolo esquemático del contador de prueba

El código en VHDL listado a continuación describe el diseño propuesto. Para iniciar la práctica, revisa la sintaxis

de este código y genera un símbolo esquemático con él. Consta de cuatro entradas: la señal de reloj (*CLK*), dos líneas de reset (*RESET* y *RESET_COUNTER*) y una línea de paro (*STOP*). Como salidas del símbolo esquemático obtendrás los 8 bits del contador ascendente (*ld7, ld6, ld5, ld4, ld3, ld2, ld1, ld0*).

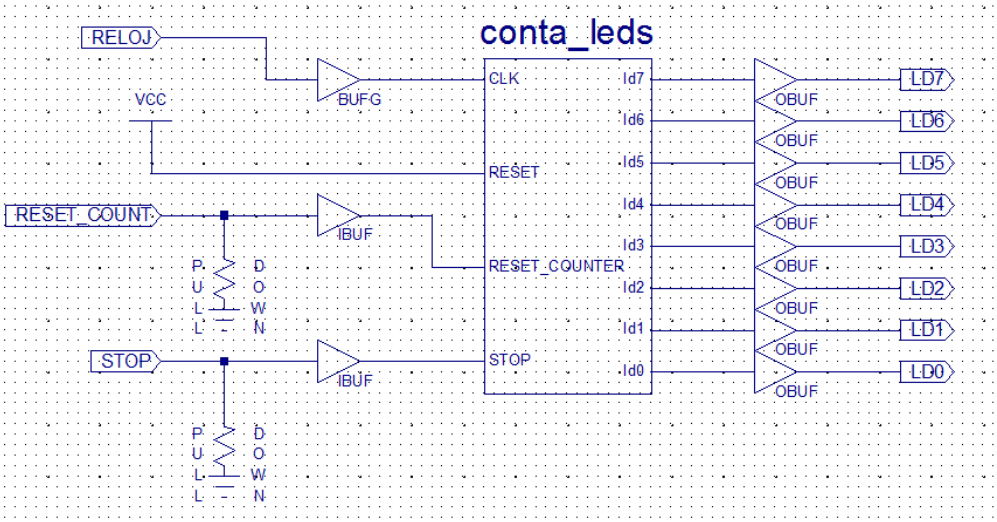
```
----------------------------------------------------
--Maestría en Tecnología de Cómputo
--CIDETEC IPN, Diseño de Procesadores Dedicados
--Juan C. Herrera L. Prueba de tarjeta Spartan 3E, USB.
LIBRARY IEEE;
USE IEEE.std_logic_1164.ALL;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

ENTITY conta_leds IS
  PORT(
         CLK, RESET, RESET_COUNTER, STOP: in STD_LOGIC;
         ld7, ld6, ld5, ld4, ld3, ld2, ld1, ld0: out STD_LOGIC -- LEDs
           );
  END conta_leds;

ARCHITECTURE archconta_leds OF conta_leds IS
signal frec_contb: STD_LOGIC;
signal COUNT: STD_LOGIC_VECTOR (24 downto 0);-- 50 MHz, DIVISOR DE TIEMPO
signal COUNTB: STD_LOGIC_VECTOR (7 downto 0);-- CONTADOR BINARIO
begin
divisor: process (CLK, RESET, COUNT)
begin
   if RESET='0' then
   COUNT <= "0000000000000000000000000";
   elsif CLK='1' and CLK'event then
               COUNT <= COUNT + 1;
         else
               COUNT <= COUNT;
   end if;
       frec_contb <= COUNT(24);
end process divisor;

contabin: process (RESET_COUNTER, frec_contb, STOP)
begin
   if RESET_COUNTER='0' then
       COUNTB <= "00000000";
       elsif  STOP='0' then
      COUNTB <= COUNTB;
   elsif frec_contb='1' and frec_contb'event then
               COUNTB <= COUNTB + 1;
         else
               COUNTB <= COUNTB;
   end if;
end process contabin;

asignaleds: process (COUNTB)
begin
--  CONTB(7:0) -> ld7|ld6|ld5|ld4|ld3|ld2|ld1|ld0
ld0 <= COUNTB(0);
ld1 <= COUNTB(1);
ld2 <= COUNTB(2);
ld3 <= COUNTB(3);
ld4 <= COUNTB(4);
ld5 <= COUNTB(5);
ld6 <= COUNTB(6);
ld7 <= COUNTB(7);
end process asignaleds;

END archconta_leds;
```

El circuito completo consta de la macro en VHDL con tres procesos: el primero (`divisor`) es el encargado de dividir el *reloj maestro de la tarjeta* de *50 MHz* a una frecuencia más baja que pueda ser utilizada en diseños de prueba. Este proceso de forma independiente y con modificaciones simples, te será muy útil para otros proyectos.

El segundo proceso, es propiamente el contador de 8 bits (`contabin`), que cambia de estados con la frecuencia dividida proveniente del proceso anterior. El tercero de los procesos (`asiganleds`) permite la asignación individual de cada bit del contador con un LED de la tarjeta de desarrollo.

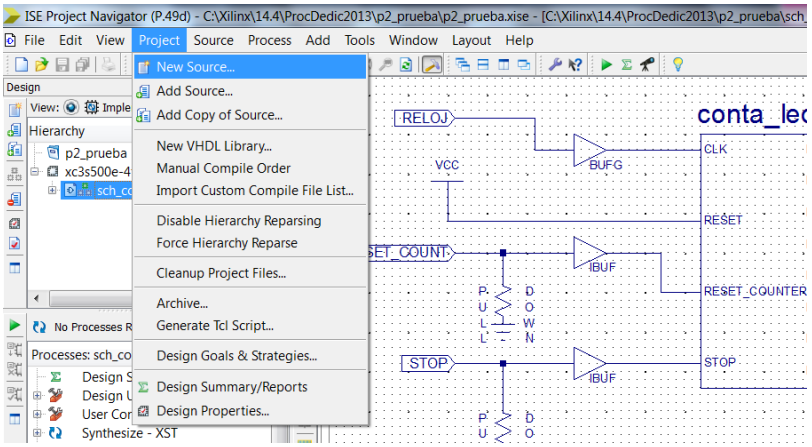
A continuación se muestra el diagrama esquemático que debes completar; para todas las líneas utiliza los mismos nombres que indica el diagrama esquemático. Observa que *RESET* está conectado a *VCC* con la intención de que siempre esté activo el contador que divide el tiempo. El componente *VCC* está disponible en la biblioteca de símbolos de la edición esquemática.

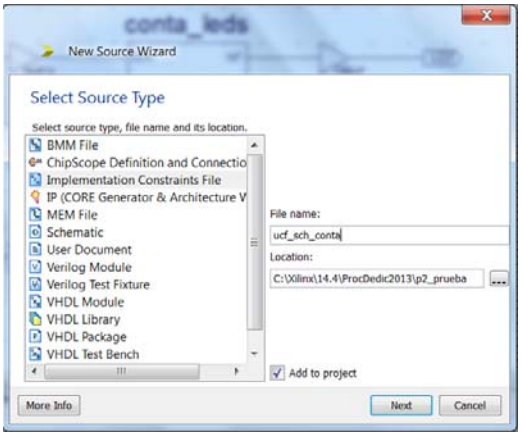
La señal de reloj se introducirá a través de un PAD etiquetado como *RELOJ* al que se le conectará un *BUFG* dado que se trata de una señal variante en el tiempo y que se puede conectar a varios elementos a la vez. De acuerdo a las características específicas de esta tarjeta de desarrollo (revisar manual de la tarjeta SPARTAN 3E-STARTER), tanto los *push-buttons* como el *dipswitch*, no tienen resistencias alambradas, por lo que se requiere utilizar resistencias de *Pull-Down* localizadas en la biblioteca de símbolos de la edición esquemática.
Observa cómo se alambraron las entradas *STOP y RESET_COUNT* con su respectivo *IBUF*. Cada una de las salidas del contador (para cada LED de la tarjeta) tiene su respectivo *OBUF*.



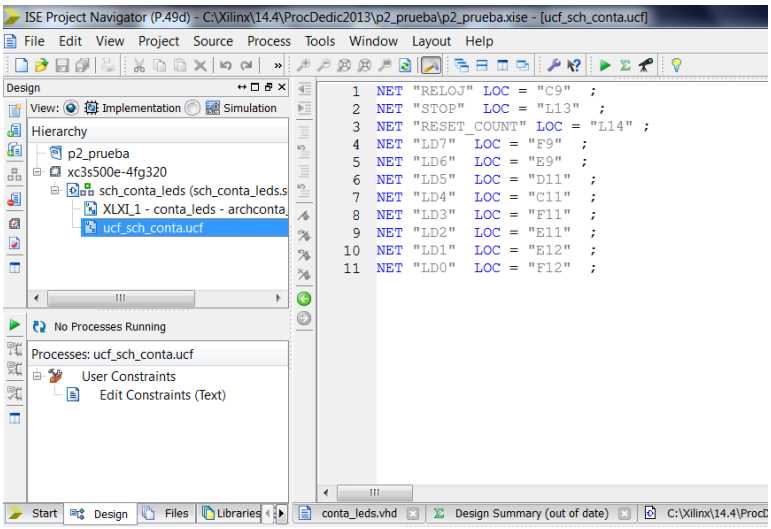
3.2 Asignando número a cada pin físico

Para asignar un número de pin, debes crear un archivo de pines *(UCF – User Constraint File)* que determinará la localización de cada pin desde un archivo de texto. Para ello, accede al menú principal y en *Project* selecciona *New Source*, posteriormente elige *Implementation Constraints File* y nombra tu archivo.





Notarás que el archivo *UCF* se adicionará automáticamente a tu proyecto. En *sources*, activa el archivo *UCF* con el *Mouse*, accediendo a los procesos que podemos realizar sobre este archivo. Da doble clic sobre *Edit Constraints (Text)* y captura los datos indicados en la siguiente figura.




```
NET "RELOJ" LOC = "C9"  ;
NET "STOP"  LOC = "L13"  ;
NET "RESET_COUNT" LOC = "L14" ;
NET "LD7"  LOC = "F9"  ;
NET "LD6"  LOC = "E9"  ;
NET "LD5"  LOC = "D11"  ;
NET "LD4"  LOC = "C11"  ;
NET "LD3"  LOC = "F11"  ;
NET "LD2"  LOC = "E11"  ;
NET "LD1"  LOC = "E12"  ;
NET "LD0"  LOC = "F12"  ;
```

`LOC = C9`, indica que la variable *RELOJ* del diagrama esquemático está asignada al *pin C9* del FPGA en cuestión De igual forma, en *L13* se tiene un *push-button*. La información de los pines "amarrados" para los leds, así como otros que puedes utilizar con propósito general (*push-buttons, dipswitch, LCD*), se consulta en el manual de usuario de la misma tarjeta. Es muy importante que el archivo *UCF* creado coincida con los nombres de las variables declaradas en los *I/O Markers (PADs)* de tu diagrama esquemático.

3.3 Implementación

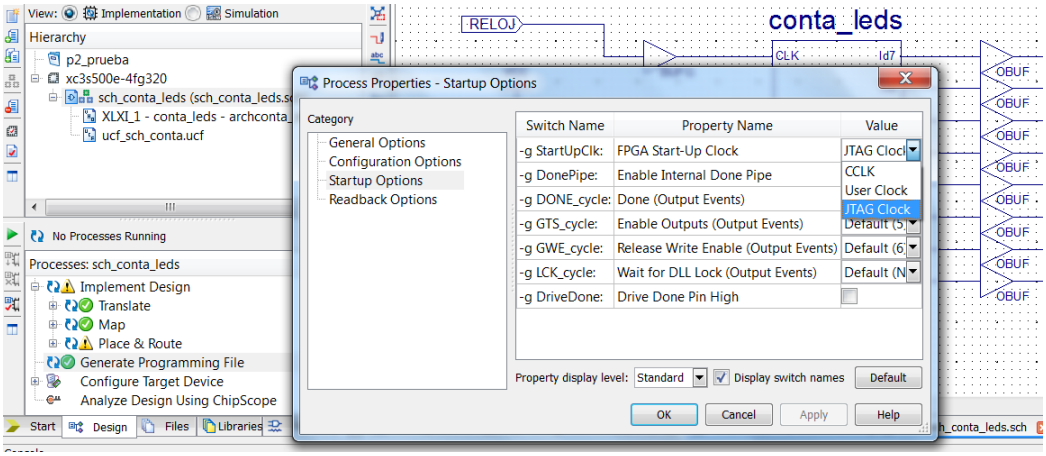
Para la *implementación* de tu circuito (todas las fases del diseño hasta el archivo de bits que configurará al FPGA) debes estar en el *Project Navigator*, manipulando el archivo del diagrama esquemático que representa la jerarquía más alta de tu diseño y para el cual generaste un archivo *UCF* que determinará qué número de pin se le asignará a cada I/O Marker.

Comprueba que sólo esté el esquemático que deseas implementar (preferentemente); esto lo puedes verificar en la lista sensitiva de tu proyecto. Expande la opción *Implement Design* del área *Process for Source*, tal y como lo muestra la siguiente pantalla, a la izquierda. Posteriormente, trasládate a *Place & Route* y da doble clic sobre esta opción. Notarás que si todo es correcto, comenzarán a validarse todos los procesos de manera automática.

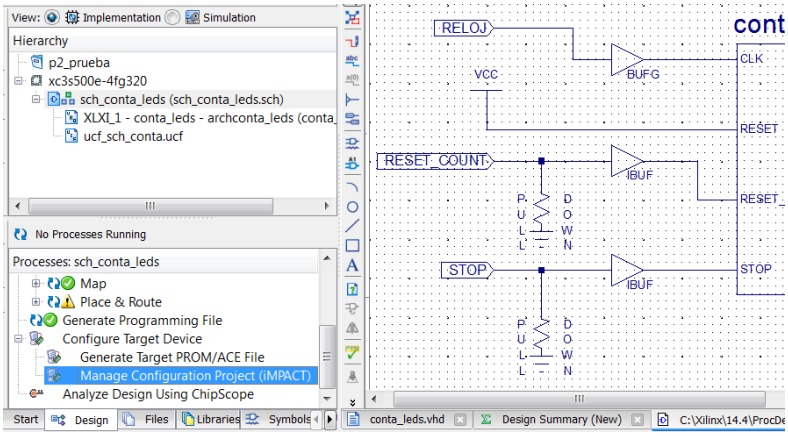
Elige el archivo *Place & Route Report* y observa la cantidad de recursos utilizados dentro del FPGA. Podrás advertir que el parámetro comparado es el número de *SLICEs* o *CLBs* (*Bloques Lógicos Configurables*) dentro del dispositivo. En ocasiones, al momento de realizar la implementación física obtienes errores derivados de conexiones flotantes (símbolos que aparecen juntos pero que no están conectados), pines definidos con el mismo nombre varias veces, pines que no tienen su respectivo buffer o señales de reloj (señales de entrada que conectas a varios bloques) que no tienen un *BUFG* (buffer global) asignado. Abre el archivo *Pad Report* para verificar la asignación física de pines.

3.4 Programación

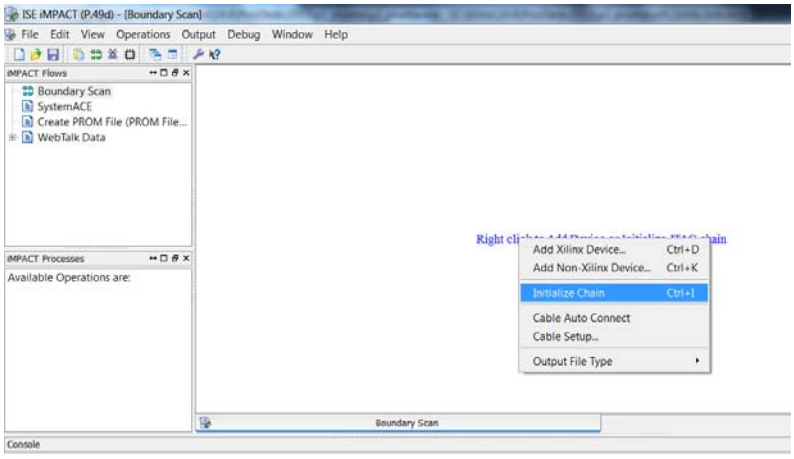
A partir de este momento debes conectar la tarjeta. Cuando lo hagas por primera vez, automáticamente se instalará el driver correspondiente.
Posiciónate con un clic del *Mouse* sobre la opción *Generate Programming File* del área *Process for Source* y con el botón derecho del mismo *Mouse* accede a las propiedades de esta opción y dirígete a la pestaña *Startup Option*, y ahí, cambia la propiedad *Start-Up Clock* a un valor *JTAG Clock* (que es el tipo de cable que utilizaremos).



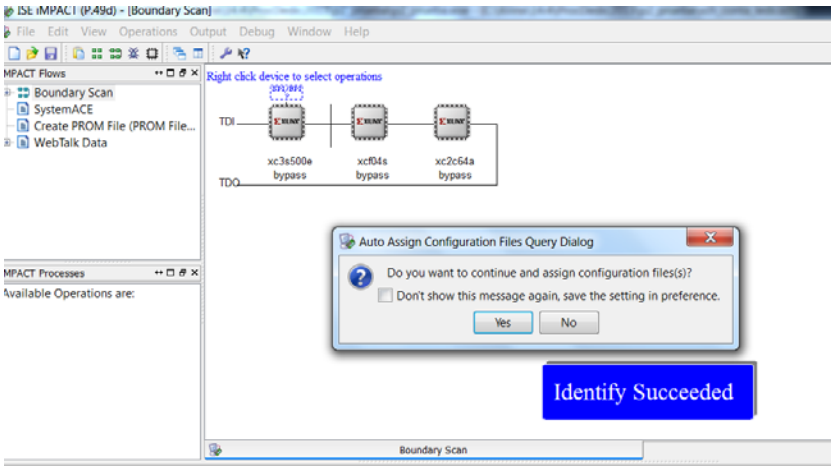
Al aceptar las modificaciones, regresa a desglosar los archivos de *Generate Programming File* y activa *Manage Configuration Project (iMPACT)*.



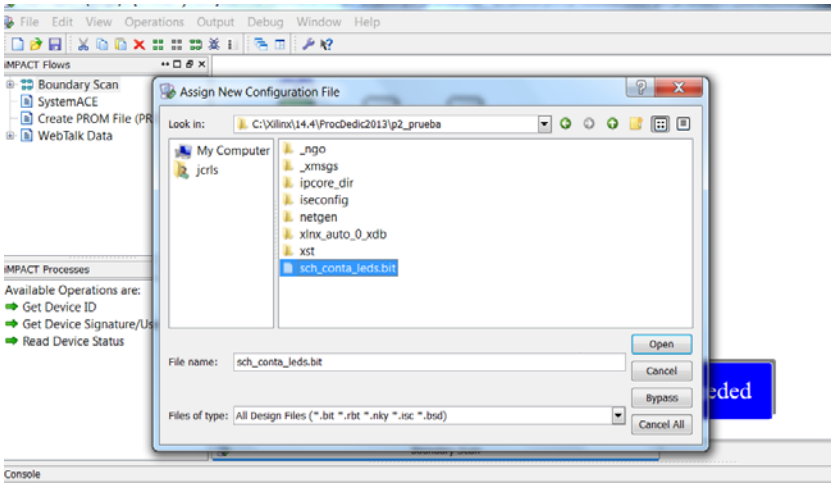
La herramienta *iMPACT*, te mostrará una pantalla en blanco. Da doble clic sobre *Boundary Scan*. Notarás que en el centro de la pantalla aparece un mensaje que te indica que con el botón derecho del Mouse puedes inicializar una cadena de programación.



Será importante iniciar esta cadena, seleccionando *Initialize Chain*; al hacerlo, observa que se muestran tres chips: el primero, a la izquierda es el FPGA Spartan 3E XC3S500E, los dos siguientes integrados son memorias que contiene la tarjeta y que se puede programar para mantener una configuración que se leería al alimentar la tarjeta. Notarás que están conectados entre sí, conformando una cadena (*chain*).



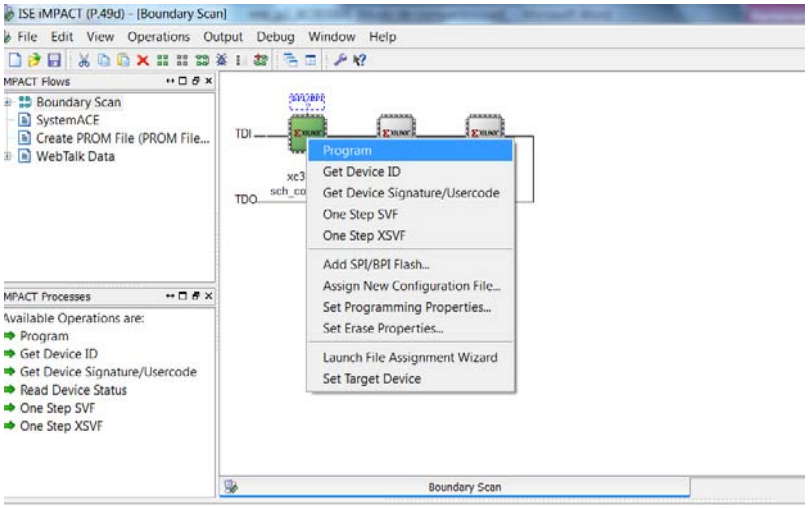
Aparecerá, primeramente, un cuadro de diálogo que te solicitará el archivo de bits con el que se programará el FPGA, tal y como se aprecia en la siguiente figura. Busca en la carpeta de tu proyecto el archivo *. *bit* que corresponde al mismo nombre de tu diagrama esquemático pero con la extensión *. *Bit*. Observa la figura siguiente.

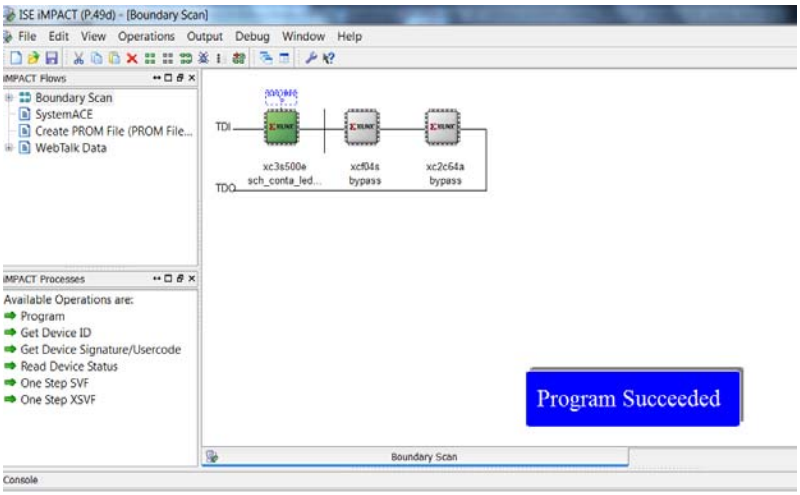


Posteriormente continuará el proceso para programar las memorias; ignóralos utilizando la opción *Bypass* ya que nosotros no programaremos estos dispositivos.

Una vez concluido el proceso de la cadena, posiciónate sobre el FPGA (chip de la izquierda en la ventana de iMPACT) y con el botón derecho del *Mouse* acciona *Program* para descargar tu archivo **.bit* al FPGA. También puedes hacerlo desde *Operations -> Program*. Aparecerán las opciones de la programación; mantenlas por omisión, presionando *OK*. Se debe visualizar como en la siguiente figura. Comenzará la descarga del archivo de bits hacia el FPGA de la tarjeta de desarrollo, tal y como se observa en las siguientes pantallas.

Está misma metodología es extensiva para todos tus proyectos. Es importante que recuerdes que el FPGA tiene algunos pines asignados para su configuración previa, que después cambian de función para acoplarse a tu diseño, por lo mismo es recomendable que desconectes momentáneamente GND de tu circuito armado (en el caso de tener componentes externos a la tarjeta), es decir, *que no haya nada que esté conectado a GND antes de que descargues la configuración a la tarjeta.*





Campo 4: **Validación de la Práctica.**

1. *(45 puntos)* Diseña un contador ascendente – descendente de 8 bits, que con ayuda de una variable externa asignada a un interruptor, te permita controlar la dirección del conteo. Considera el RESET y el STOP, conectándolos a dos *Push-Buttons* de la tarjeta de desarrollo.

2. *(35 puntos)* Diseña un contador ascendente – descendente de 8 bits, que con ayuda de una variable externa asignada a un interruptor, de manera ascendente cuente sólo números pares y de manera descendente sólo números impares. Considera el RESET y el STOP, conectándolos a dos *Push-Buttons* de la tarjeta de desarrollo.

3. *(20 puntos)* Realiza un listado sencillo de las principales características de la tarjeta Spartan 3E.

Campo 5: **Conclusiones individuales.**