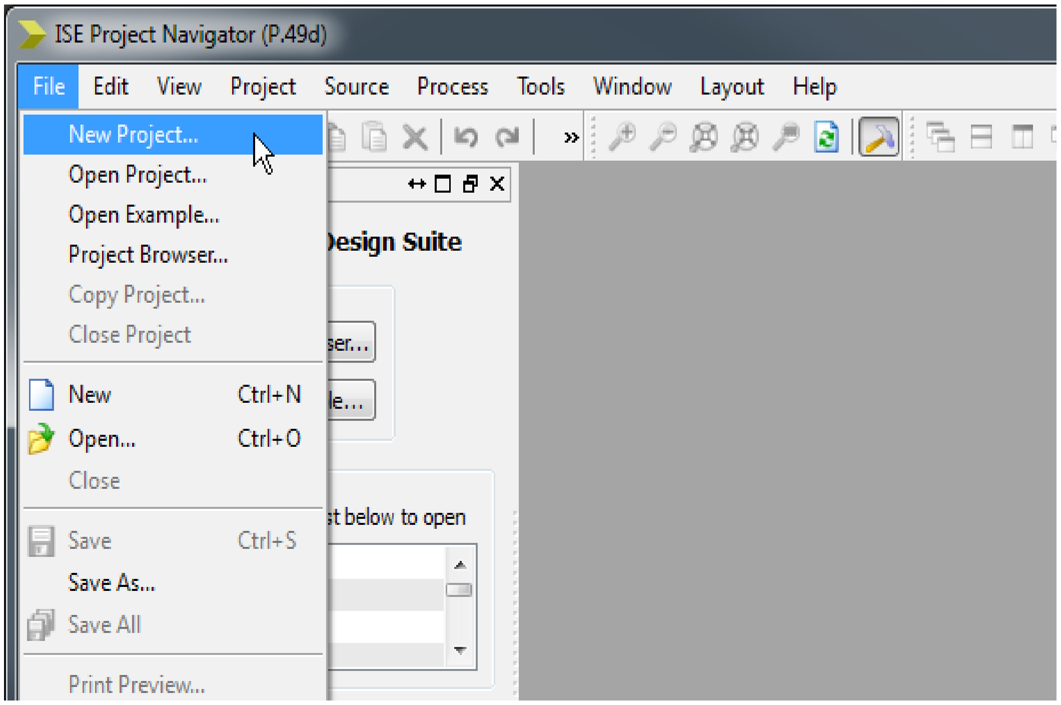
C.1 “Microblaze! Now What!?”

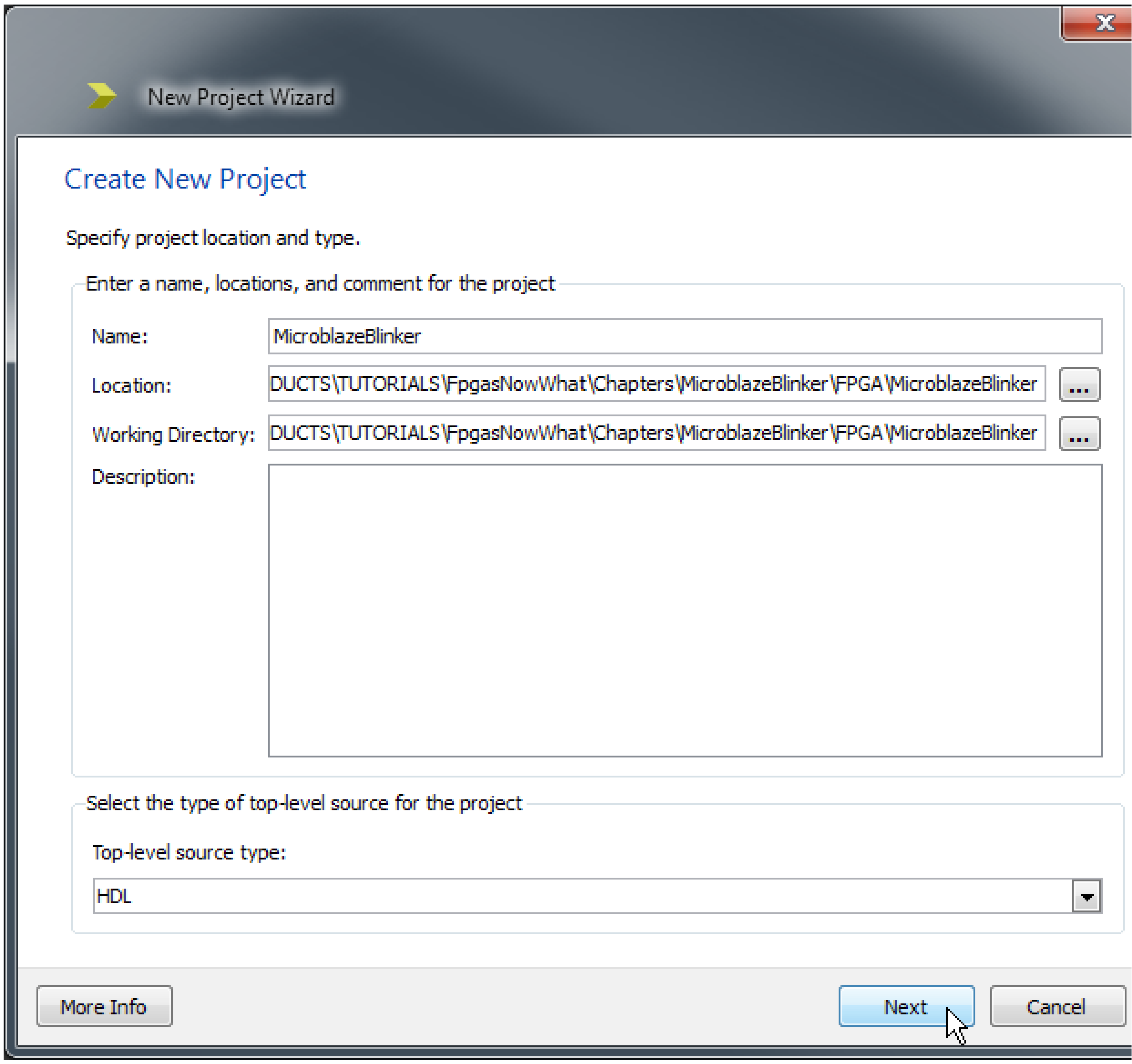
Start with the ISE

Start the Xilinx ISE tools. In this example, I'm using the ISE System Edition with a 30-day evaluation license so I can have full access to the Embedded Development Kit (EDK).

As usual, I begin by starting a new project:



I selected a directory to store my project and gave it a title. (I chose "MicroblazeBlinker".)



Create New Project

Specify project location and type.

Enter a name, locations, and comment for the project

Name: MicroblazeBlinker

Location: DUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker

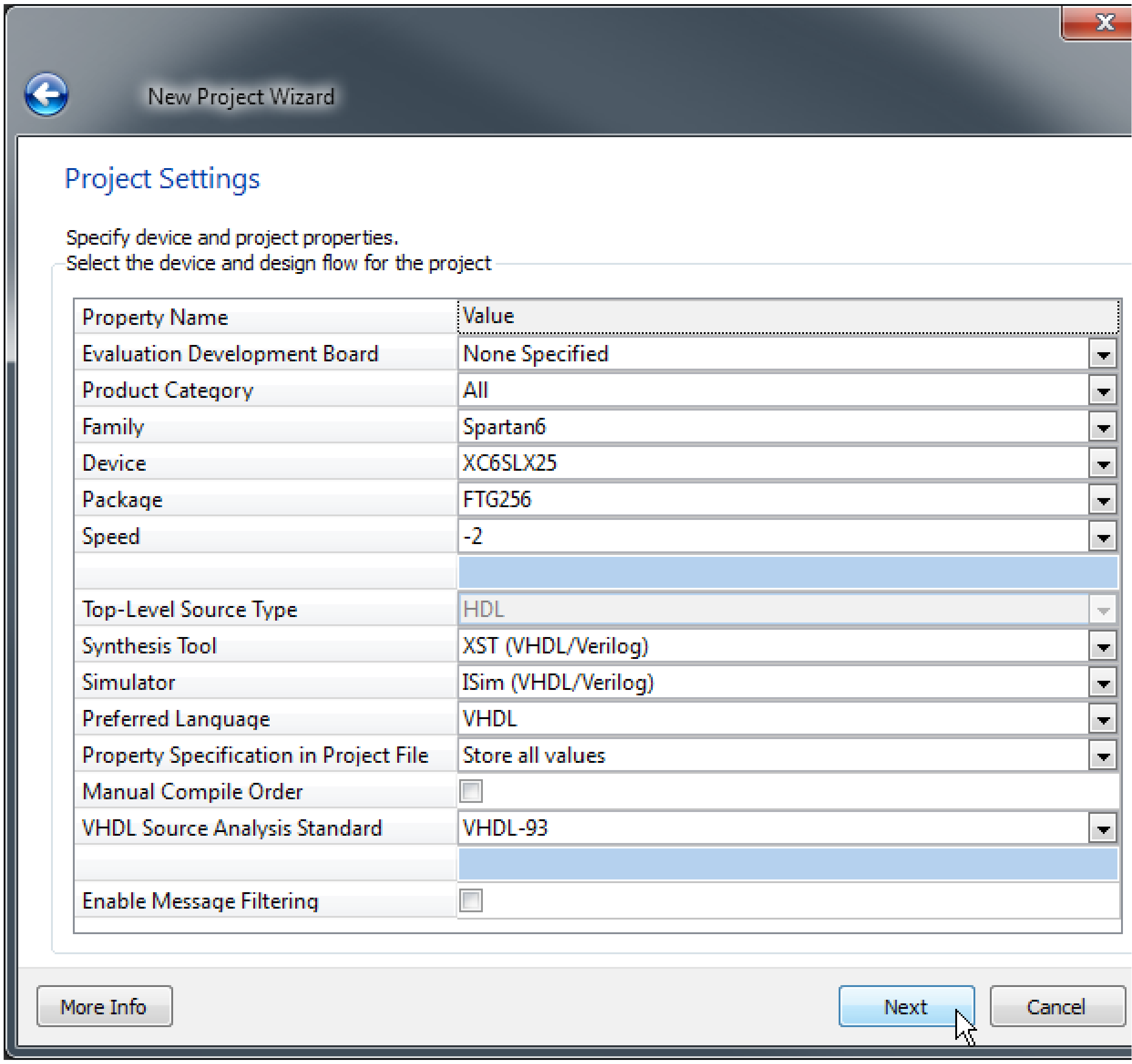
Working Directory: DUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker

Description:

Select the type of top-level source for the project

Top-level source type: HDL

Here are the project settings for a XuLA2 FPGA board:

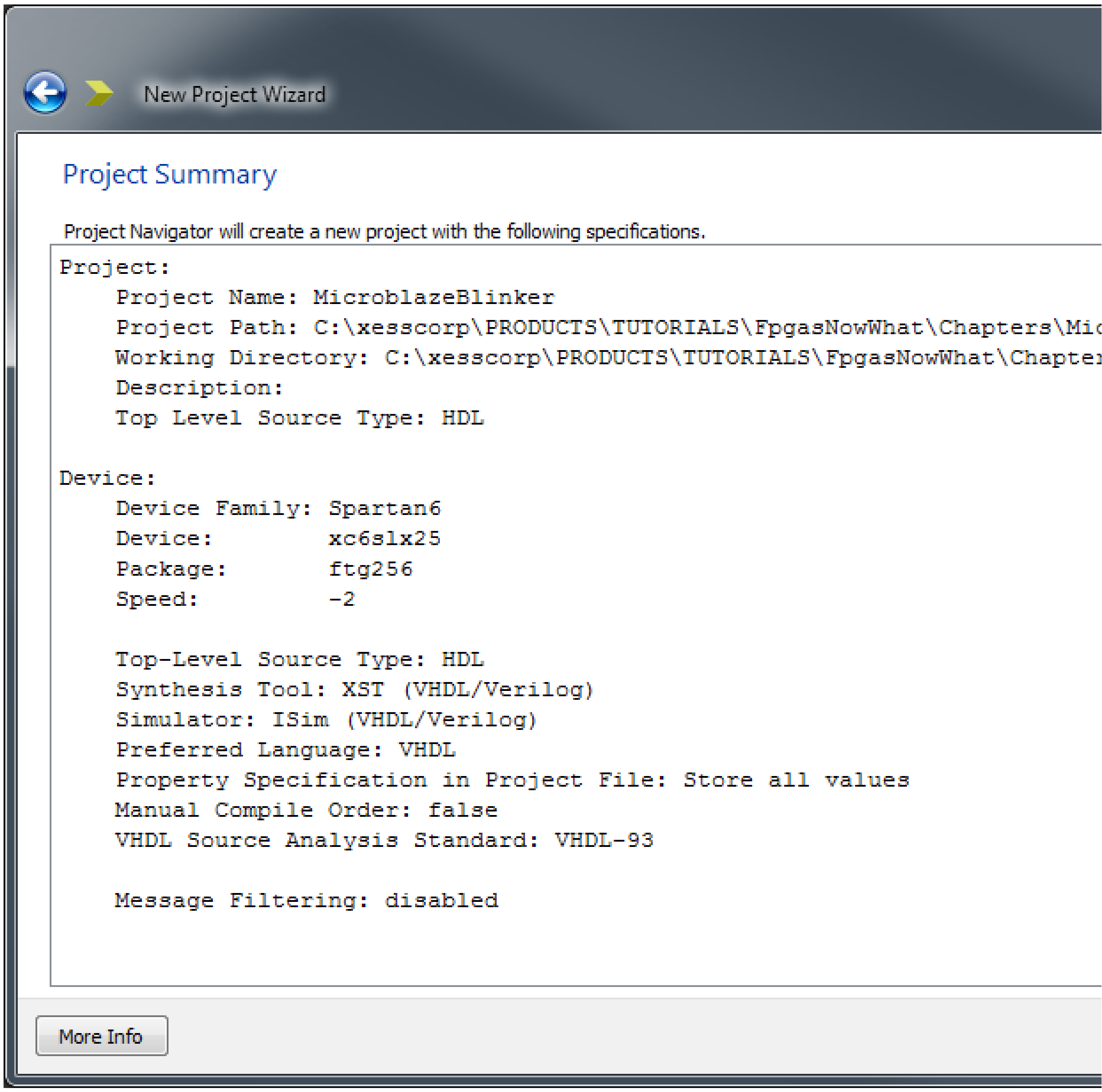


Project Settings

Specify device and project properties.
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan6
Device	XC6SLX25
Package	FTG256
Speed	-2
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

Here's the project summary after clicking the Next button:

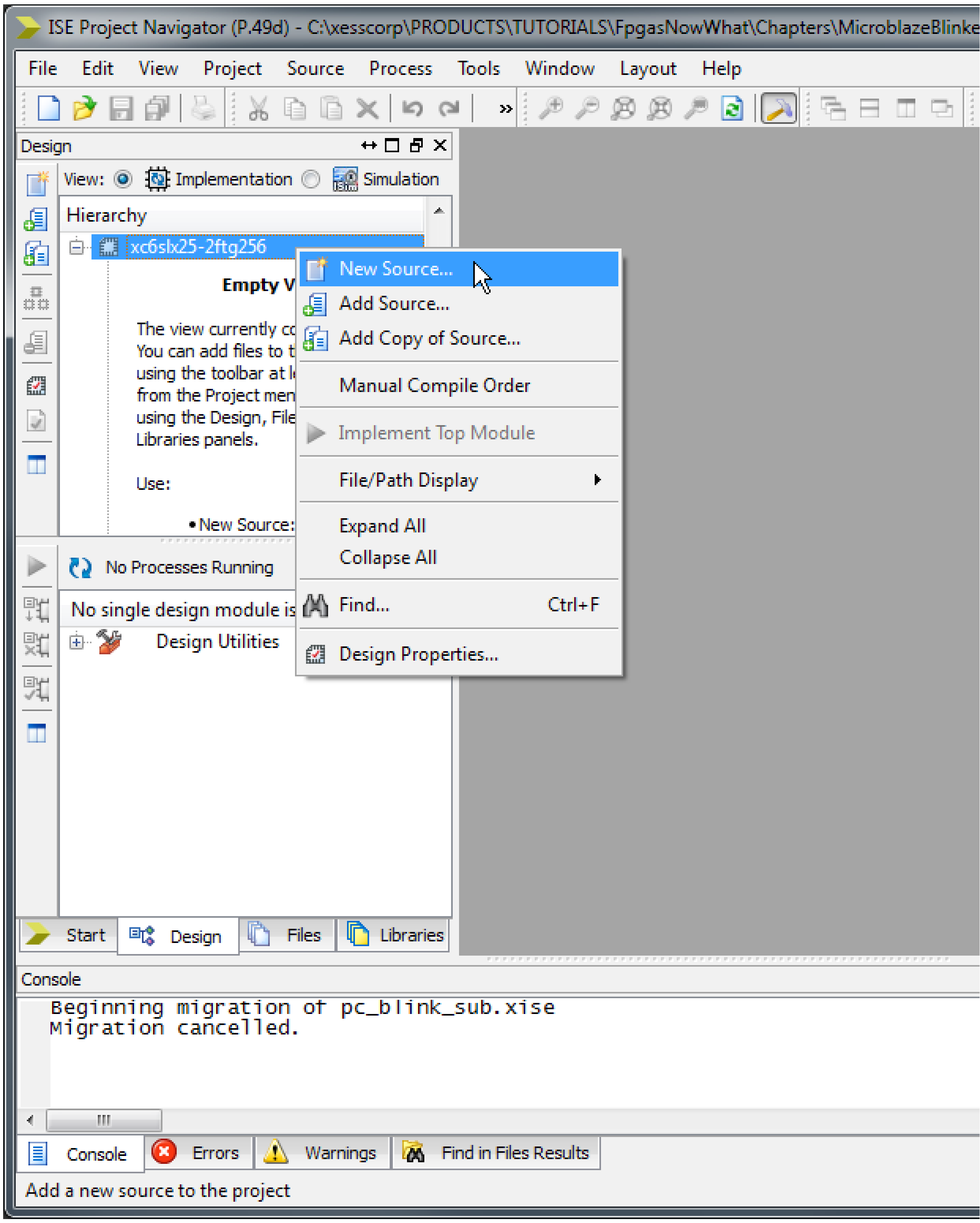


Project Summary

Project Navigator will create a new project with the following specifications.

```
Project:
    Project Name: MicroblazeBlinker
    Project Path: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\Mic
    Working Directory: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapter
    Description:
    Top Level Source Type: HDL

Device:
    Device Family: Spartan6
    Device:        xc6slx25
    Package:       ftg256
    Speed:         -2

    Top-Level Source Type: HDL
    Synthesis Tool: XST (VHDL/Verilog)
    Simulator: ISim (VHDL/Verilog)
    Preferred Language: VHDL
    Property Specification in Project File: Store all values
    Manual Compile Order: false
    VHDL Source Analysis Standard: VHDL-93

    Message Filtering: disabled
```

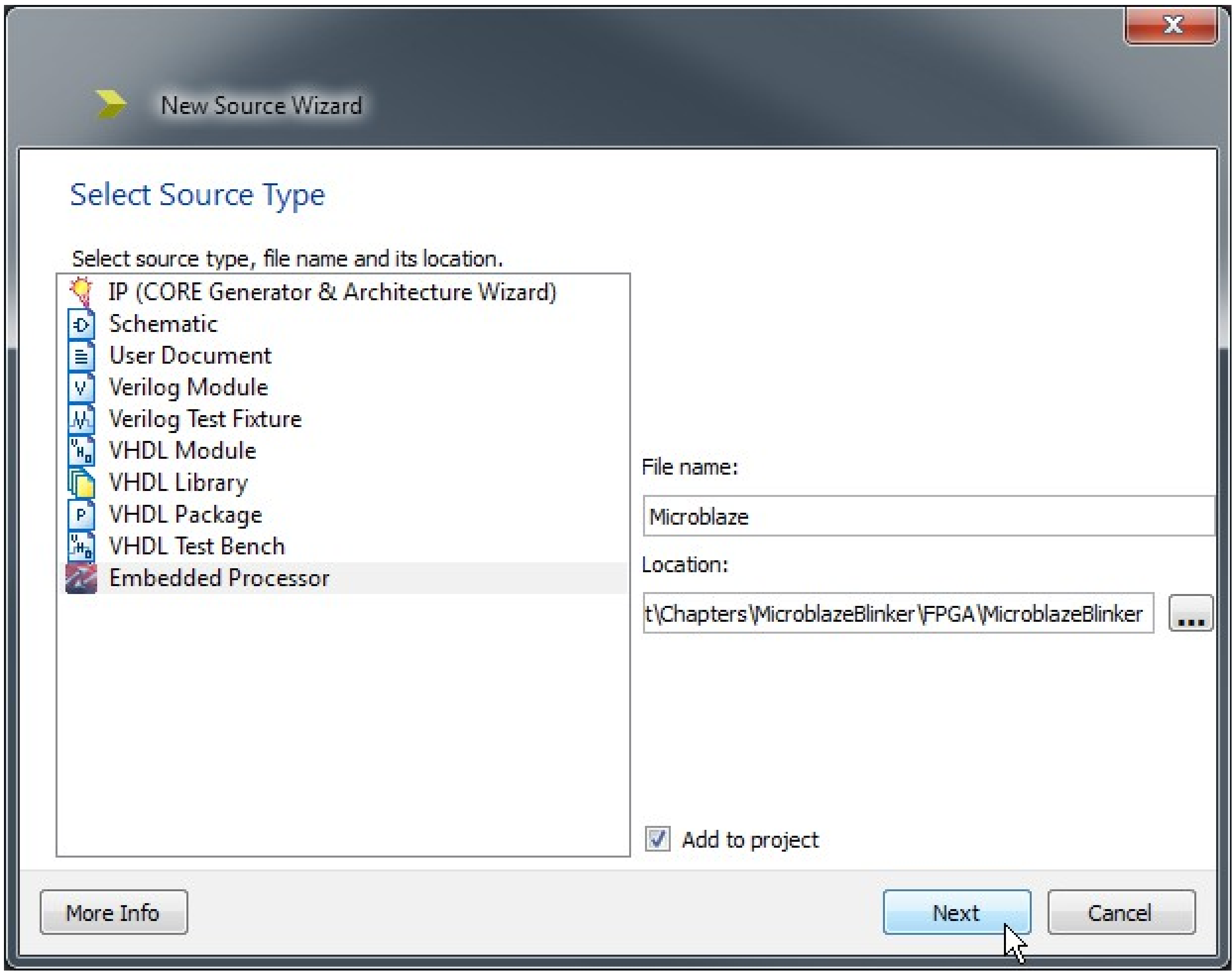
Now that the project is started, I'll add a Microblaze processor to it:



```
Beginning migration of pc_blink_sub.xise
Migration cancelled.
```

Add a new source to the project

I chose "Embedded Processor" and gave it a name. (I chose "Microblaze because I'm so damn creative.)



Select Source Type

Select source type, file name and its location.

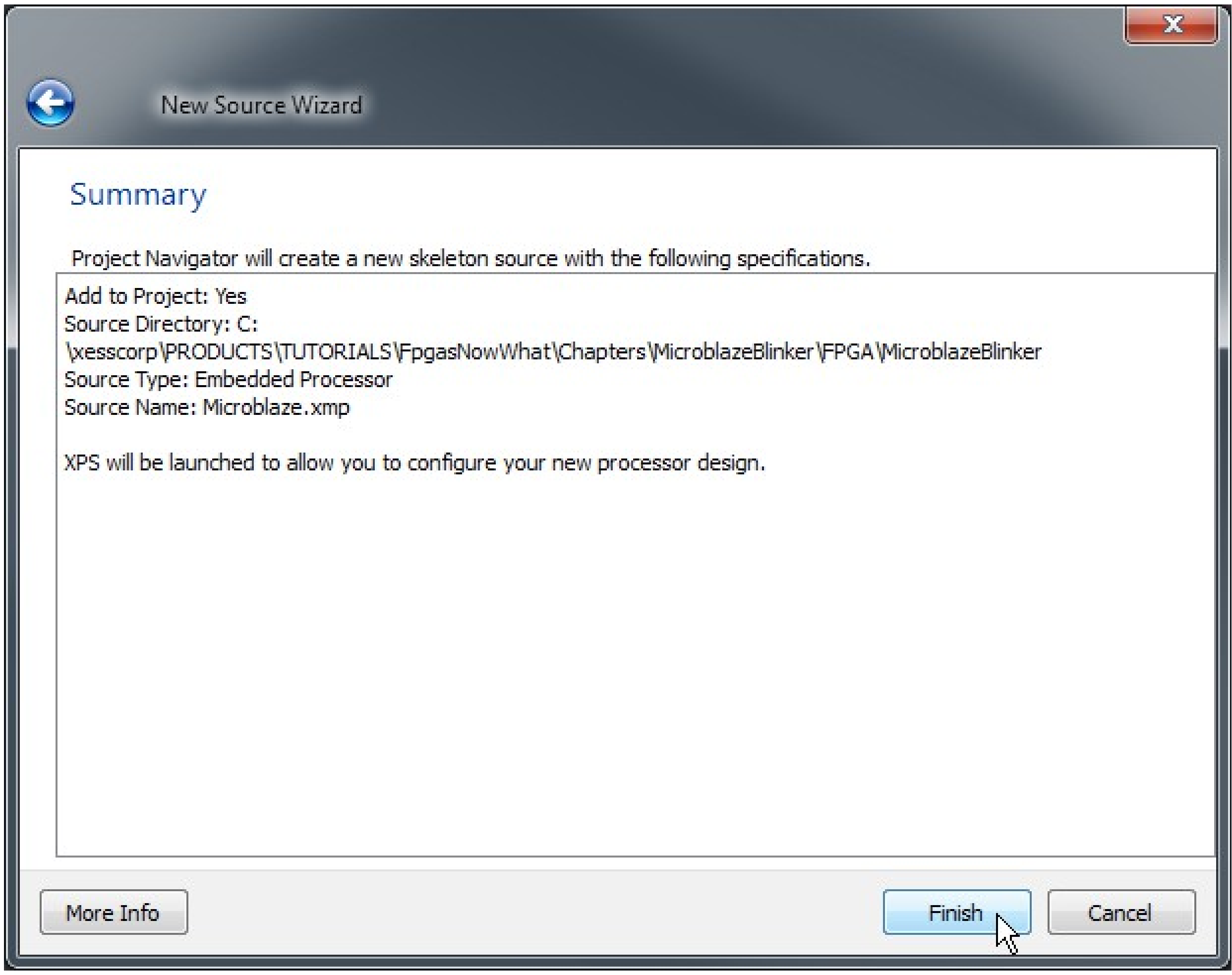
- IP (CORE Generator & Architecture Wizard)
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: Microblaze

Location: t\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker

[✓] Add to project

Here's the summary for the Microblaze processor that I added. That was simple! But it gets more complicated from here.



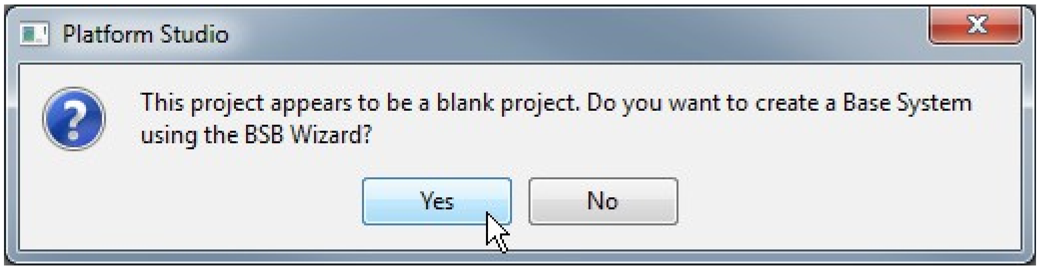
Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:
\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker
Source Type: Embedded Processor
Source Name: Microblaze.xmp

XPS will be launched to allow you to configure your new processor design.

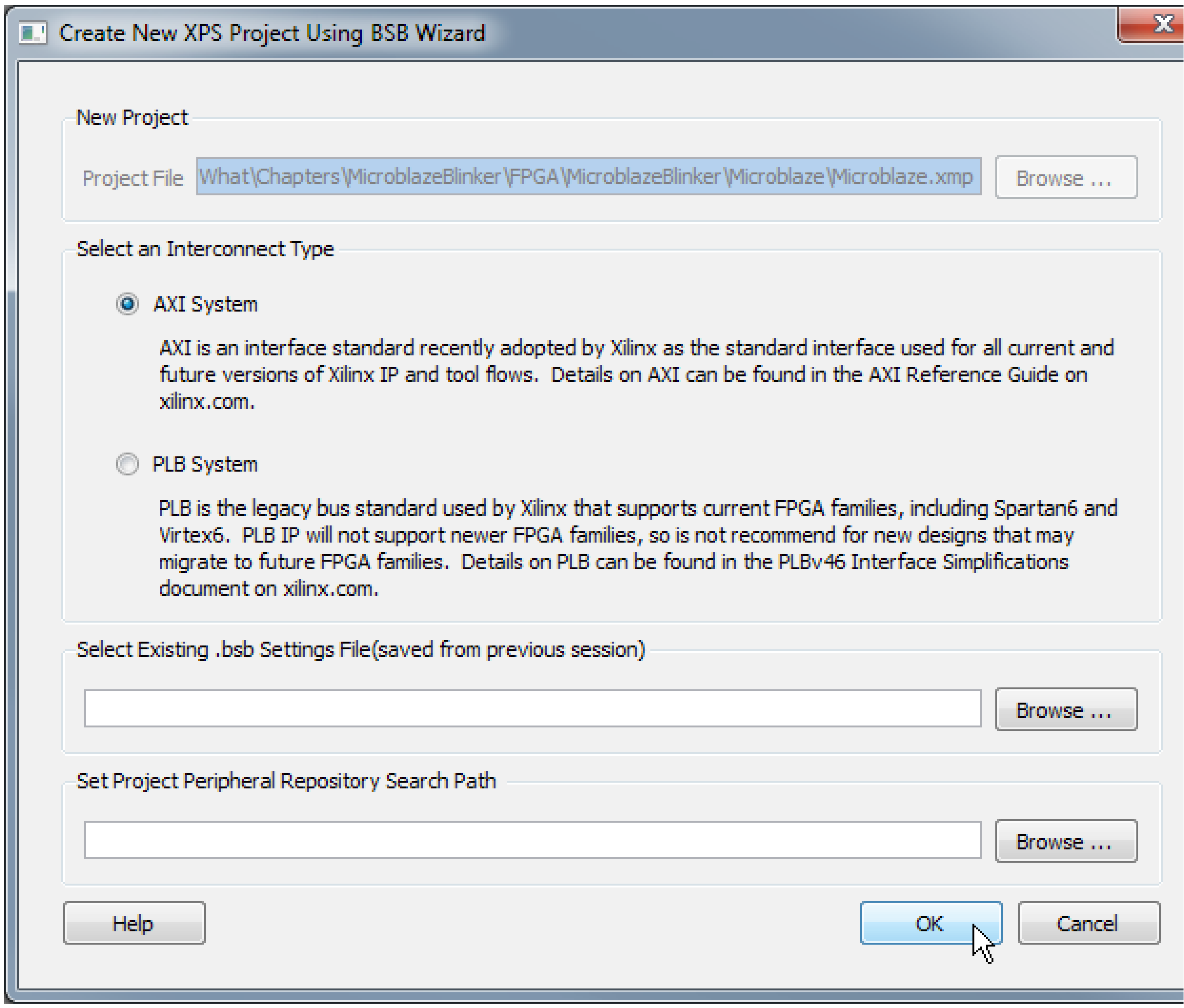
In the BSB

The tools notice the Microblaze processor hasn't been augmented with any program memory, data memory or peripherals so it offers to start the Base System Builder (BSB) for me. Thanks, ISE, you're so nice!



This project appears to be a blank project. Do you want to create a Base System using the BSB Wizard?

The default settings in this dialog were cool, so I just clicked OK and moved on:



New Project

Project File: What\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker\Microblaze\Microblaze.xmp

Select an Interconnect Type

[✓ AXI System]

AXI is an interface standard recently adopted by Xilinx as the standard interface used for all current and future versions of Xilinx IP and tool flows. Details on AXI can be found in the AXI Reference Guide on xilinx.com.

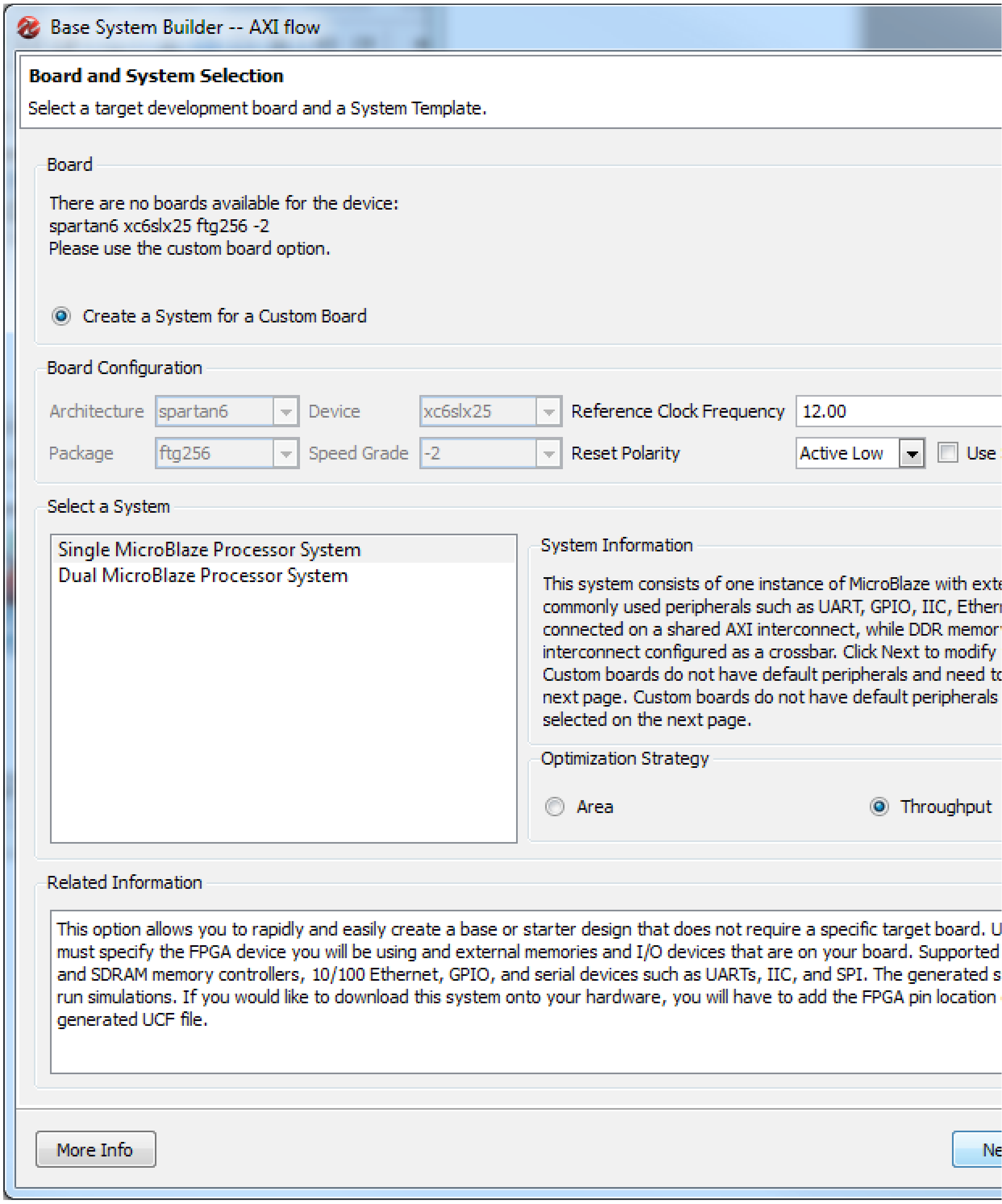
PLB System

PLB is the legacy bus standard used by Xilinx that supports current FPGA families, including Spartan6 and Virtex6. PLB IP will not support newer FPGA families, so is not recommend for new designs that may migrate to future FPGA families. Details on PLB can be found in the PLBv46 Interface Simplifications document on xilinx.com.

Select Existing .bsb Settings File(saved from previous session)

Set Project Peripheral Repository Search Path

The XuLA2 isn't one of Xilinx's anointed boards, so I needed to create a custom board. I set the reference clock to 12 MHz because that's what's on the XuLA2 board. Other than that, I left everything else as it was. (Take note that the reset for the Microblaze is active low; I'll need that later on.)


Base System Builder -- AXI flow

Board and System Selection

Select a target development board and a System Template.

Board

There are no boards available for the device:
spartan6 xc6slx25 ftg256 -2
Please use the custom board option.

[✓ Create a System for a Custom Board]

Board Configuration

Architecture	spartan6	Device	xc6slx25	Reference Clock Frequency	12.00
Package	ftg256	Speed Grade	-2	Reset Polarity	Active Low

Select a System

Single MicroBlaze Processor System
Dual MicroBlaze Processor System

System Information

This system consists of one instance of MicroBlaze with exte
commonly used peripherals such as UART, GPIO, IIC, Ether
connected on a shared AXI interconnect, while DDR memor
interconnect configured as a crossbar. Click Next to modify
Custom boards do not have default peripherals and need t
next page. Custom boards do not have default peripherals
selected on the next page.

Optimization Strategy

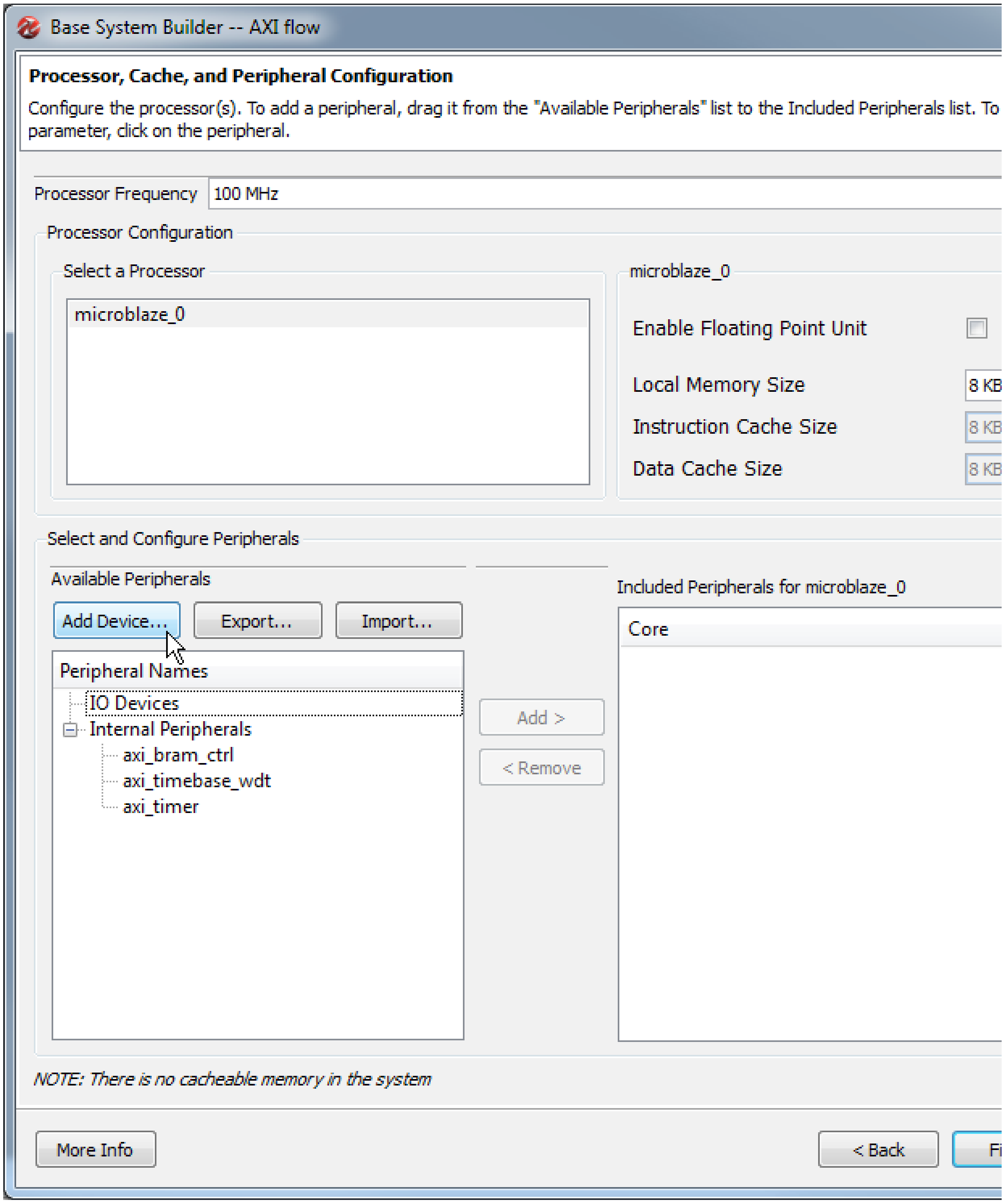
Area [✓ Throughput]

Related Information

This option allows you to rapidly and easily create a base or starter design that does not require a specific target board. U
must specify the FPGA device you will be using and external memories and I/O devices that are on your board. Supported
and SDRAM memory controllers, 10/100 Ethernet, GPIO, and serial devices such as UARTs, IIC, and SPI. The generated s
run simulations. If you would like to download this system onto your hardware, you will have to add the FPGA pin location
generated UCF file.

I left the top portion of the next screen as it was with the single Microblaze core set to run at 100 MHz with 8 KB of memory. (I'm just flashing an LED! How much memory can that take?)

Since I'm flashing an LED, I needed a single output pin that the processor could drive high and low. So I click on the "Add Device..." button to begin that process.



Processor, Cache, and Peripheral Configuration

Configure the processor(s). To add a peripheral, drag it from the "Available Peripherals" list to the Included Peripherals list. To parameter, click on the peripheral.

Processor Frequency 100 MHz

Processor Configuration

Select a Processor: microblaze_0

microblaze_0

Enable Floating Point Unit	☐
Local Memory Size	8 KB
Instruction Cache Size	8 KB
Data Cache Size	8 KB

Select and Configure Peripherals

Available Peripherals

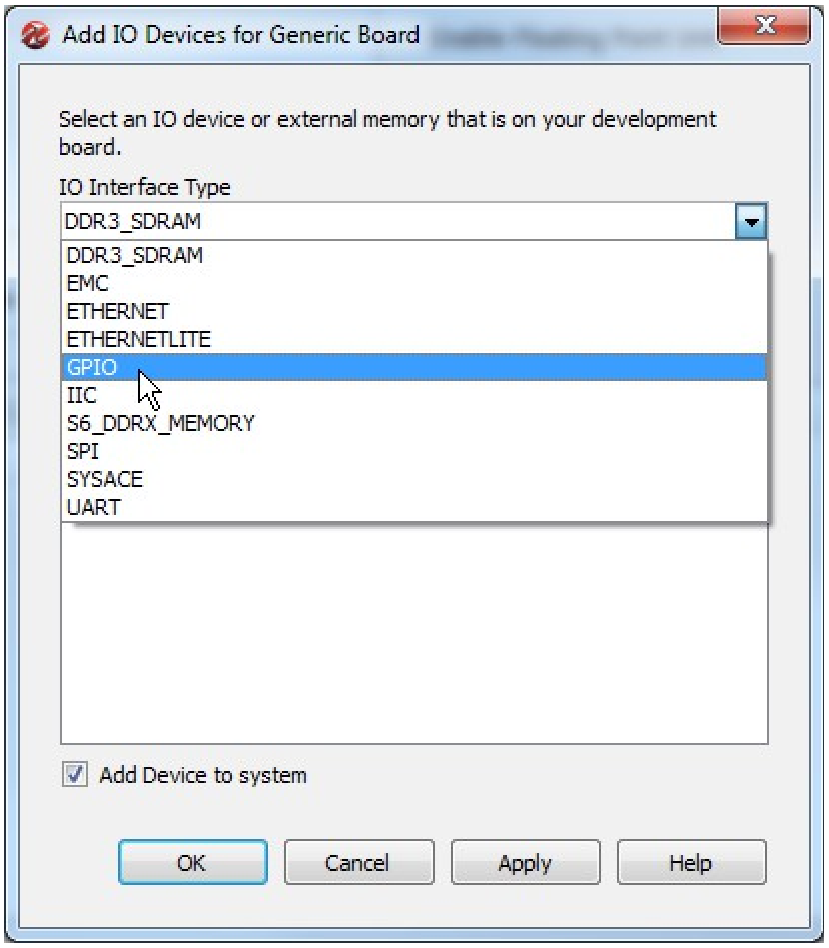
Peripheral Names
- IO Devices
- Internal Peripherals
  - axi_bram_ctrl
  - axi_timebase_wdt
  - axi_timer

Included Peripherals for microblaze_0

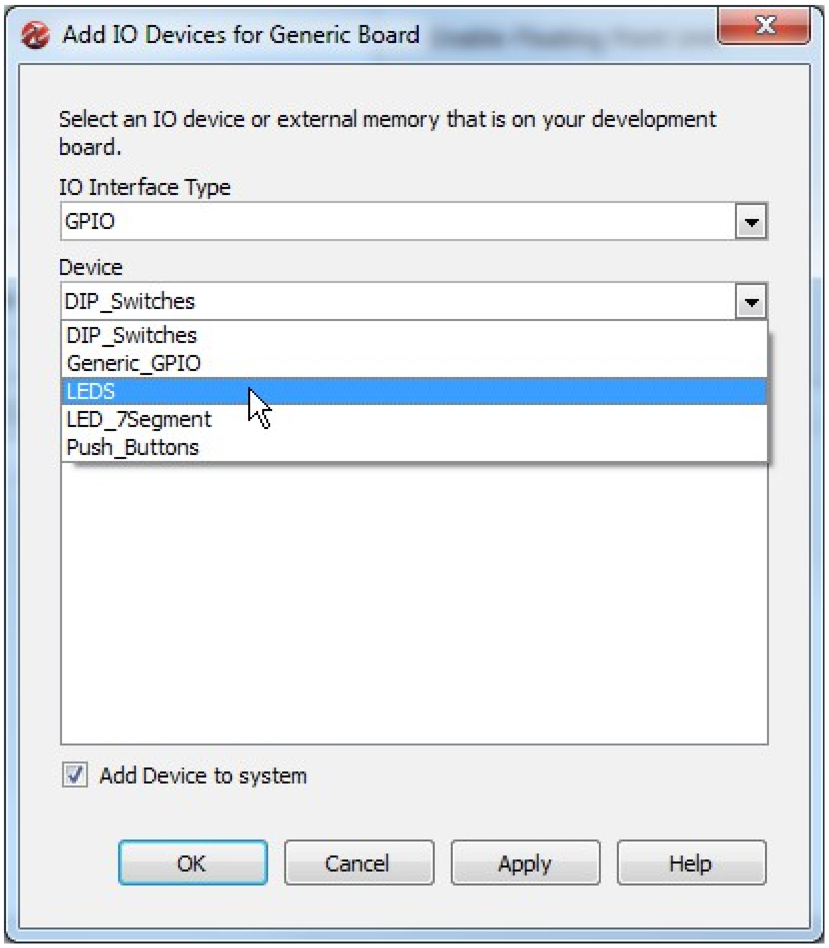
Core

NOTE: There is no cacheable memory in the system

I select "GPIO" from the list of I/O types.



Then I select an LED driver as a specific type of I/O:

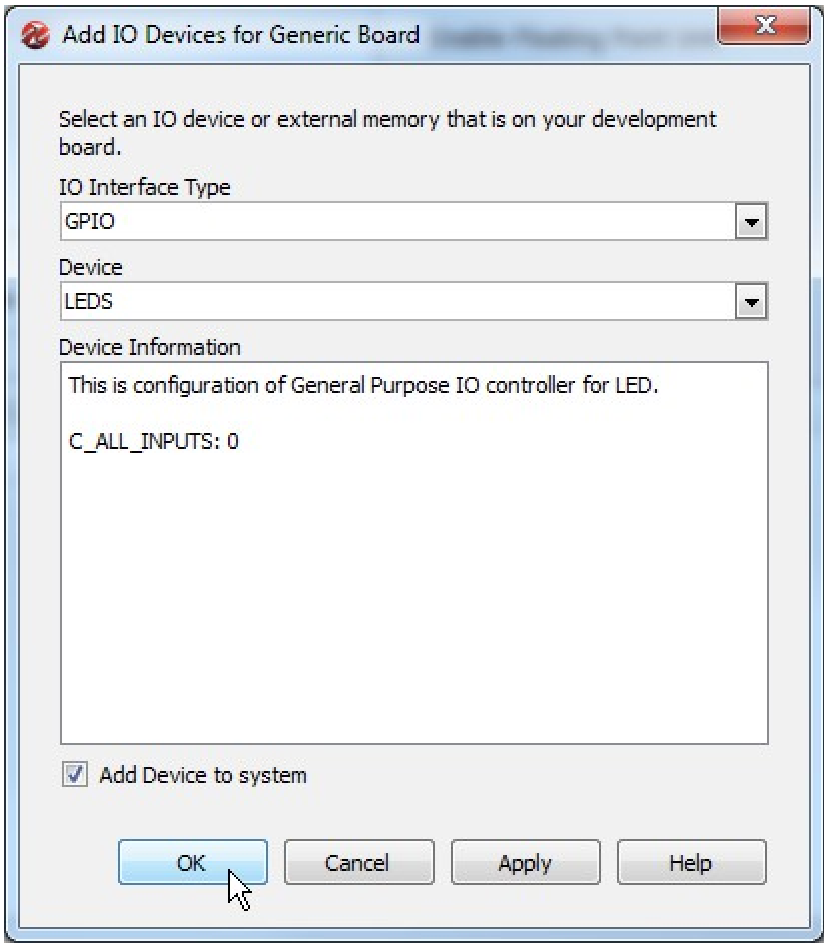

Add IO Devices for Generic Board

Select an IO device or external memory that is on your development board.

IO Interface Type: GPIO

Device: DIP_Switches

After that, I clicked the OK button to instantiate the GPIO.



Select an IO device or external memory that is on your development board.

IO Interface Type

GPIO

Device

LEDS

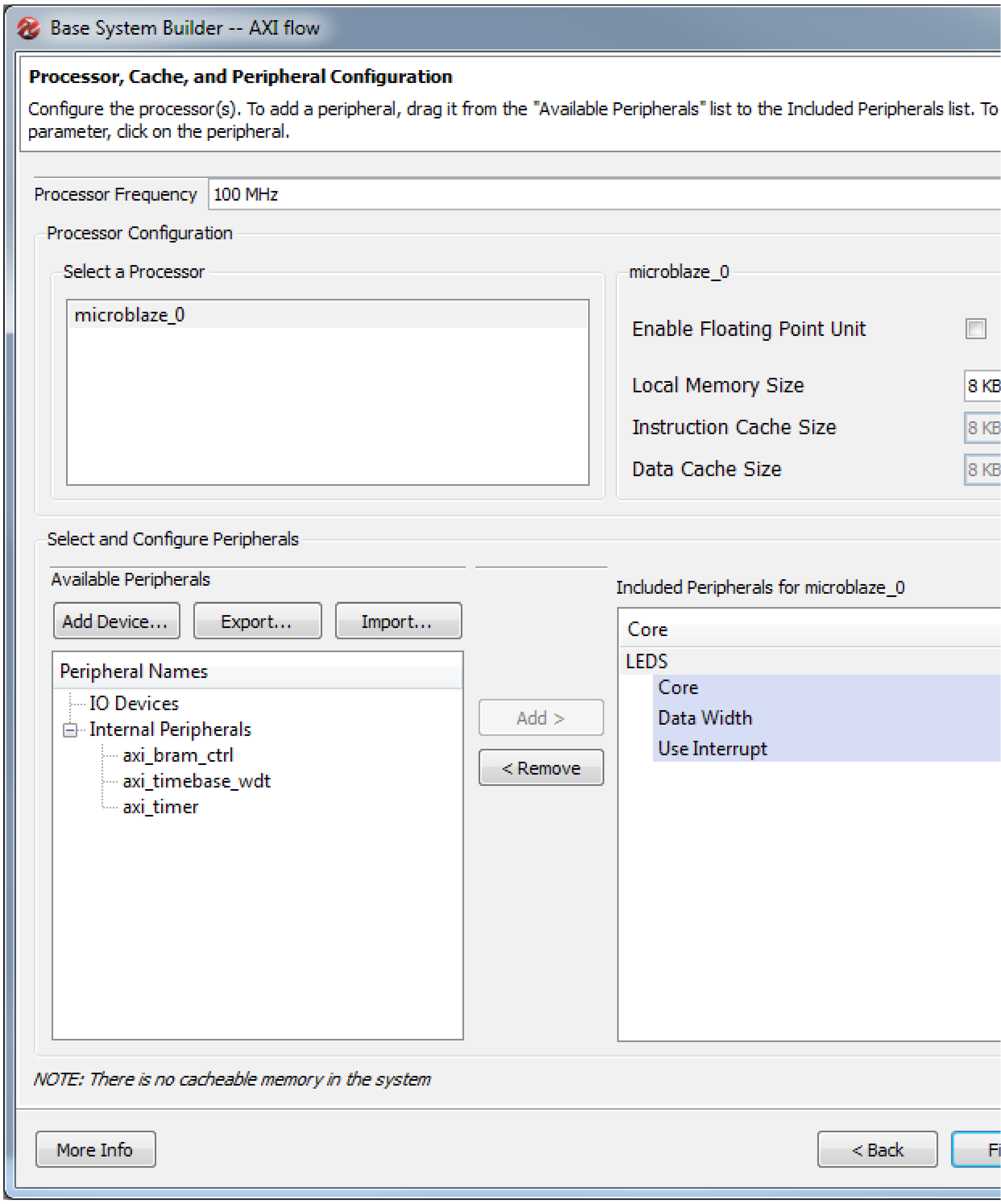
Device Information

This is configuration of General Purpose IO controller for LED.

C_ALL_INPUTS: 0

[✓] Add Device to system

The LED driver was added to the "Included Peripherals" pane. I clicked on the "Data Width" field and set it to a single bit since I'll only be blinking a single LED.



Processor, Cache, and Peripheral Configuration

Configure the processor(s). To add a peripheral, drag it from the "Available Peripherals" list to the Included Peripherals list. To
parameter, click on the peripheral.

Processor Frequency 100 MHz

Processor Configuration

Select a Processor: microblaze_0

microblaze_0

Enable Floating Point Unit ☐

Local Memory Size 8 KB

Instruction Cache Size 8 KB

Data Cache Size 8 KB

Select and Configure Peripherals

Available Peripherals

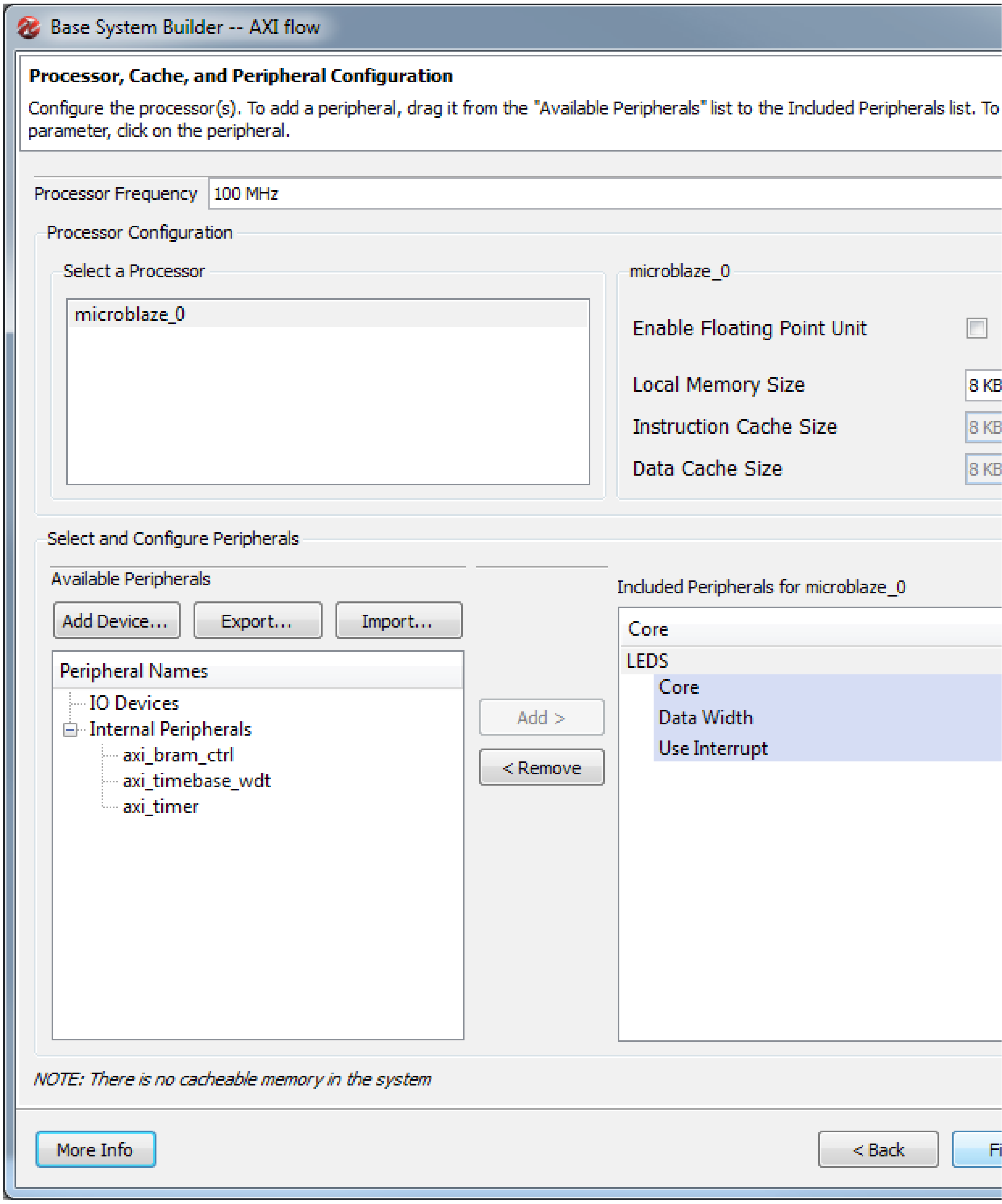
Peripheral Names
- IO Devices
- Internal Peripherals
  - axi_bram_ctrl
  - axi_timebase_wdt
  - axi_timer

Included Peripherals for microblaze_0

Core
- LEDS
  - Core
  - Data Width
  - Use Interrupt

NOTE: There is no cacheable memory in the system

That was the only peripheral I needed, so I clicked Finish.



Base System Builder -- AXI flow

Processor, Cache, and Peripheral Configuration

Configure the processor(s). To add a peripheral, drag it from the "Available Peripherals" list to the Included Peripherals list. To
parameter, click on the peripheral.

Processor Frequency: 100 MHz

Processor Configuration

Select a Processor: microblaze_0

microblaze_0

Enable Floating Point Unit	☐
Local Memory Size	8 KB
Instruction Cache Size	8 KB
Data Cache Size	8 KB

Select and Configure Peripherals

Available Peripherals

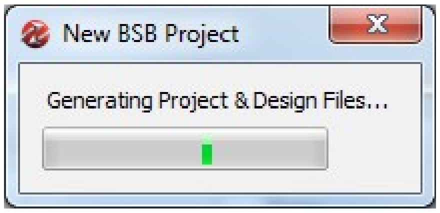
- IO Devices
- Internal Peripherals
  - axi_bram_ctrl
  - axi_timebase_wdt
  - axi_timer

Included Peripherals for microblaze_0

- Core
- LEDS
  - Core
  - Data Width
  - Use Interrupt

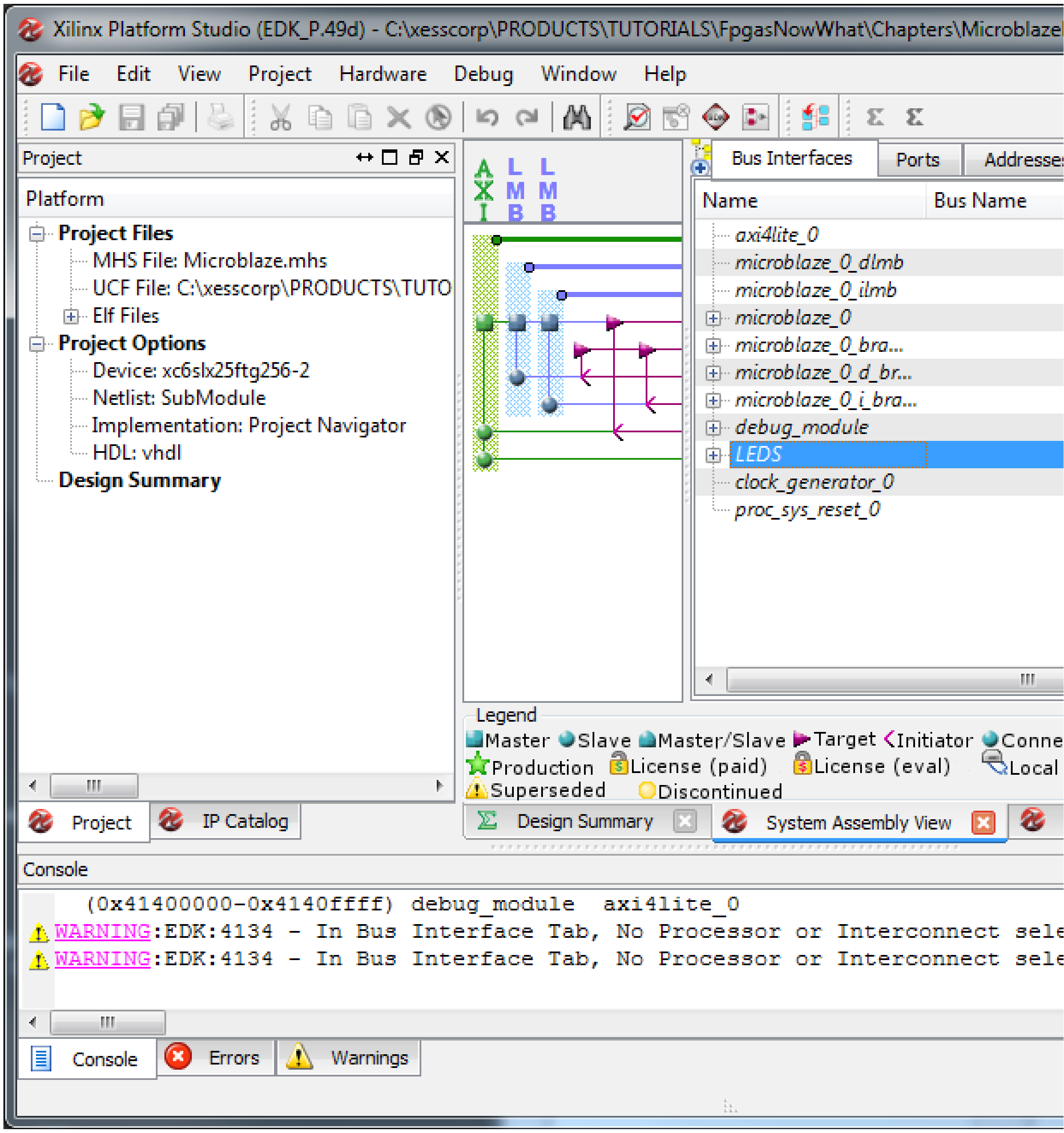
NOTE: There is no cacheable memory in the system

Now the BSB generates all the necessary file for the processor core, memory and peripherals.



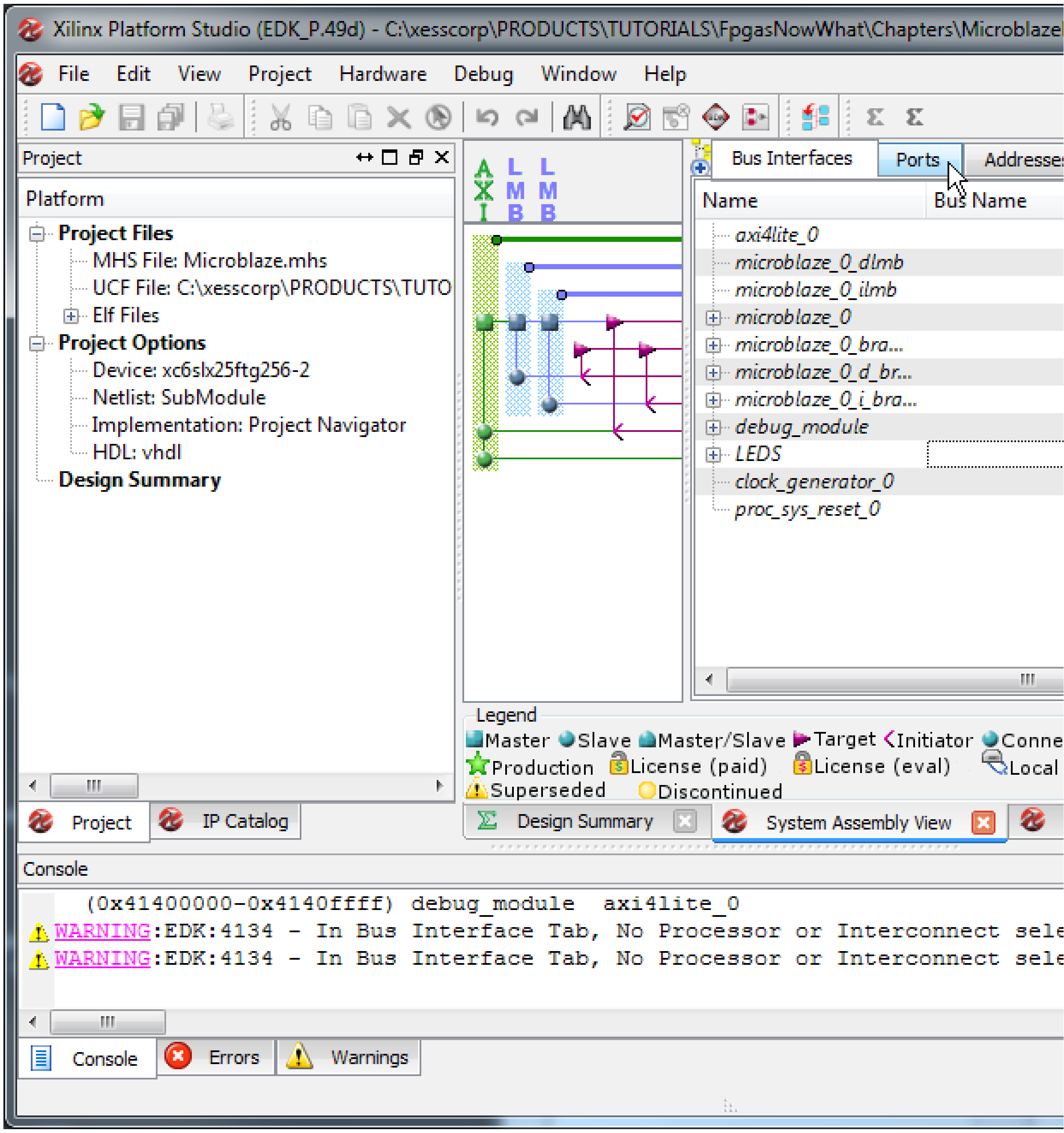
Over to the XPS

After the BSB is finished, it automatically starts the Xilinx Platform Studio (XPS) where I can finish configuring the Microblaze. (Thanks again, ISE!)



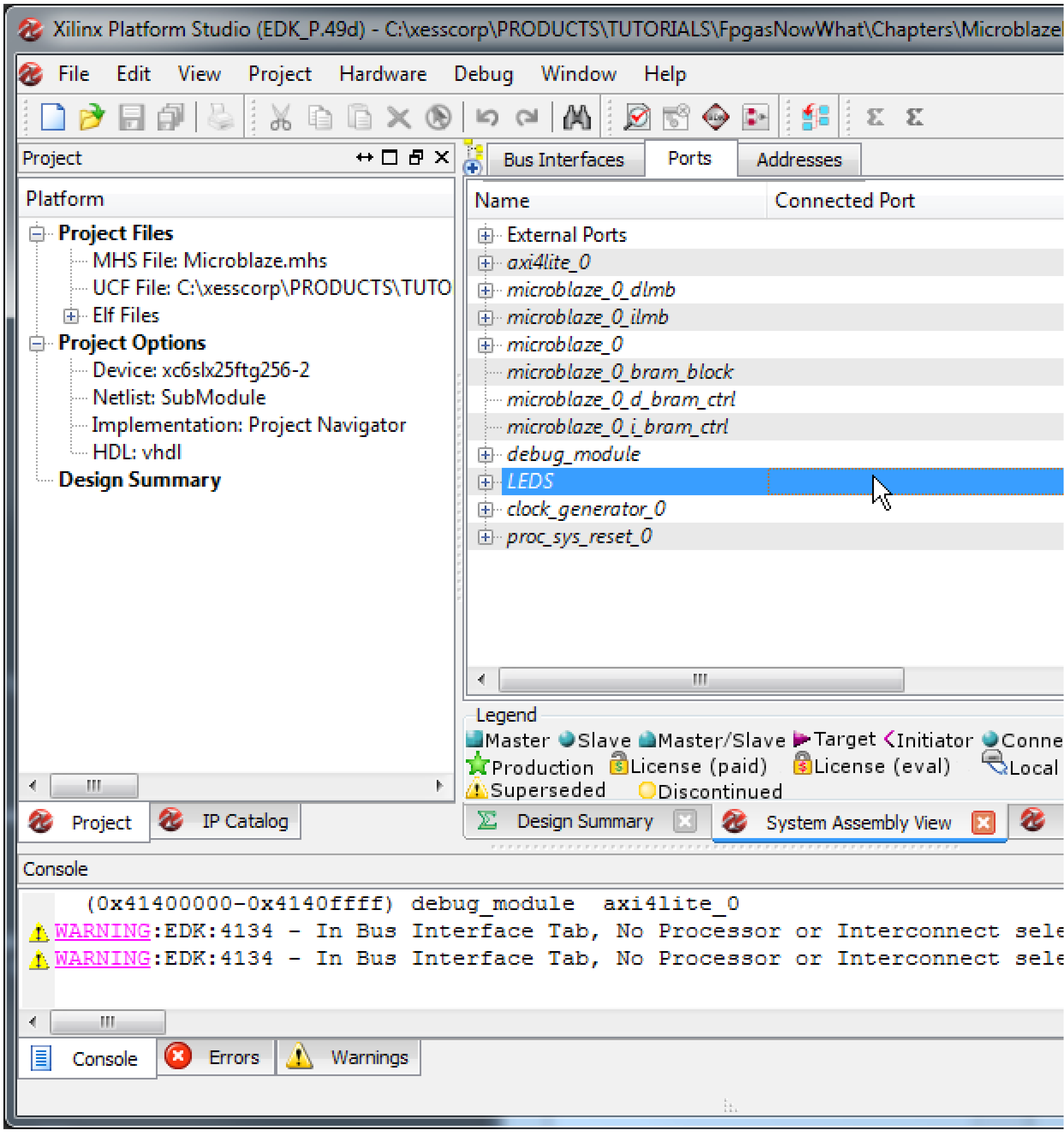
```
   (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

What I need to do is get the LED driver connected to an external pin and attach the external clock signal to the processor. To do this, I switch to the "Ports" tab:



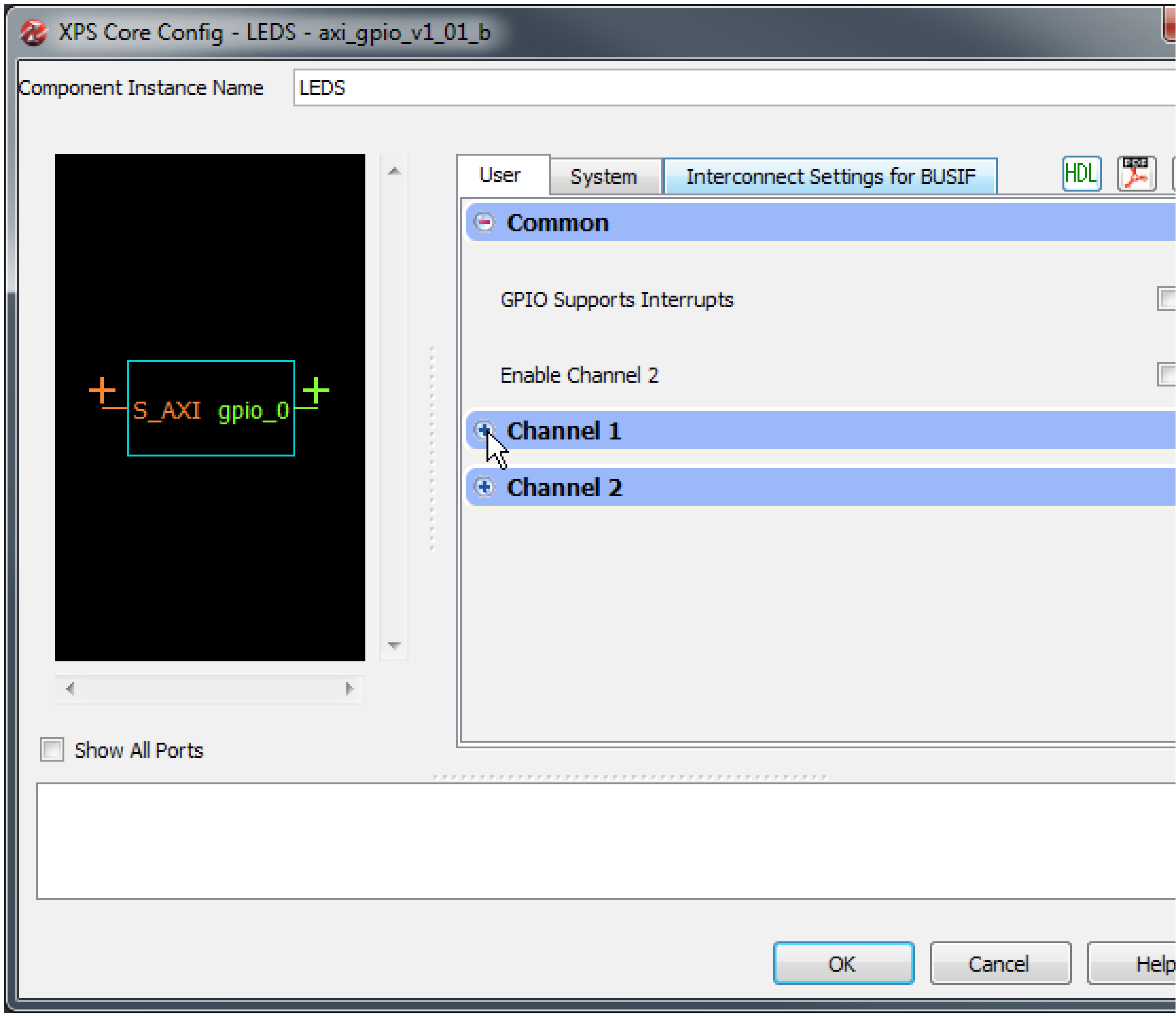
```
   (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

In the Ports tab, I double-click the LEDS device to bring up its configuration screen:



```
   (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

By default, the LED driver I/O doesn't generate interrupts to the processor. It also provides only a single I/O channel. There's no need to change these.


XPS Core Config - LEDS - axi_gpio_v1_01_b
User
System
Interconnect Settings for BUSIF
HDL
S_AXI
gpio_0
Show All Ports
OK
Cancel
Help

Component Instance Name: LEDS

Common

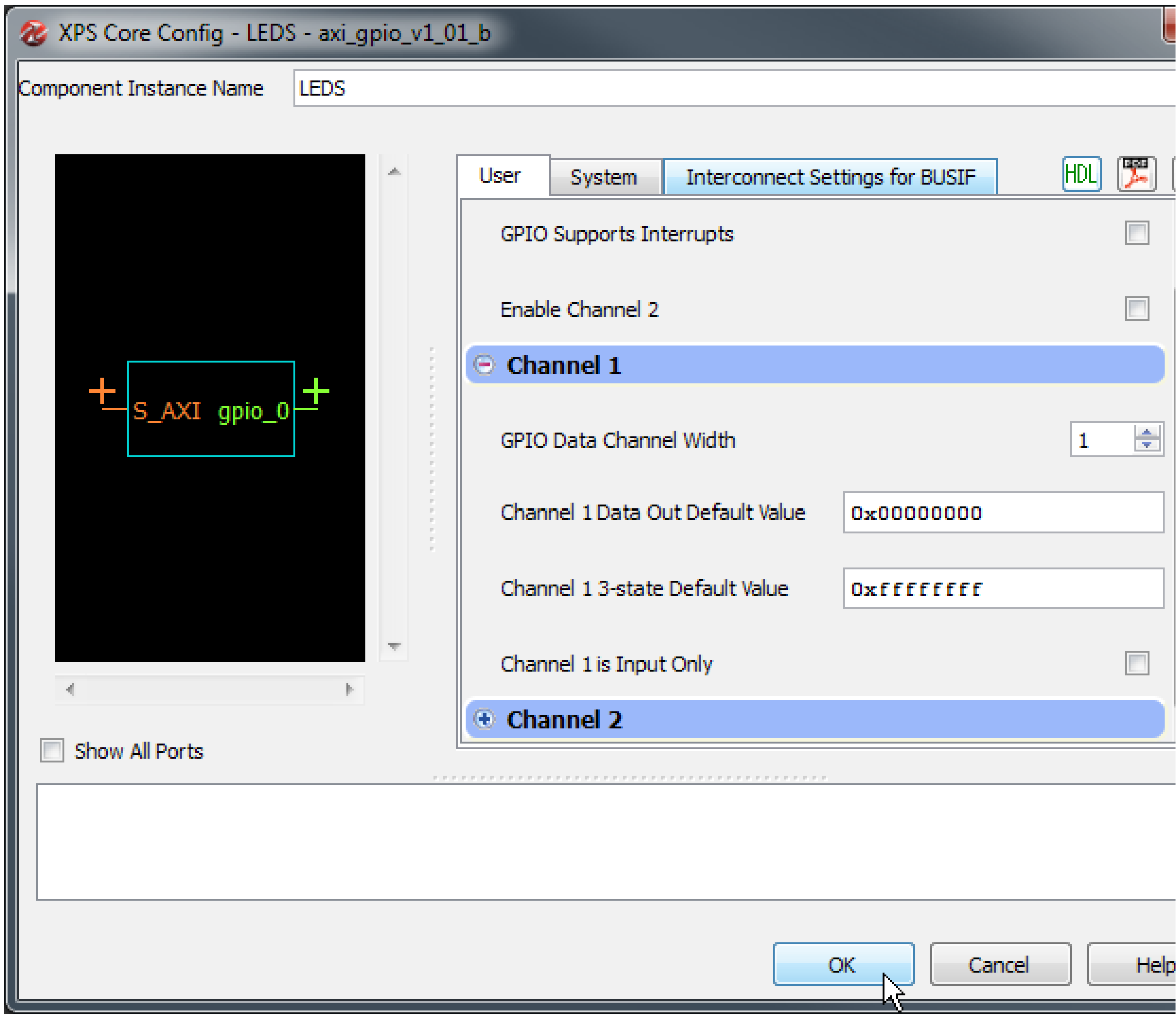
GPIO Supports Interrupts

Enable Channel 2

Channel 1

Channel 2

I clicked on the "+" symbol of Channel 1 so I could see its configuration. It's a single bit wide as I specified it in the BSB. That's all I needed to see, so I clicked on OK to close the configuration screen.


XPS Core Config - LEDS - axi_gpio_v1_01_b

Component Instance Name: LEDS

GPIO Supports Interrupts ☐

Enable Channel 2 ☐

Channel 1

GPIO Data Channel Width: 1

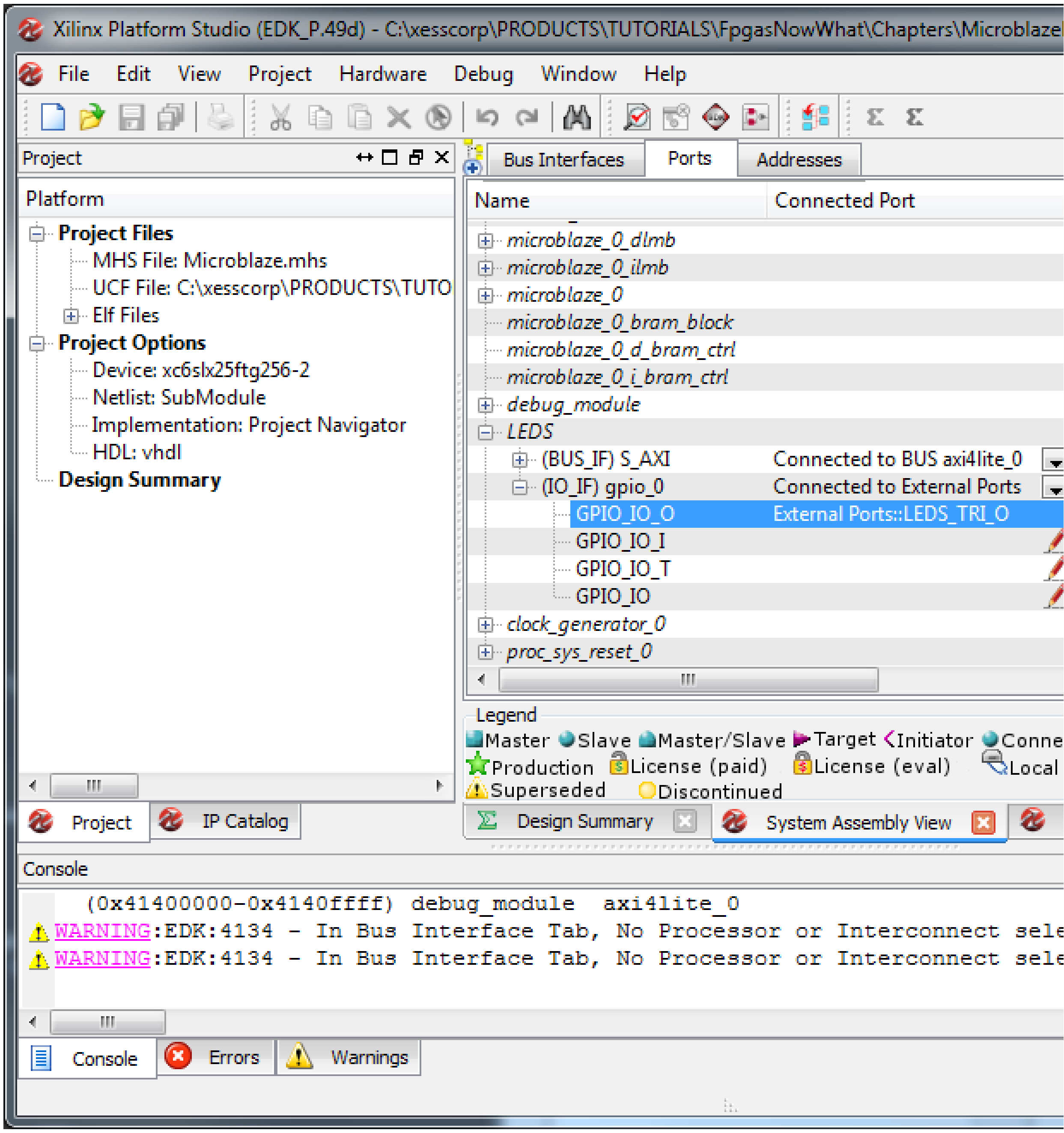
Channel 1 Data Out Default Value: 0x00000000

Channel 1 3-state Default Value: 0xffffffff

Channel 1 is Input Only ☐

Channel 2

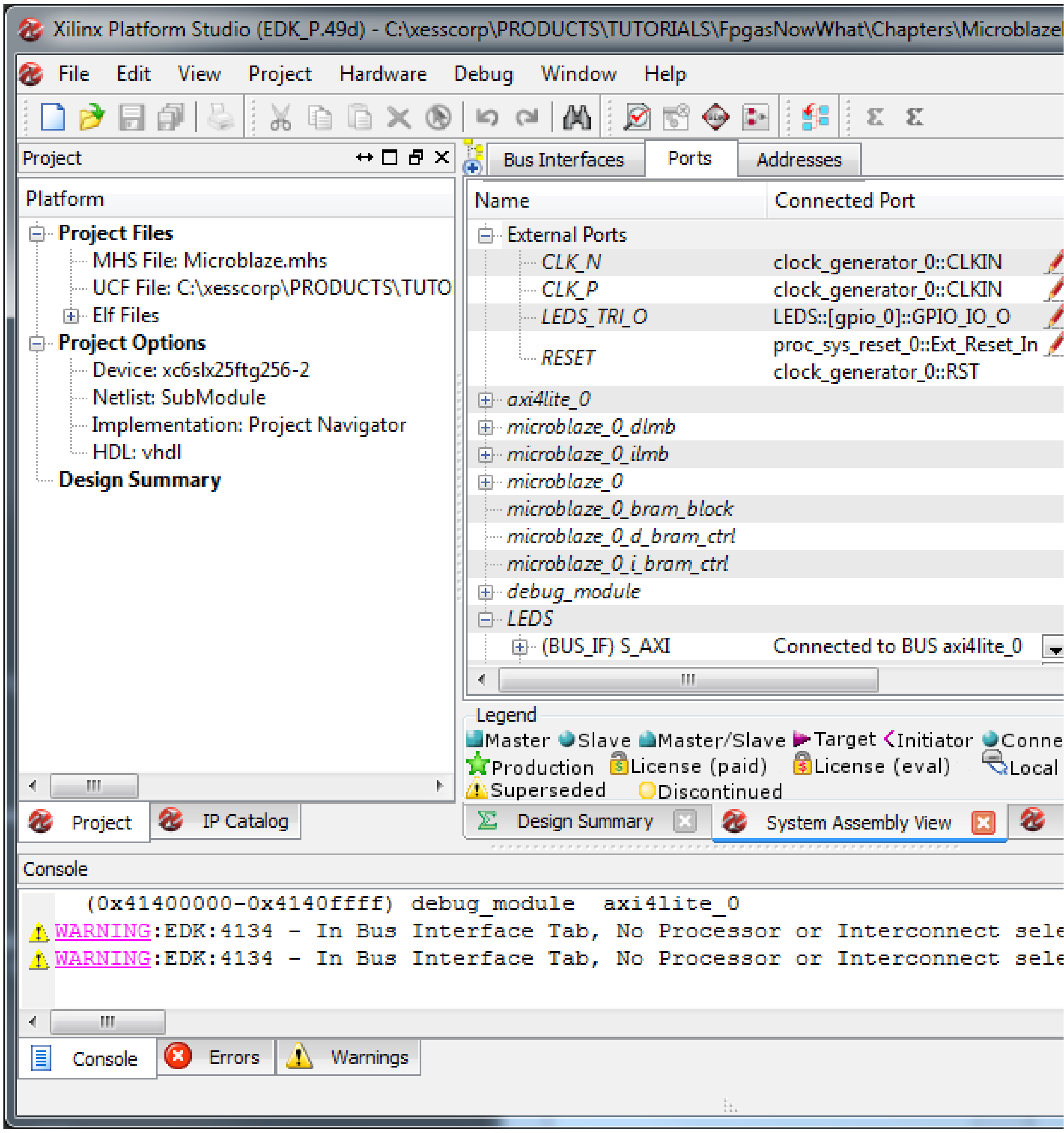
Returning to the XPS main screen, I expanded the LEDS entry in the Ports tab. Further expanding the I/O interface (IO_IF) shows there is a single output attached to an external port.


Xilinx Platform Studio (EDK_P.49d) - C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\Microblaze
File Edit View Project Hardware Debug Window Help
Project
Platform
Project Files
MHS File: Microblaze.mhs
UCF File: C:\xesscorp\PRODUCTS\TUTO
Elf Files
Project Options
Device: xc6slx25ftg256-2
Netlist: SubModule
Implementation: Project Navigator
HDL: vhdl
Design Summary
Bus Interfaces
Ports
Addresses
Legend
Master Slave Master/Slave Target Initiator Conne
Production License (paid) License (eval) Local
Superseded Discontinued
Project
IP Catalog
Design Summary
System Assembly View
Console
Errors
Warnings

Name	Connected Port
microblaze_0_dlmb	
microblaze_0_ilmb	
microblaze_0	
microblaze_0_bram_block	
microblaze_0_d_bram_ctrl	
microblaze_0_i_bram_ctrl	
debug_module	
LEDS	
(BUS_IF) S_AXI	Connected to BUS axi4lite_0
(IO_IF) gpio_0	Connected to External Ports
GPIO_IO_O	External Ports::LEDS_TRI_O
GPIO_IO_I	
GPIO_IO_T	
GPIO_IO	
clock_generator_0	
proc_sys_reset_0	

```
    (0x41400000-0x4140ffff) debug_module   axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

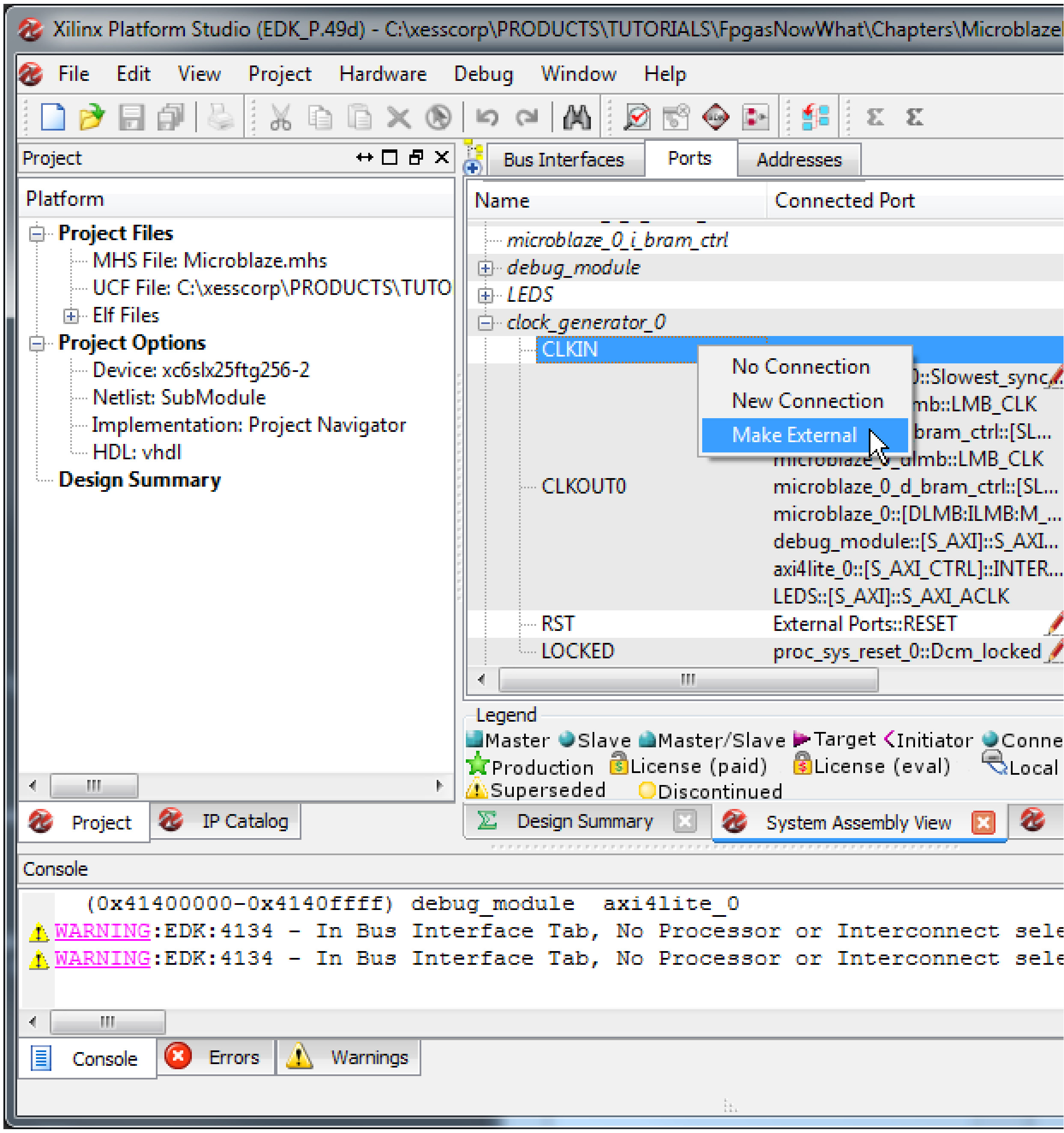
Expanding the External Ports entry shows a one-bit-wide bus (LEDS_TRI_O) that's connected to the LED GPIO output. That's what I expected, so everything's good so far.



Name	Connected Port
External Ports	
CLK_N	clock_generator_0::CLKIN
CLK_P	clock_generator_0::CLKIN
LEDS_TRI_O	LEDS::[gpio_0]::GPIO_IO_O
RESET	proc_sys_reset_0::Ext_Reset_In clock_generator_0::RST
axi4lite_0	
microblaze_0_dlmb	
microblaze_0_ilmb	
microblaze_0	
microblaze_0_bram_block	
microblaze_0_d_bram_ctrl	
microblaze_0_i_bram_ctrl	
debug_module	
LEDS	
(BUS_IF) S_AXI	Connected to BUS axi4lite_0

```
   (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

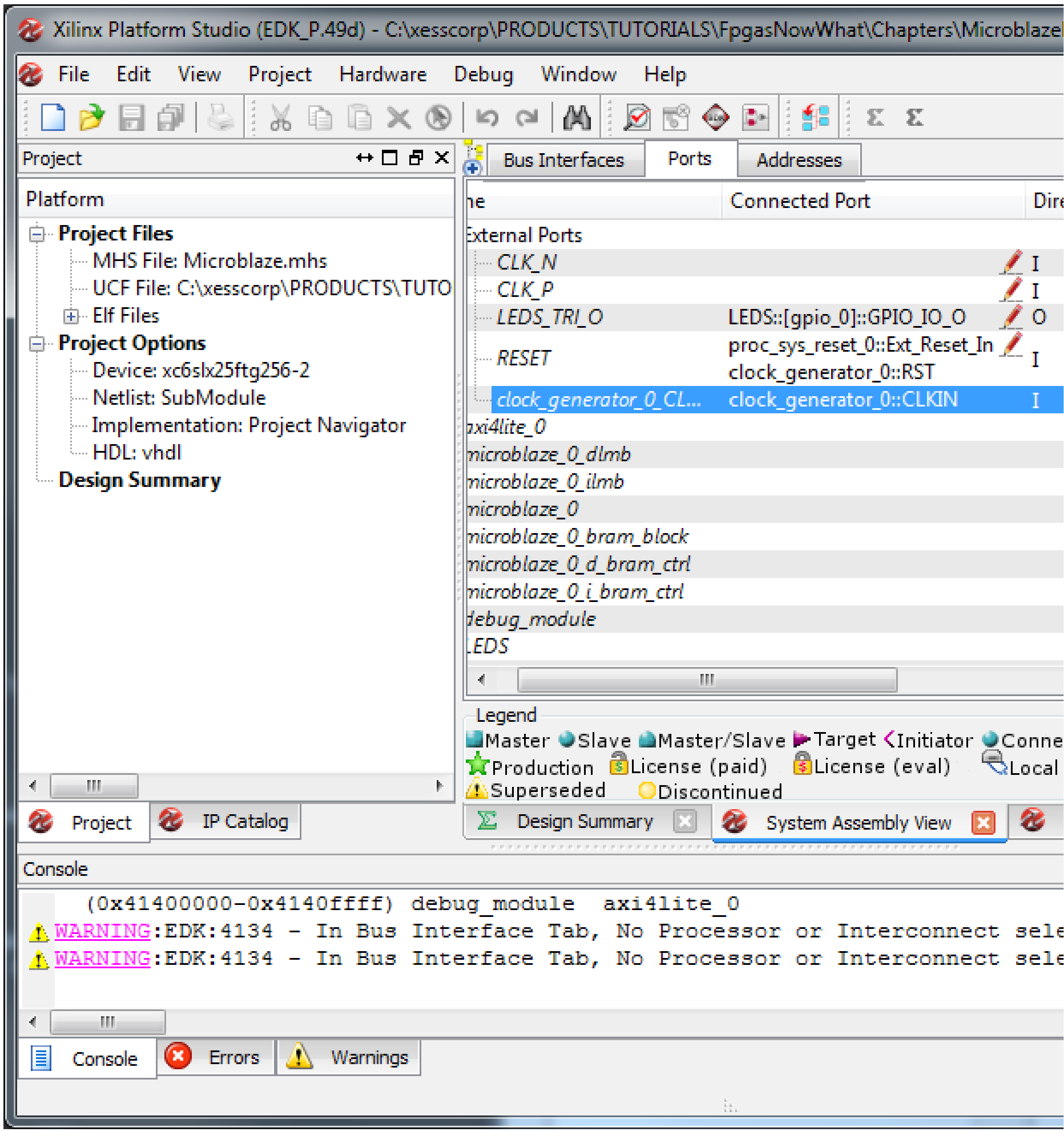
Next, I want to attach an external clock to the input of the clock generator. (The clock generator will create the 100 MHz system clock from the 12 MHz clock on the XuLA2 board.) Expanding the clock generator entry and right-clicking on the CLKIN connection, I select "Make External" to connect this input to an external pin.



Name	Connected Port
CLKIN	::Slowest_sync
	mb::LMB_CLK
	bram_ctrl::[SL...
	microblaze_0_dlmb::LMB_CLK
CLKOUT0	microblaze_0_d_bram_ctrl::[SL...
	microblaze_0::[DLMB:ILMB:M_...
	debug_module::[S_AXI]::S_AXI...
	axi4lite_0::[S_AXI_CTRL]::INTER...
	LEDS::[S_AXI]::S_AXI_ACLK
RST	External Ports::RESET
LOCKED	proc_sys_reset_0::Dcm_locked

```
    (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

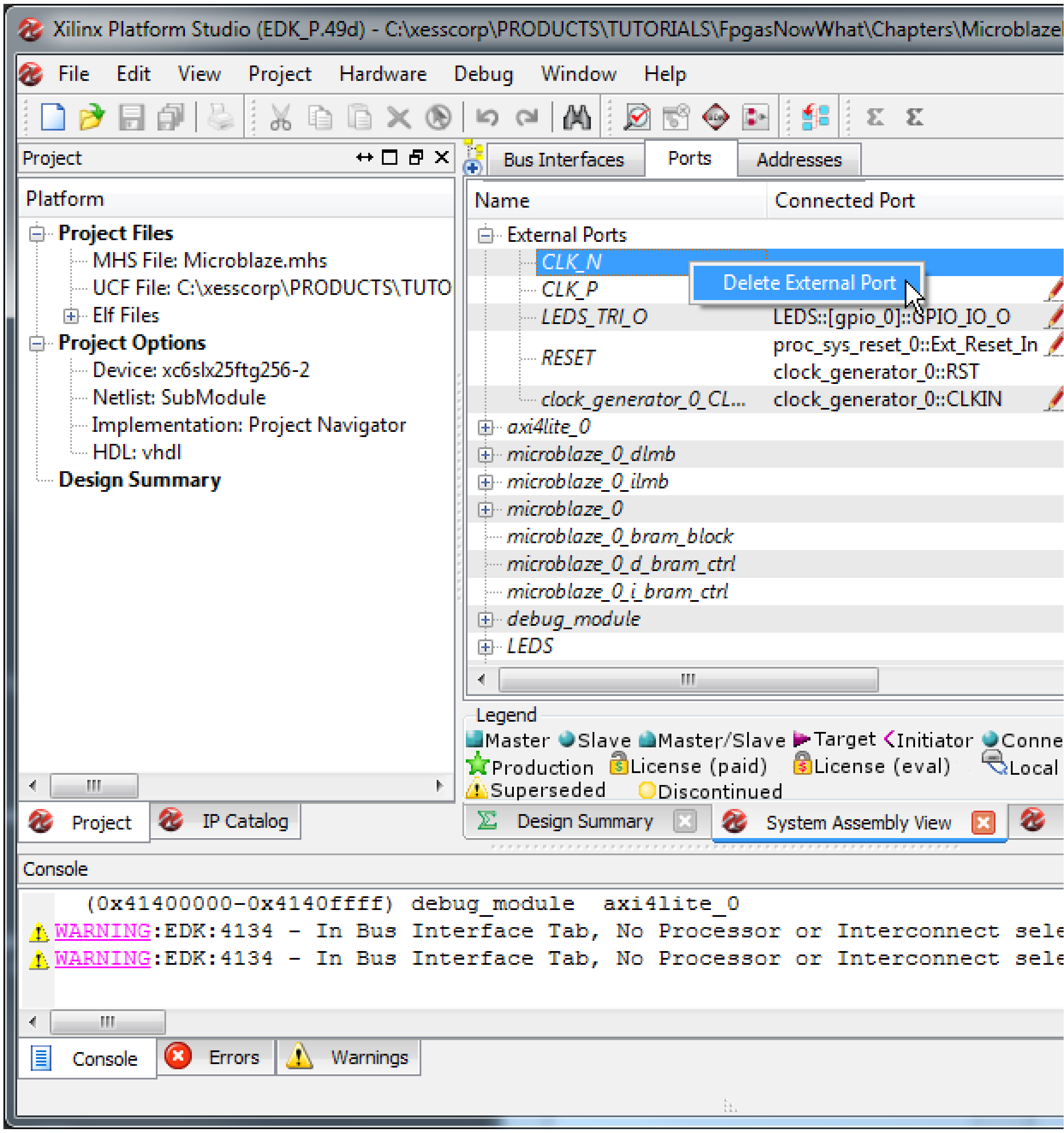
Looking back in the External Ports list, I see that a new port has been connected to the clock generator input:



ne	Connected Port	Dire
xternal Ports		
CLK_N		I
CLK_P		I
LEDS_TRI_O	LEDS::[gpio_0]::GPIO_IO_O	O
RESET	proc_sys_reset_0::Ext_Reset_In clock_generator_0::RST	I
clock_generator_0_CL...	clock_generator_0::CLKIN	I
axi4lite_0		
microblaze_0_dlmb		
microblaze_0_ilmb		
microblaze_0		
microblaze_0_bram_block		
microblaze_0_d_bram_ctrl		
microblaze_0_i_bram_ctrl		
debug_module		
LEDS		

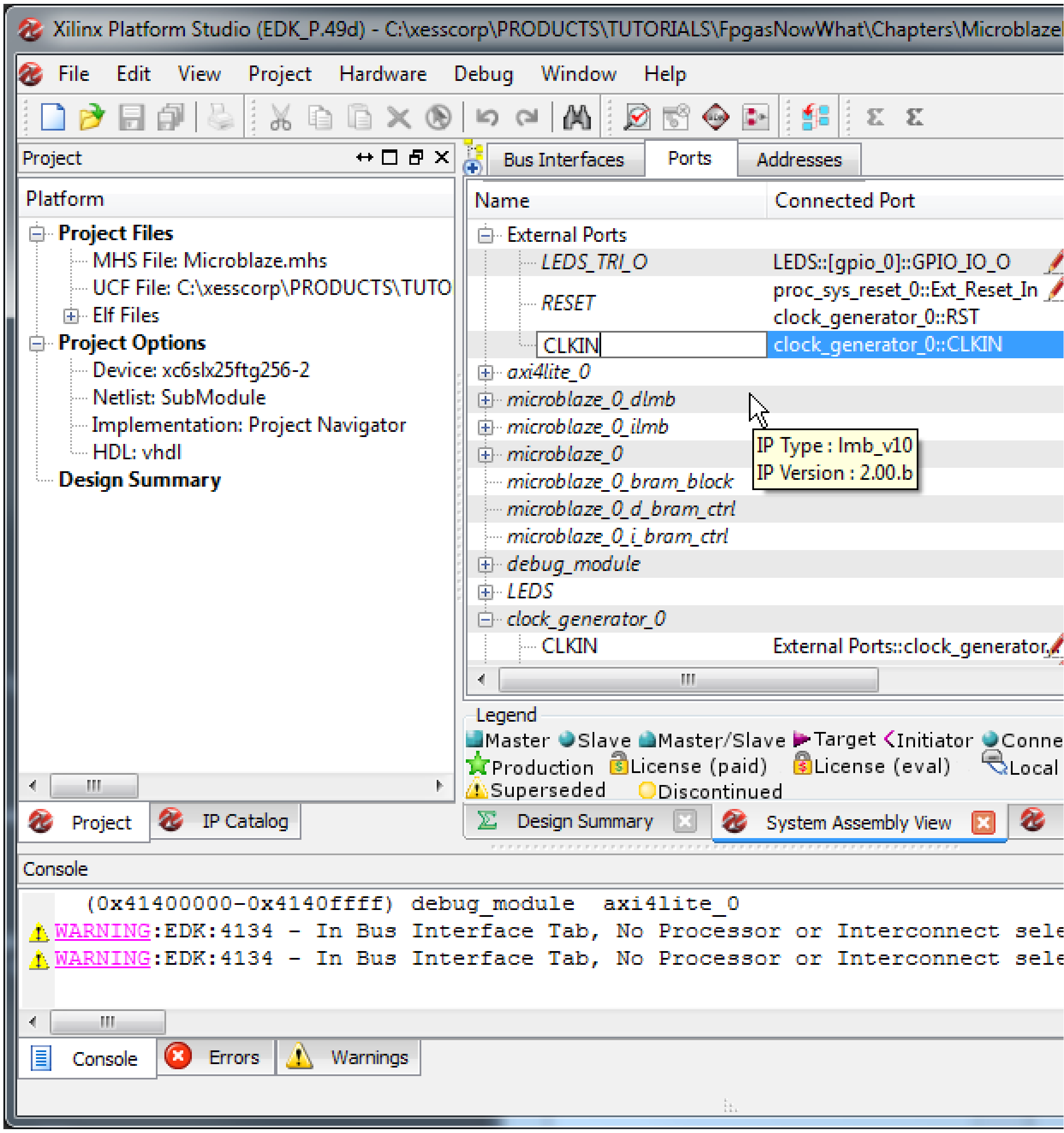
```
   (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

Now I notice that I don't need the differential clock inputs (CLK_N and CLK_P), so I right-click on them and delete them:



```
    (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

Finally, I click on the name field of the newly-created clock input and simplify its name to CLKIN:


Xilinx Platform Studio (EDK_P.49d) - C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\Microblaze

File Edit View Project Hardware Debug Window Help

Project

Platform
- Project Files
  - MHS File: Microblaze.mhs
  - UCF File: C:\xesscorp\PRODUCTS\TUTO
  - Elf Files
- Project Options
  - Device: xc6slx25ftg256-2
  - Netlist: SubModule
  - Implementation: Project Navigator
  - HDL: vhdl
- Design Summary

Bus Interfaces | Ports | Addresses

Name	Connected Port
External Ports	
LEDS_TRI_O	LEDS::[gpio_0]::GPIO_IO_O
RESET	proc_sys_reset_0::Ext_Reset_In clock_generator_0::RST
CLKIN	clock_generator_0::CLKIN
axi4lite_0	
microblaze_0_dlmb	
microblaze_0_ilmb	
microblaze_0	
microblaze_0_bram_block	
microblaze_0_d_bram_ctrl	
microblaze_0_i_bram_ctrl	
debug_module	
LEDS	
clock_generator_0	
CLKIN	External Ports::clock_generator

IP Type : lmb_v10
IP Version : 2.00.b

Legend: Master, Slave, Master/Slave, Target, Initiator, Conne, Production, License (paid), License (eval), Local, Superseded, Discontinued

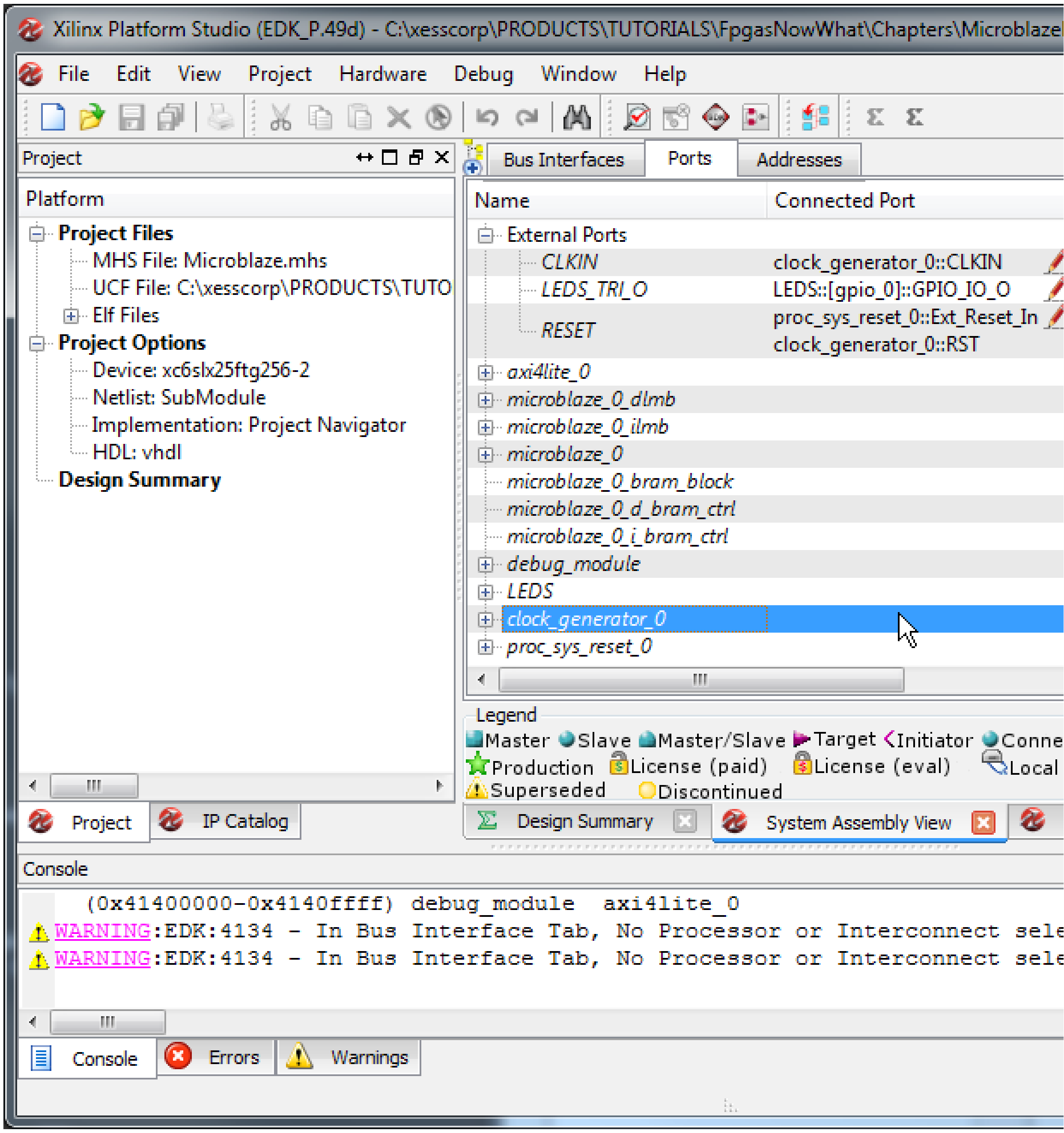
Project | IP Catalog | Design Summary | System Assembly View

Console

```
    (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

Console | Errors | Warnings

Finally, I need to do a little work on the clock generator. I double-click the clock generator entry in the Ports tab and that opens the configuration window for that device.


Xilinx Platform Studio (EDK_P.49d) - C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\Microblaze
File Edit View Project Hardware Debug Window Help
Project
Bus Interfaces
Ports
Addresses
Project
IP Catalog
Design Summary
System Assembly View
Console
Errors
Warnings

Platform
- Project Files
  - MHS File: Microblaze.mhs
  - UCF File: C:\xesscorp\PRODUCTS\TUTO
  - Elf Files
- Project Options
  - Device: xc6slx25ftg256-2
  - Netlist: SubModule
  - Implementation: Project Navigator
  - HDL: vhdl
- Design Summary

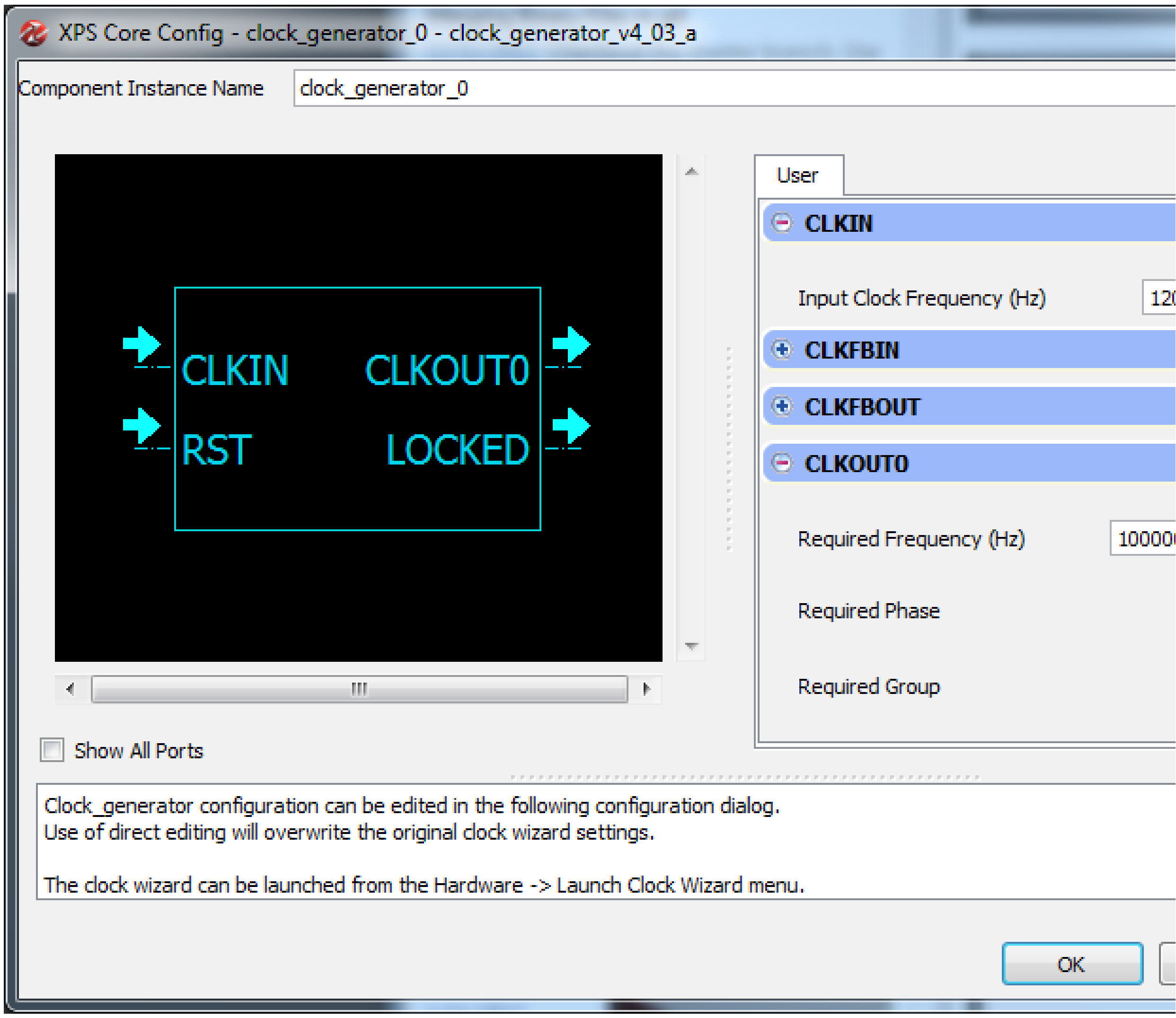
Name	Connected Port
External Ports	
CLKIN	clock_generator_0::CLKIN
LEDS_TRI_O	LEDS::[gpio_0]::GPIO_IO_O
RESET	proc_sys_reset_0::Ext_Reset_In clock_generator_0::RST
axi4lite_0	
microblaze_0_dlmb	
microblaze_0_ilmb	
microblaze_0	
microblaze_0_bram_block	
microblaze_0_d_bram_ctrl	
microblaze_0_i_bram_ctrl	
debug_module	
LEDS	
clock_generator_0	
proc_sys_reset_0	

Legend
Master Slave Master/Slave Target Initiator Conne
Production License (paid) License (eval) Local
Superseded Discontinued

```
   (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

In the configuration window, I expand the CLKIN and CLKOUT0 entries and see their frequencies are 12 and 100 MHz, respectively. That's what I specified when I was back in the BSB, but this is where I could change those if I needed to.

In addition, I change the "Required Group" entry from "NONE" to "DCM0". If I don't do this, ISE will complain that the 12 MHz input clock frequency is too low to be used with the Spartan-6 phase-locked loops (PLL). Forcing ISE to use a Digital Clock Manager (DCM) with looser input requirements avoids this problem.


XPS Core Config - clock_generator_0 - clock_generator_v4_03_a
CLKIN
RST
CLKOUT0
LOCKED
User
CLKFBIN
CLKFBOUT
OK

Component Instance Name: clock_generator_0

CLKIN
Input Clock Frequency (Hz): 12

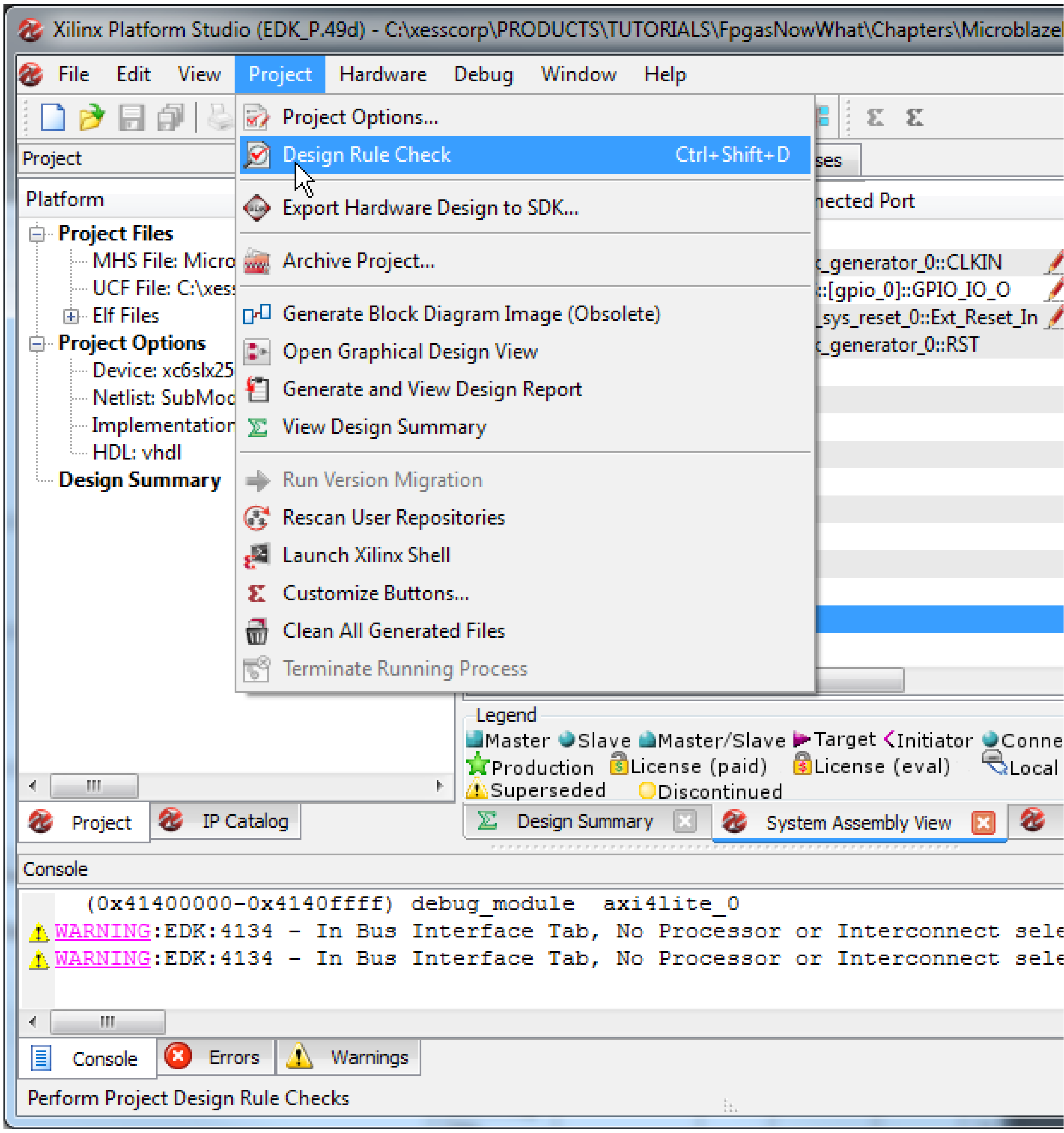
CLKOUT0
Required Frequency (Hz): 10000
Required Phase
Required Group

Show All Ports

Clock_generator configuration can be edited in the following configuration dialog.
Use of direct editing will overwrite the original clock wizard settings.

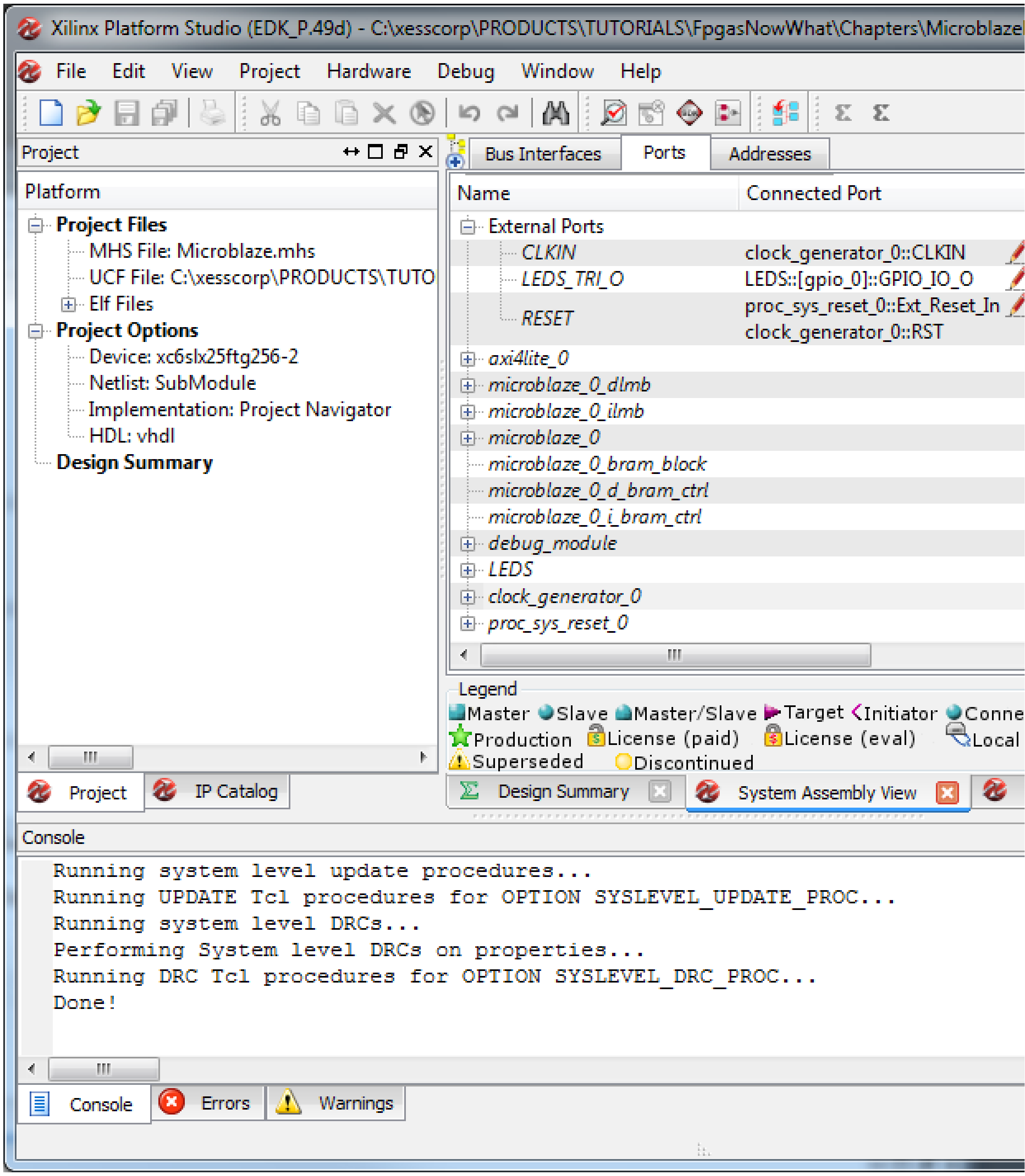
The clock wizard can be launched from the Hardware -> Launch Clock Wizard menu.

After everything has been configured, I run the design rule checker (DRC) to see if any obvious errors exist...


Xilinx Platform Studio (EDK_P.49d) - C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\Microblaze
File Edit View Project Hardware Debug Window Help
Project Options...
Design Rule Check Ctrl+Shift+D
Export Hardware Design to SDK...
Archive Project...
Generate Block Diagram Image (Obsolete)
Open Graphical Design View
Generate and View Design Report
View Design Summary
Run Version Migration
Rescan User Repositories
Launch Xilinx Shell
Customize Buttons...
Clean All Generated Files
Terminate Running Process
Project
Platform
Project Files
MHS File: Micro
UCF File: C:\xes
Elf Files
Project Options
Device: xc6slx25
Netlist: SubMod
Implementation
HDL: vhdl
Design Summary
Legend
Master Slave Master/Slave Target Initiator Conne
Production License (paid) License (eval) Local
Superseded Discontinued
Project IP Catalog
Design Summary System Assembly View
Console
Console Errors Warnings
Perform Project Design Rule Checks

```
    (0x41400000-0x4140ffff) debug_module  axi4lite_0
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
WARNING:EDK:4134 - In Bus Interface Tab, No Processor or Interconnect sele
```

The DRC doesn't find anything, so it looks like I'm good to go! Clicking on File => Exit takes me back to ISE.


Xilinx Platform Studio (EDK_P.49d) - C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\Microblaze

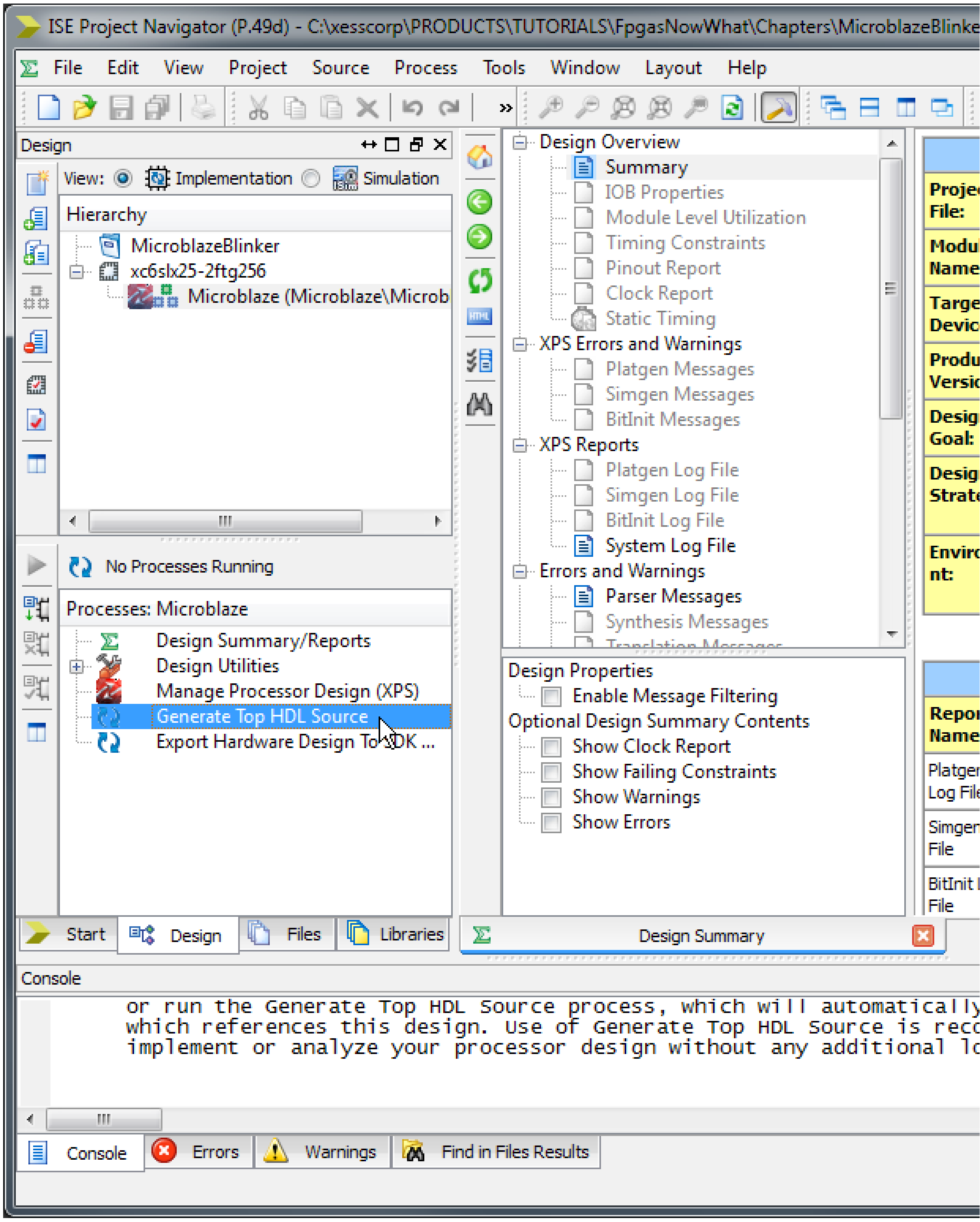
Platform
- Project Files
  - MHS File: Microblaze.mhs
  - UCF File: C:\xesscorp\PRODUCTS\TUTO
  - Elf Files
- Project Options
  - Device: xc6slx25ftg256-2
  - Netlist: SubModule
  - Implementation: Project Navigator
  - HDL: vhdl
- Design Summary

Name	Connected Port
External Ports	
CLKIN	clock_generator_0::CLKIN
LEDS_TRI_O	LEDS::[gpio_0]::GPIO_IO_O
RESET	proc_sys_reset_0::Ext_Reset_In clock_generator_0::RST
axi4lite_0	
microblaze_0_dlmb	
microblaze_0_ilmb	
microblaze_0	
microblaze_0_bram_block	
microblaze_0_d_bram_ctrl	
microblaze_0_i_bram_ctrl	
debug_module	
LEDS	
clock_generator_0	
proc_sys_reset_0	

```
Running system level update procedures...
Running UPDATE Tcl procedures for OPTION SYSLEVEL_UPDATE_PROC...
Running system level DRCs...
Performing System level DRCs on properties...
Running DRC Tcl procedures for OPTION SYSLEVEL_DRC_PROC...
Done!
```

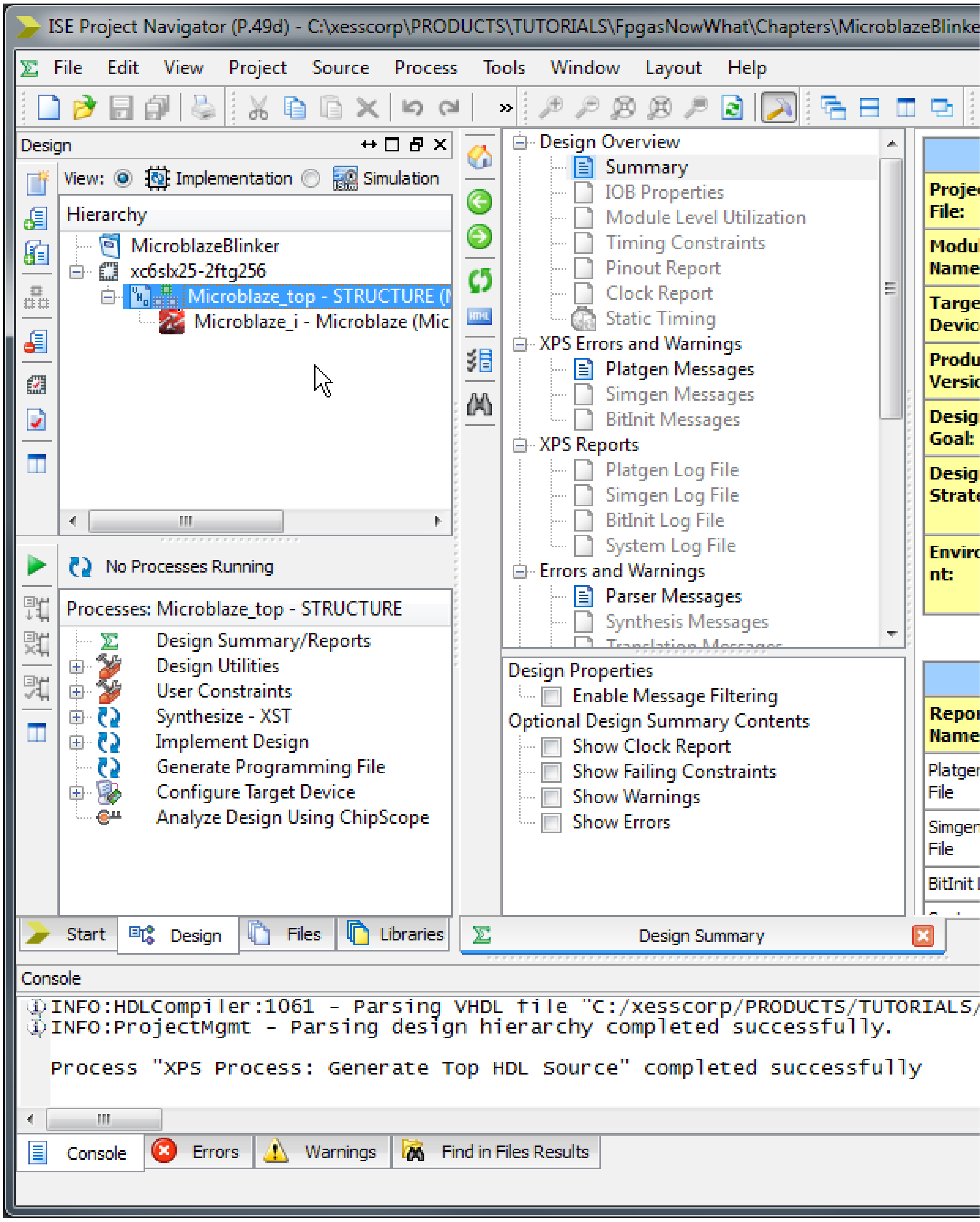
Back to the ISE

After configuring the Microblaze with XPS, There's still some work to do in ISE before an FPGA bitstream can be generated. First of all, a top-level VHDL module has to be wrapped around the processor core. This is done by highlighting the Microblaze module in the Hierarchy pane and then double-clicking the "Generate Top HDL Source" entry that appears in the Processes pane:



```
or run the Generate Top HDL Source process, which will automatically
which references this design. Use of Generate Top HDL Source is reco
implement or analyze your processor design without any additional lo
```

After a few seconds, the top-level module appears with the Microblaze core as its child:



```
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/xesscorp/PRODUCTS/TUTORIALS/
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.

Process "XPS Process: Generate Top HDL Source" completed successfully
```

Double-clicking the Microblaze_top module displays the following VHDL source in the editor. Note that there are three I/O signals on the top level and these connect directly to the Microblaze core I/O:

RESET: The active-low reset input.

LEDS_TRI_O: A single-bit bus output that drives an external LED.

CLKIN: The 12 MHz input clock from the XuLA board.

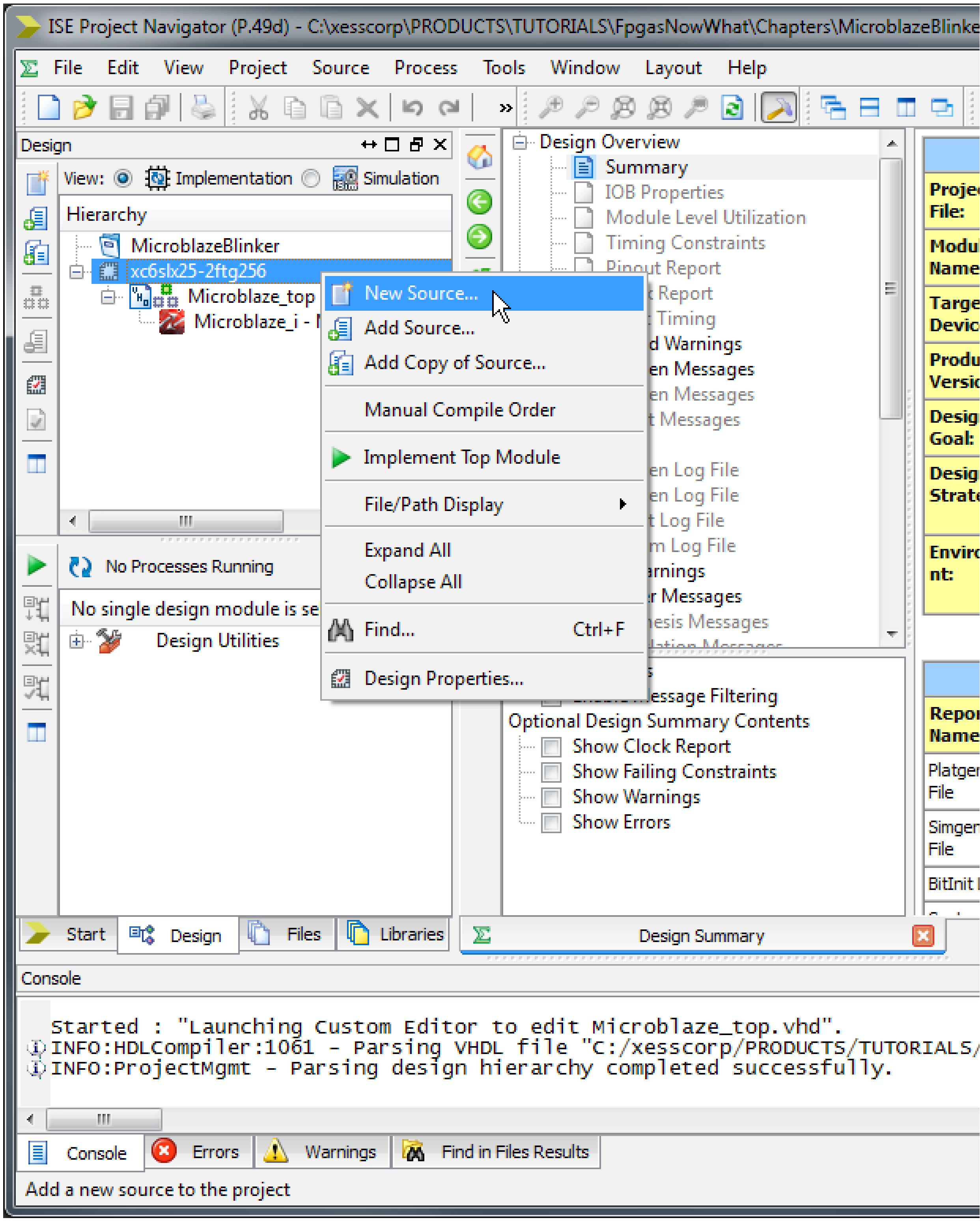
```
--------------------------------------------------------------------------------
-- Microblaze_top.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
use UNISIM.VCOMPONENTS.ALL;

entity Microblaze_top is
  port (
    RESET : in std_logic;
    LEDS_TRI_O : out std_logic_vector(0 to 0);
    CLKIN : in std_logic
  );
end Microblaze_top;

architecture STRUCTURE of Microblaze_top is

  component Microblaze is
    port (
      RESET : in std_logic;
      LEDS_TRI_O : out std_logic_vector(0 to 0);
      CLKIN : in std_logic
    );
  end component;

attribute BOX_TYPE : STRING;
attribute BOX_TYPE of Microblaze : component is "user_black_box";

begin

  Microblaze_i : Microblaze
    port map (
      RESET => RESET,
      LEDS_TRI_O => LEDS_TRI_O(0 to 0),
      CLKIN => CLKIN
    );

end architecture STRUCTURE;
```

That's OK, but I want to simplify it a bit. I remove the RESET input from the top-level entity declaration, and then I force the RESET input high in the instantiation of the Microblaze core. This makes the Microblaze start executing as soon as it gets loaded into the FPGA. It also means I don't have to bother with pulling the RESET input high externally to the FPGA.

```
--------------------------------------------------------------------------------
-- Microblaze_top.vhd
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

library UNISIM;
```

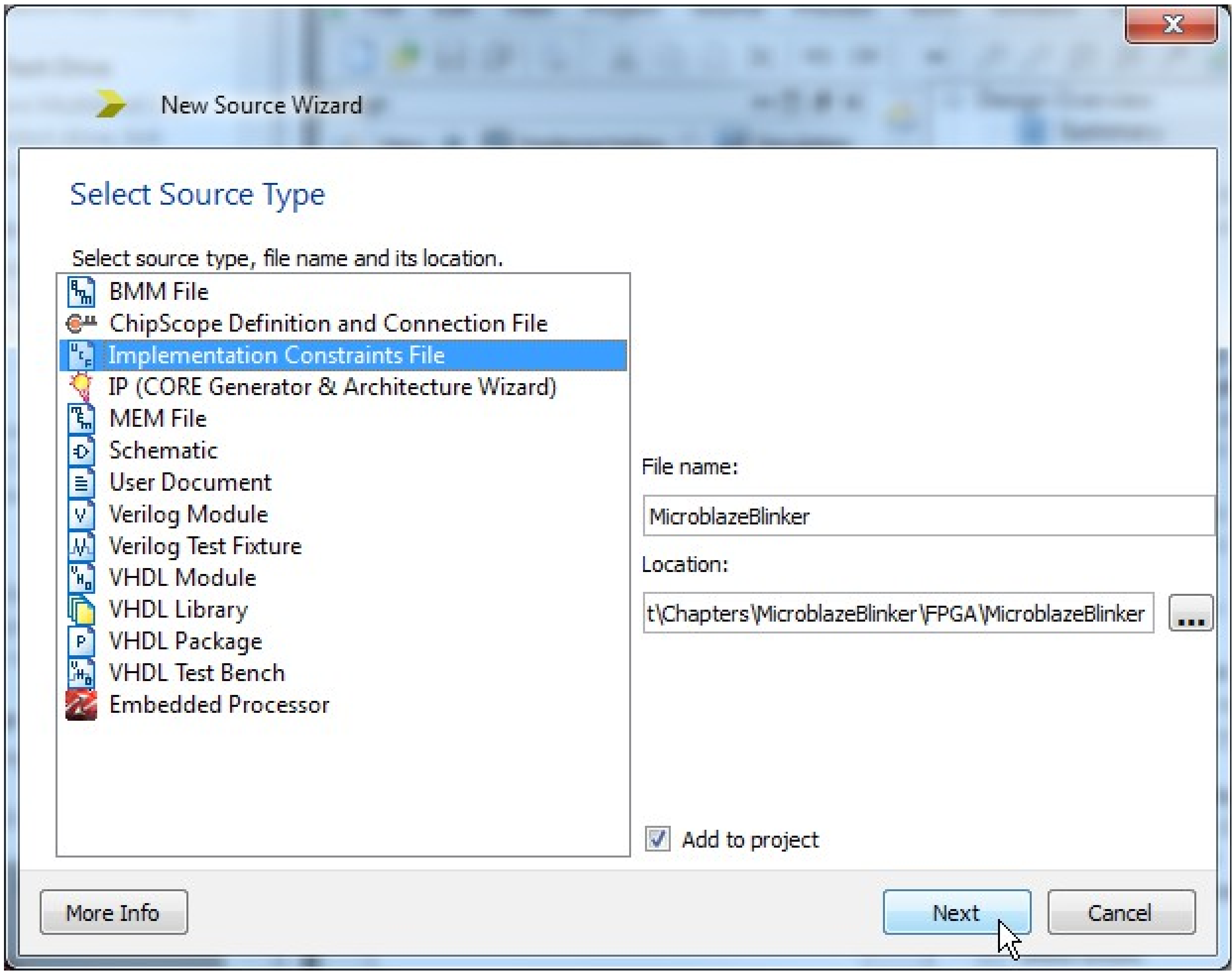
```
use UNISIM.VCOMPONENTS.ALL;

entity Microblaze_top is
  port (
    LEDS_TRI_O : out std_logic_vector(0 to 0);
    CLKIN : in std_logic
  );
end Microblaze_top;

architecture STRUCTURE of Microblaze_top is

  component Microblaze is
    port (
      RESET : in std_logic;
      LEDS_TRI_O : out std_logic_vector(0 to 0);
      CLKIN : in std_logic
    );
  end component;

attribute BOX_TYPE : STRING;
attribute BOX_TYPE of Microblaze : component is "user_black_box";

begin

  Microblaze_i : Microblaze
    port map (
      RESET => '1', -- Enable uBlaze by forcing active-low reset high.
      LEDS_TRI_O => LEDS_TRI_O(0 to 0),
      CLKIN => CLKIN
    );

end architecture STRUCTURE;
```

Now I just have two I/Os to worry about: the LED driver and the input clock. I need to create a UCF file to map these I/Os to specific pins of the FPGA:



```
Started : "Launching Custom Editor to edit Microblaze_top.vhd".
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/xesscorp/PRODUCTS/TUTORIALS/
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.
```

I select an "Implementation Constraints File" type and call it "MicroblazeBlinker" (thus maintaining my streak of creative file names):



Select Source Type

Select source type, file name and its location.

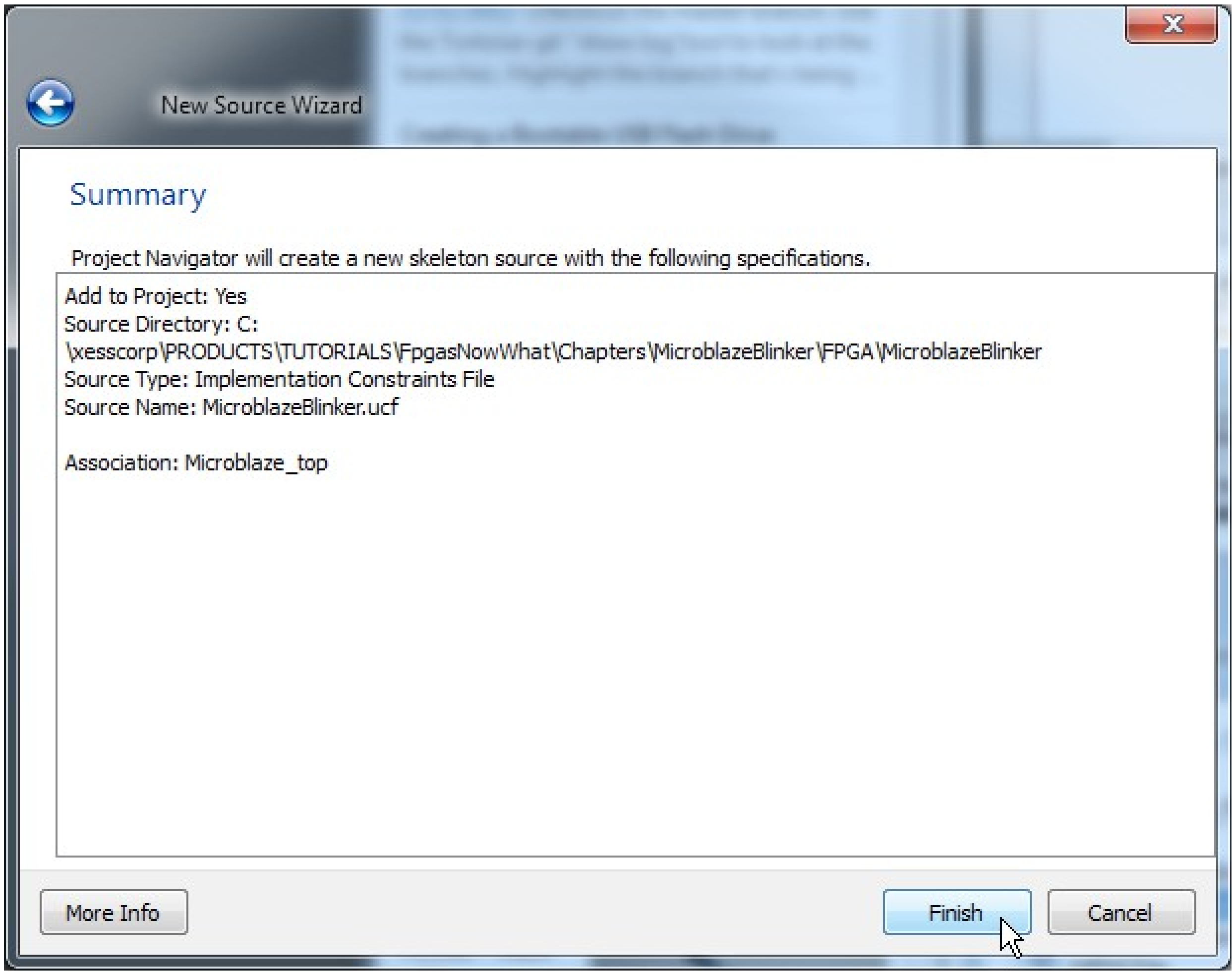
- BMM File
- ChipScope Definition and Connection File
- Implementation Constraints File
- IP (CORE Generator & Architecture Wizard)
- MEM File
- Schematic
- User Document
- Verilog Module
- Verilog Test Fixture
- VHDL Module
- VHDL Library
- VHDL Package
- VHDL Test Bench
- Embedded Processor

File name: MicroblazeBlinker

Location: t\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker

[✓] Add to project

I go through the usual hoo-hah with a confirmation screen telling me what I've just finished doing:



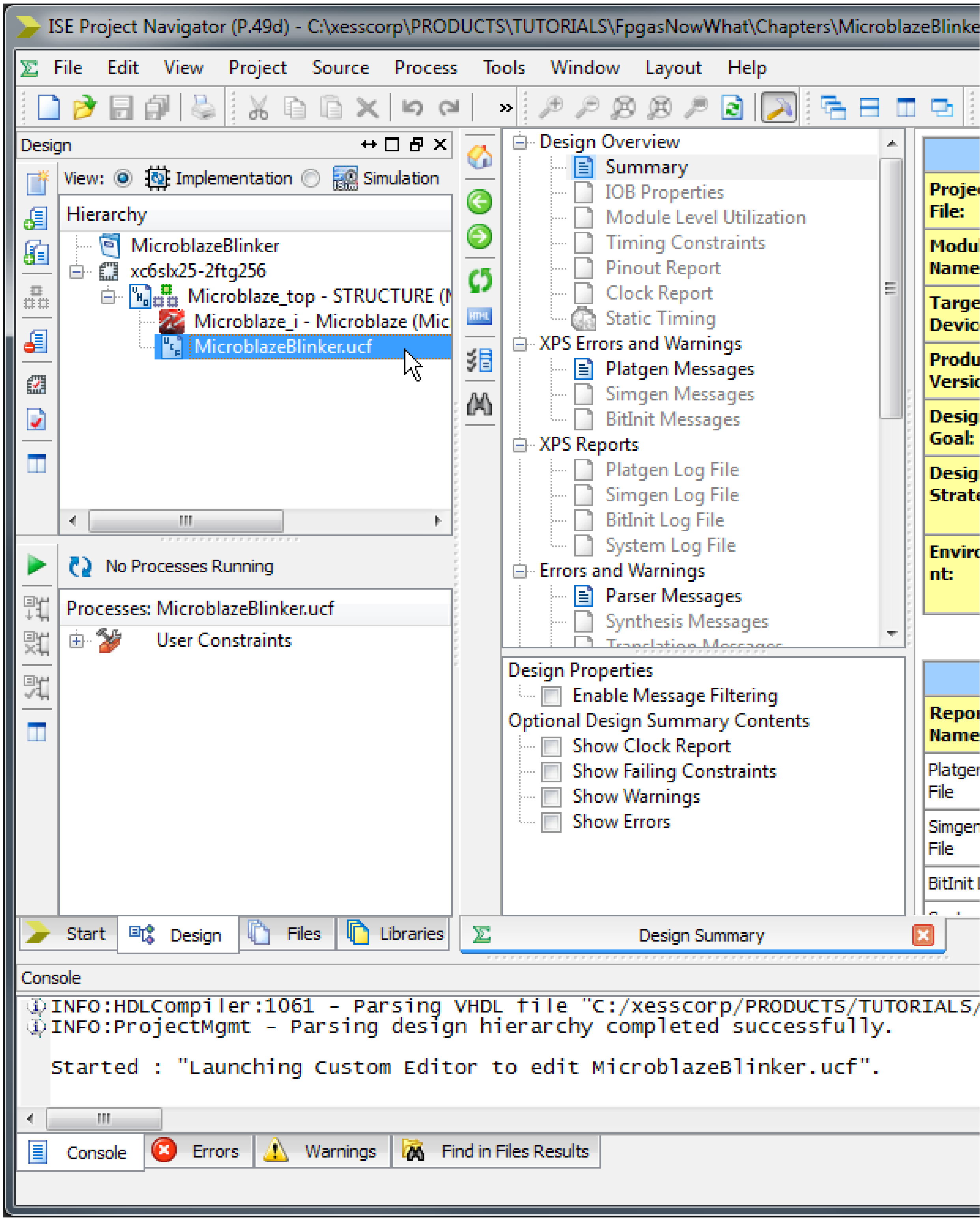
Summary

Project Navigator will create a new skeleton source with the following specifications.

Add to Project: Yes
Source Directory: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker
Source Type: Implementation Constraints File
Source Name: MicroblazeBlinker.ucf

Association: Microblaze_top

Once the constraints file has been added to the ISE project, I can double click it so I can enter the pin assignments:

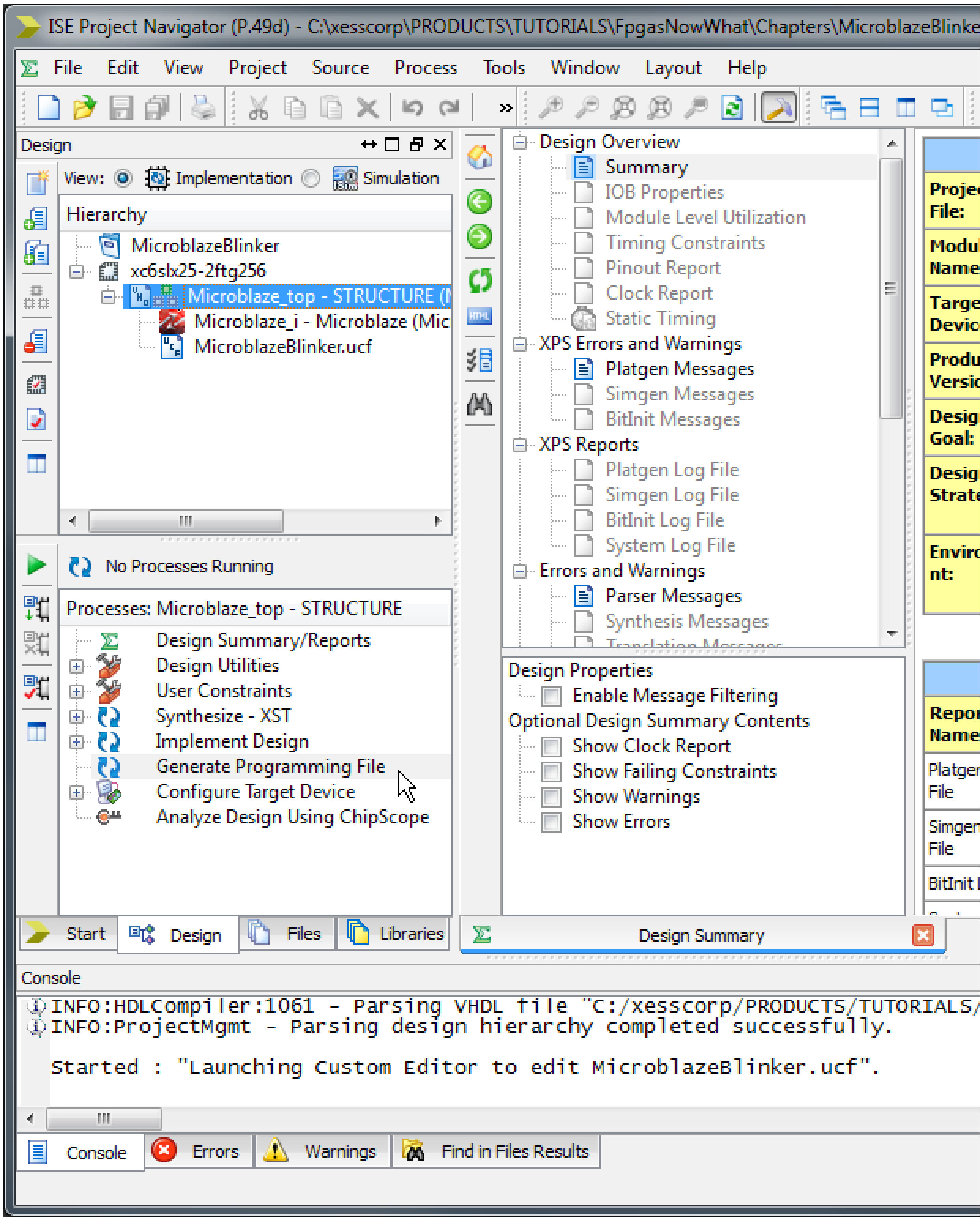


```
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/xesscorp/PRODUCTS/TUTORIALS/
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.

Started : "Launching Custom Editor to edit MicroblazeBlinker.ucf".
```

Here are the pin assignment constraints I used for the XuLA2 board:

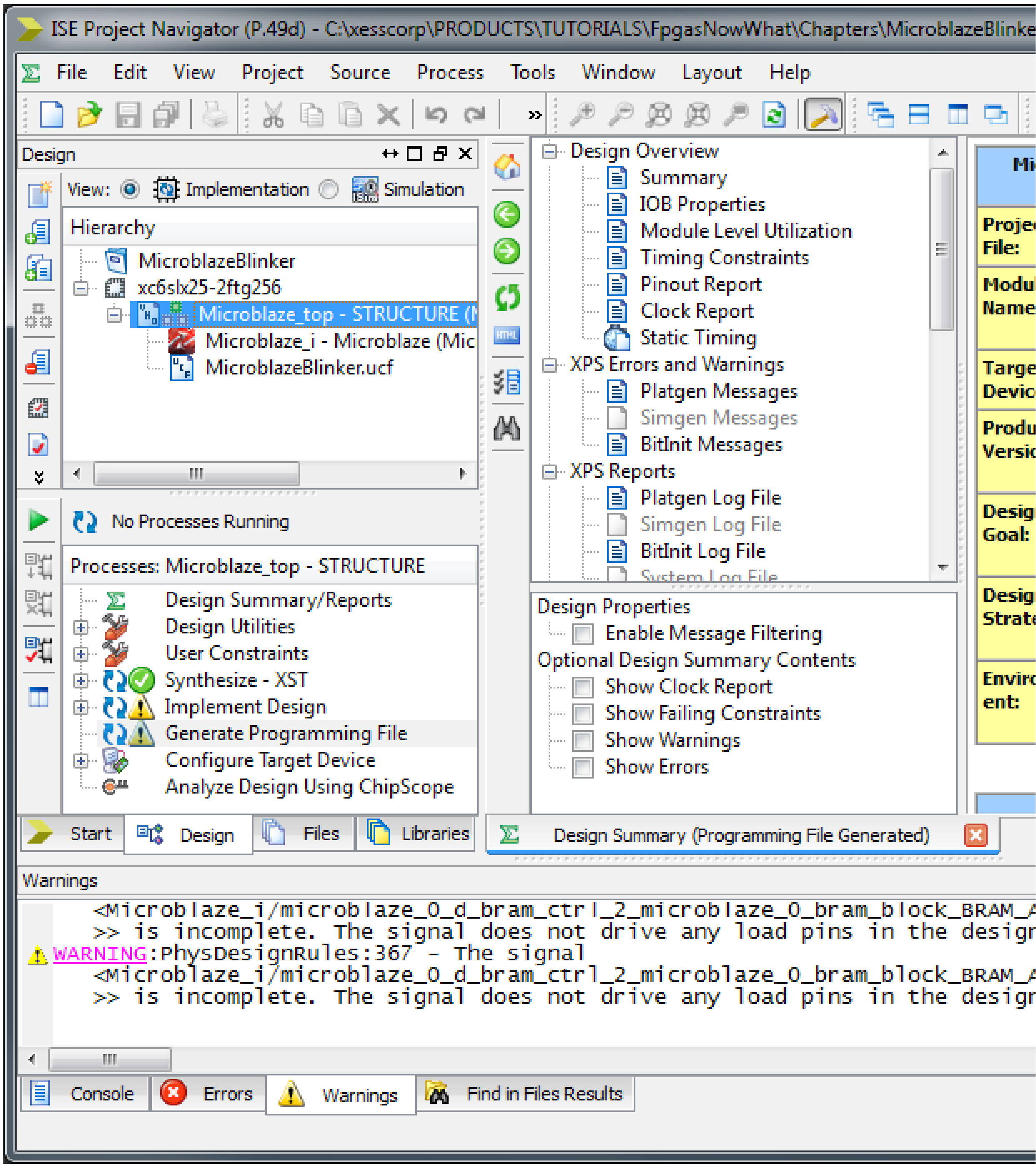
```
net CLKIN          loc = A9; # 12 MHz clock on XuLA2.
net LEDS_TRI_O<0> loc = R7; # Channel 0 on XuLA2.
```

After closing the constraints file, I'm finally ready to generate a bitstream. I highlight the top-level module and double click "Generate Programming File".



```
INFO:HDLCompiler:1061 - Parsing VHDL file "C:/xesscorp/PRODUCTS/TUTORIALS/
INFO:ProjectMgmt - Parsing design hierarchy completed successfully.

Started : "Launching Custom Editor to edit MicroblazeBlinker.ucf".
```

The design is compiled into a bitstream after a few minutes (the Microblaze is *big*!). A lot of warnings get generated because much of the Microblaze core I/O isn't connected to anything in such a small system. The main thing is that no errors were encountered that would prematurely halt the bitstream generation.

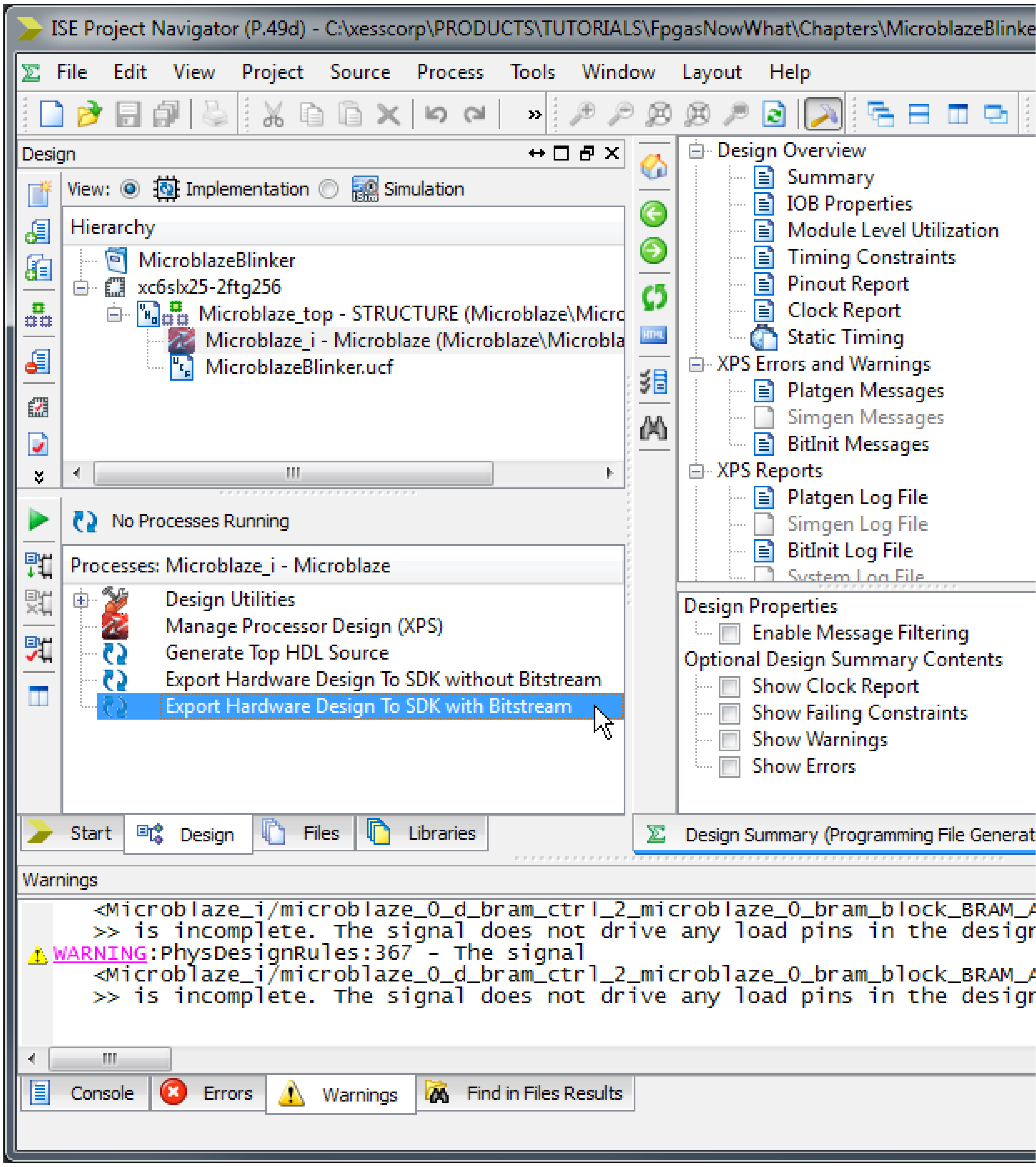


```
    <Microblaze_i/microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block_BRAM_
    >> is incomplete. The signal does not drive any load pins in the desig
WARNING:PhysDesignRules:367 - The signal
    <Microblaze_i/microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block_BRAM_
    >> is incomplete. The signal does not drive any load pins in the desig
```

So now there's a bitstream containing a Microblaze processor with a couple of I/O pins tied to the external world. But it seems like I'm forgetting something; what could it be? Anybody know? *Anybody!?*

How about a program for the processor to run? In order to write the processor code, I have to start the Software Development Kit (SDK) by highlighting the Microblaze module in the Hierarchy pane and then double-clicking the "Export Hardware Design to SDK with

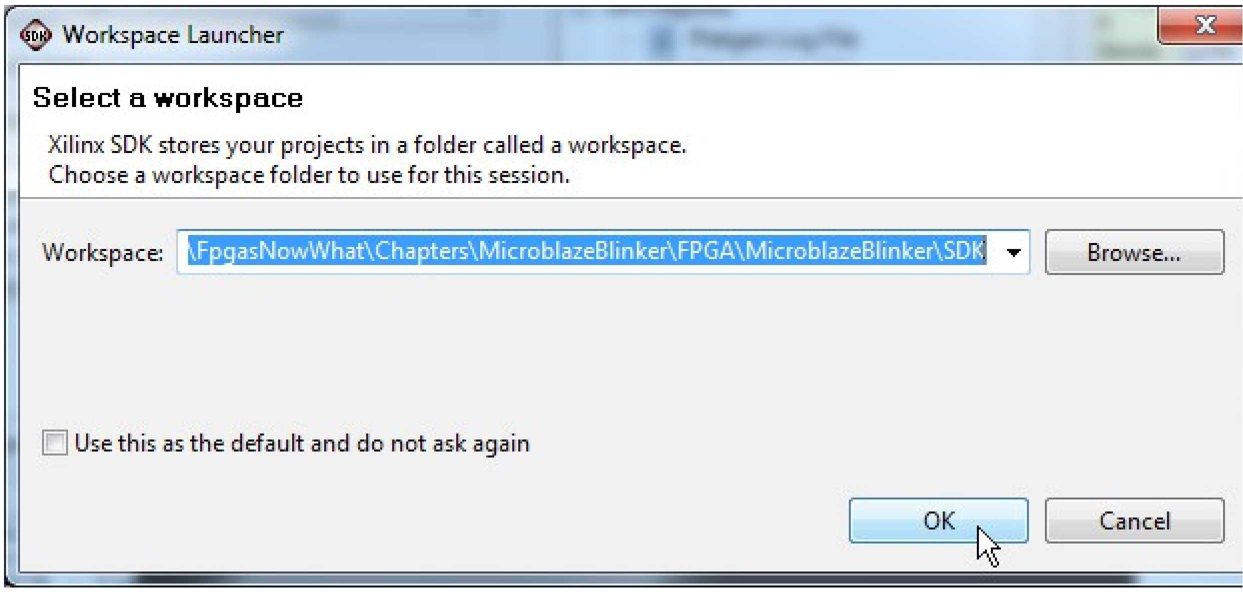
Bitstream" in the Processes pane.


ISE Project Navigator (P.49d) - C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinke
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Hierarchy
MicroblazeBlinker
xc6slx25-2ftg256
Microblaze_top - STRUCTURE (Microblaze\Micro
Microblaze_i - Microblaze (Microblaze\Microbla
MicroblazeBlinker.ucf
No Processes Running
Processes: Microblaze_i - Microblaze
Design Utilities
Manage Processor Design (XPS)
Generate Top HDL Source
Export Hardware Design To SDK without Bitstream
Export Hardware Design To SDK with Bitstream
Start Design Files Libraries
Design Overview
Summary
IOB Properties
Module Level Utilization
Timing Constraints
Pinout Report
Clock Report
Static Timing
XPS Errors and Warnings
Platgen Messages
Simgen Messages
BitInit Messages
XPS Reports
Platgen Log File
Simgen Log File
BitInit Log File
System Log File
Design Properties
Enable Message Filtering
Optional Design Summary Contents
Show Clock Report
Show Failing Constraints
Show Warnings
Show Errors
Design Summary (Programming File Generat
Warnings
Console Errors Warnings Find in Files Results

```
    <Microblaze_i/microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block_BRAM_A
    >> is incomplete. The signal does not drive any load pins in the design
WARNING:PhysDesignRules:367 - The signal
    <Microblaze_i/microblaze_0_d_bram_ctrl_2_microblaze_0_bram_block_BRAM_A
    >> is incomplete. The signal does not drive any load pins in the design
```

Finishing Off with the SDK

Once the SDK has started, it asks me where I want to place my software project. By default, it's placed in the SDK subdirectory within the main ISE project directory, and that's fine with me so I just click on the OK button:

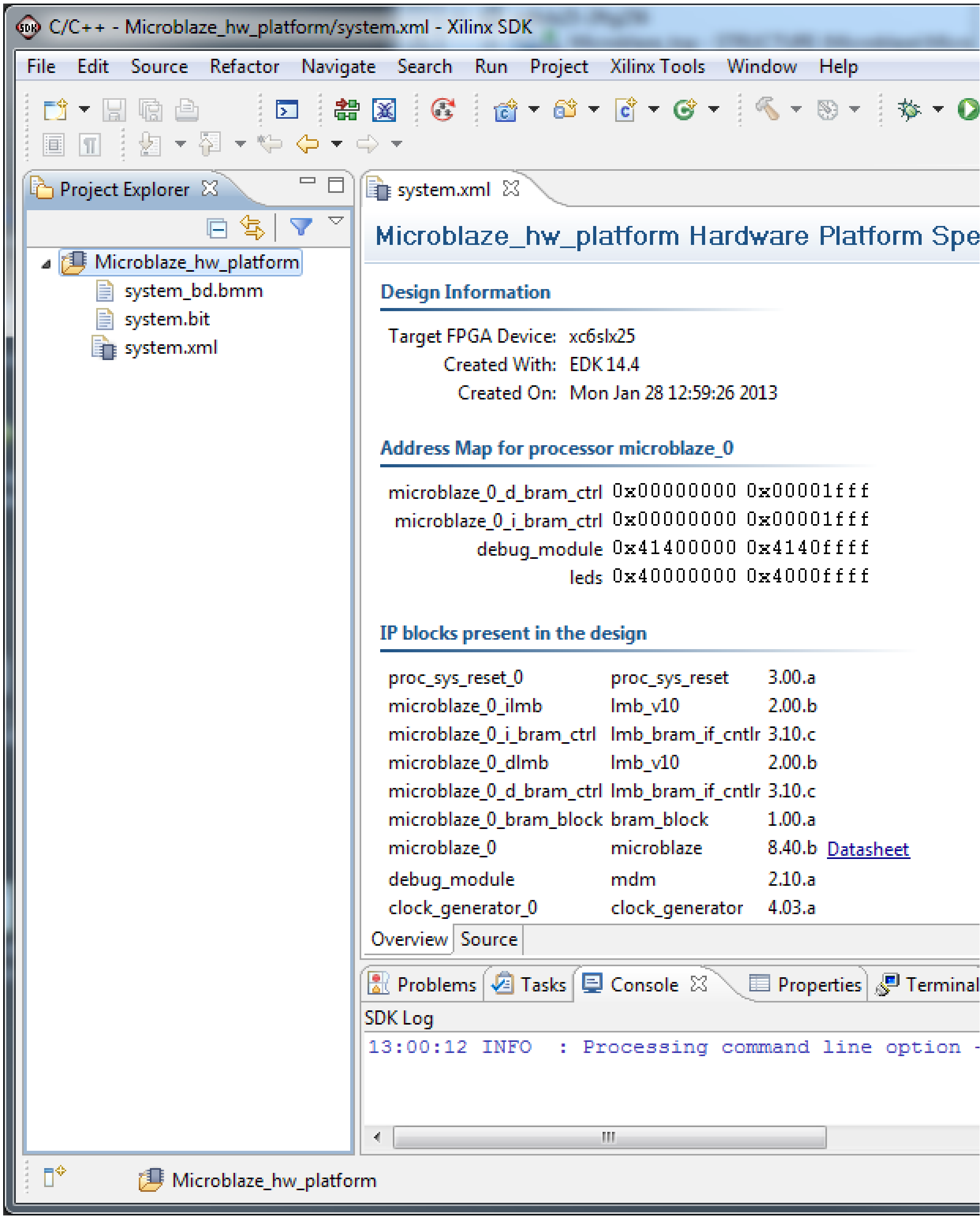

Workspace Launcher

Select a workspace

Xilinx SDK stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: \FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\MicroblazeBlinker\SDK

That brings up the main SDK window with just the hardware system description files:


C/C++ - Microblaze_hw_platform/system.xml - Xilinx SDK
File Edit Source Refactor Navigate Search Run Project Xilinx Tools Window Help
Project Explorer
Microblaze_hw_platform
system_bd.bmm
system.bit
system.xml
Overview Source
Problems Tasks Console Properties Terminal
Microblaze_hw_platform

Microblaze_hw_platform Hardware Platform Spe

Design Information

Target FPGA Device:	xc6slx25
Created With:	EDK 14.4
Created On:	Mon Jan 28 12:59:26 2013

Address Map for processor microblaze_0

microblaze_0_d_bram_ctrl	0x00000000	0x00001fff
microblaze_0_i_bram_ctrl	0x00000000	0x00001fff
debug_module	0x41400000	0x4140ffff
leds	0x40000000	0x4000ffff

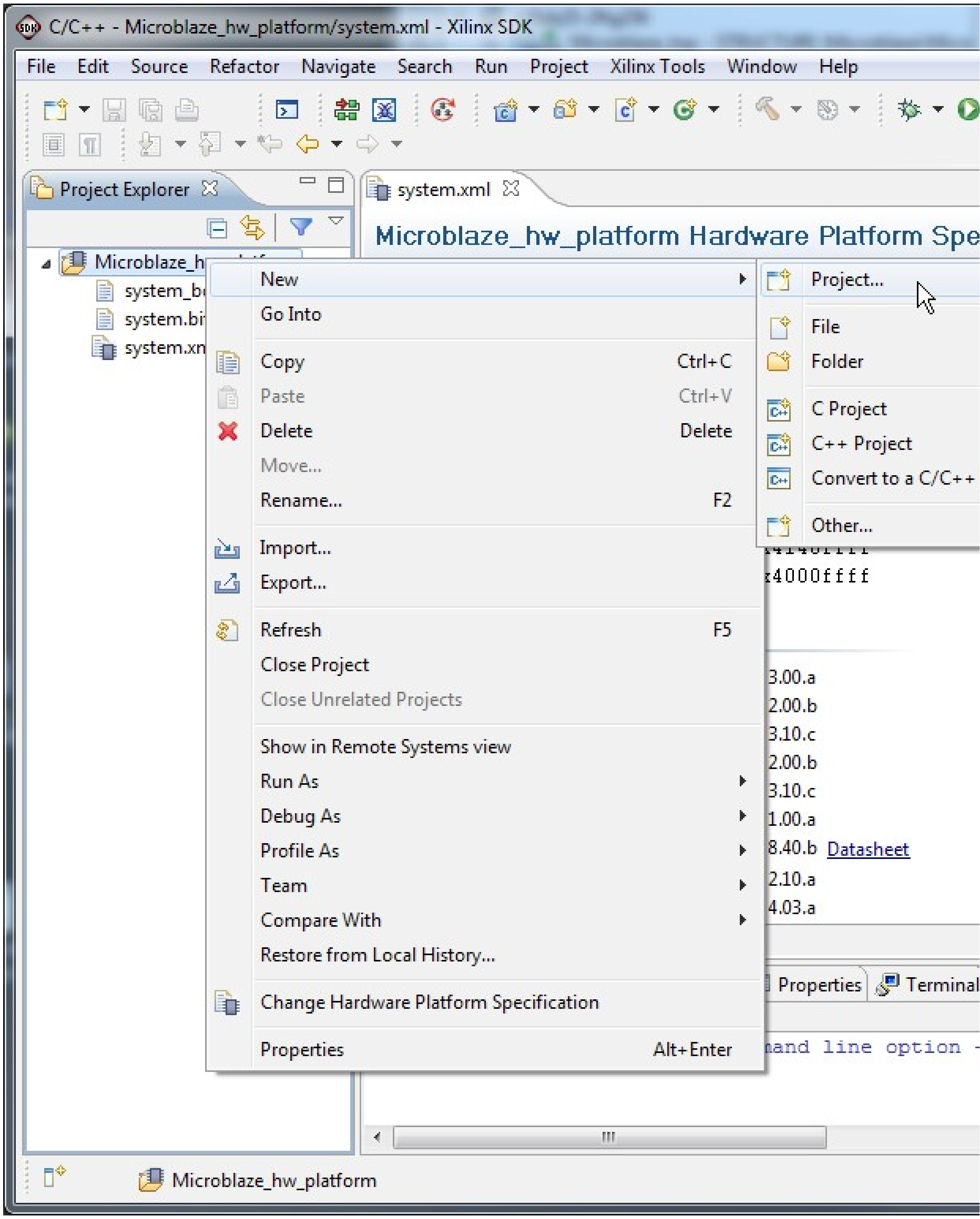
IP blocks present in the design

proc_sys_reset_0	proc_sys_reset	3.00.a	
microblaze_0_ilmb	lmb_v10	2.00.b	
microblaze_0_i_bram_ctrl	lmb_bram_if_cntlr	3.10.c	
microblaze_0_dlmb	lmb_v10	2.00.b	
microblaze_0_d_bram_ctrl	lmb_bram_if_cntlr	3.10.c	
microblaze_0_bram_block	bram_block	1.00.a	
microblaze_0	microblaze	8.40.b	Datasheet
debug_module	mdm	2.10.a	
clock_generator_0	clock_generator	4.03.a	

SDK Log

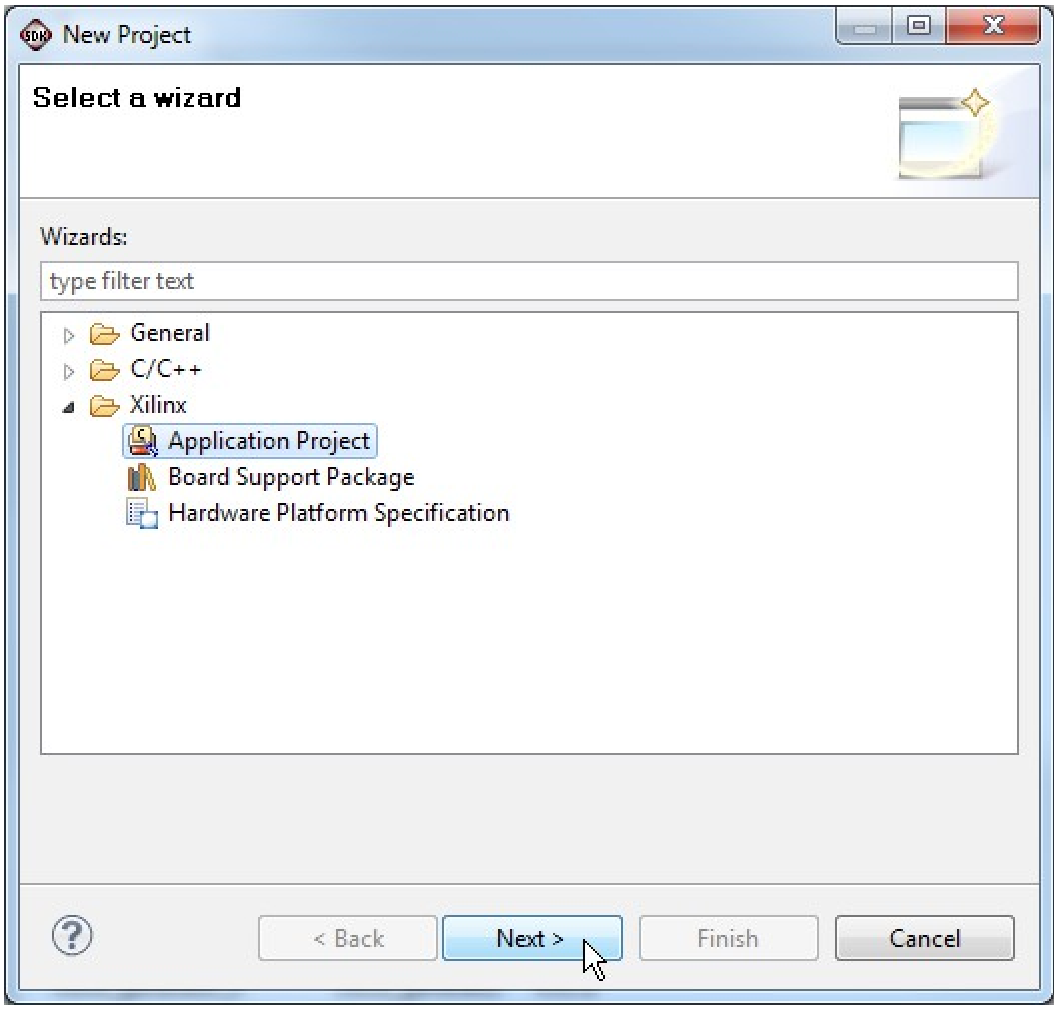
```
13:00:12 INFO  : Processing command line option -
```

First, I need to create a software project for the code the Microblaze will run. Right-clicking on the Microblaze_hw_platform entry in the Project Explorer tab, I select the "Project..." item in the pop-up menu:



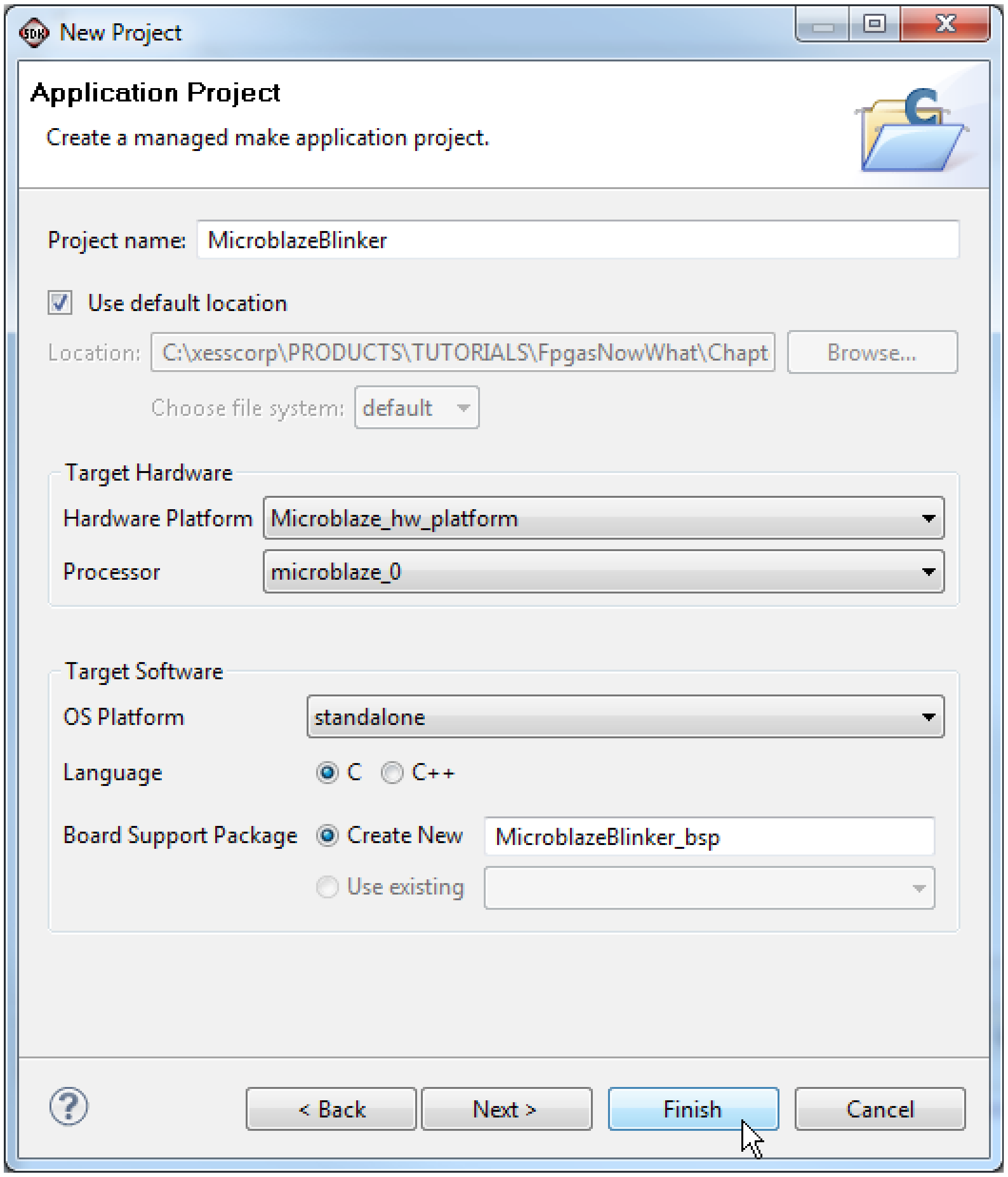
```
x4000ffff
3.00.a
2.00.b
3.10.c
2.00.b
3.10.c
1.00.a
8.40.b Datasheet
2.10.a
4.03.a
and line option -
```

In the New Project window, I elect to begin a Xilinx Application Project:



Once again, another project requiring another project name. I'll just follow tradition and name it "MicroblazeBlinker".

I don't need an operating system to blink an LED, so I select "standalone" in the OS Platform field. And I elect to write the code in C since I don't really want to build and instantiate an LED object C++ class.



Application Project

Create a managed make application project.

Project name: MicroblazeBlinker

[✓] Use default location

Location: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapt

Choose file system: default

Target Hardware

Hardware Platform: Microblaze_hw_platform

Processor: microblaze_0

Target Software

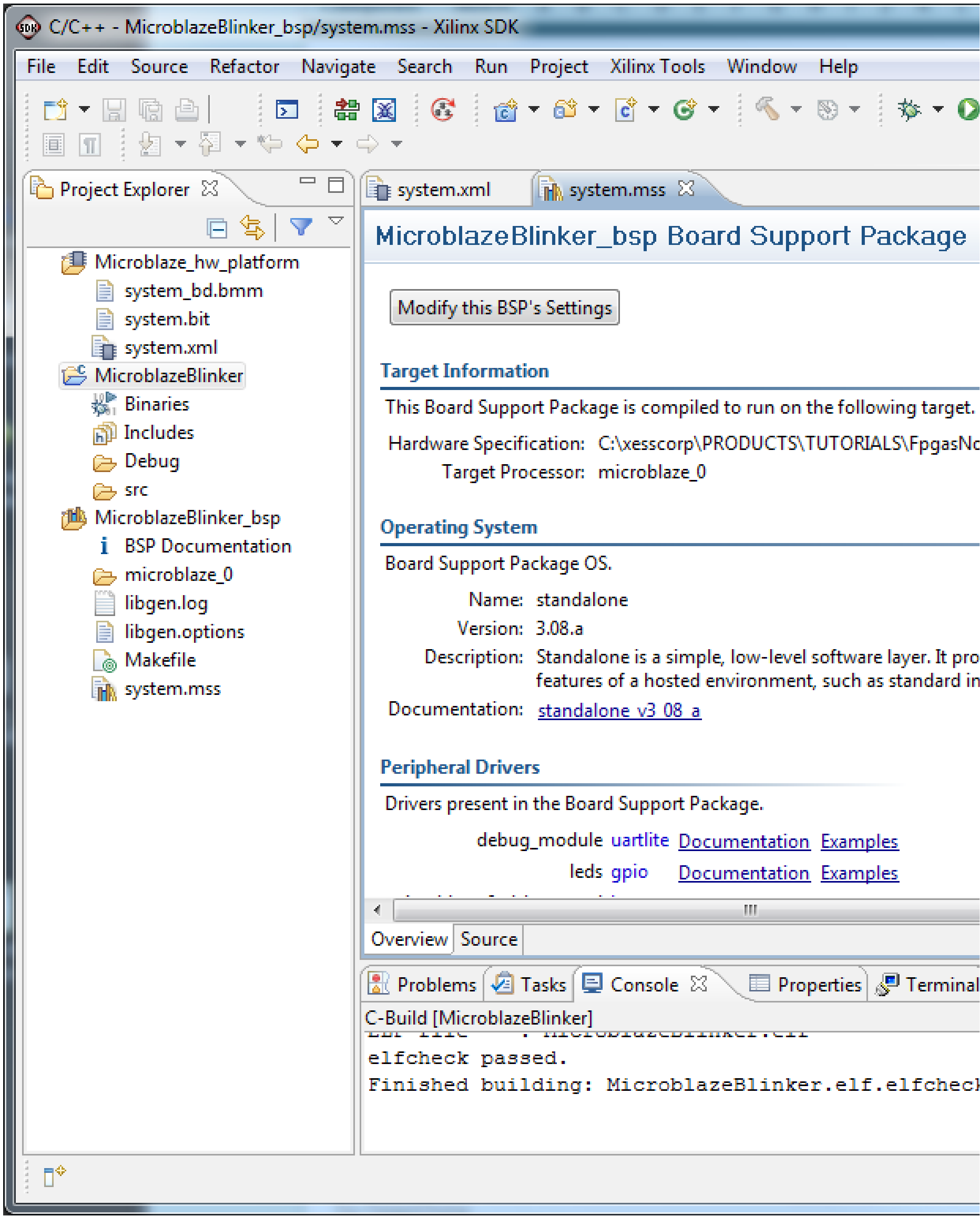
OS Platform: standalone

Language: [● C] C++

Board Support Package: [● Create New] MicroblazeBlinker_bsp; Use existing

After clicking Finish in the New Project window, two new things enter the Project Explorer pane:

- A Board Support Package (BSP) that contains a bunch of header files, library code and documentation for the Microblaze core and peripherals I've included in my ISE project.
- A default software project for the Microblaze.



MicroblazeBlinker_bsp Board Support Package

Target Information

This Board Support Package is compiled to run on the following target.

Hardware Specification: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNc

Target Processor: microblaze_0

Operating System

Board Support Package OS.

Name: standalone

Version: 3.08.a

Description: Standalone is a simple, low-level software layer. It pro features of a hosted environment, such as standard in

Documentation: standalone_v3_08_a

Peripheral Drivers

Drivers present in the Board Support Package.

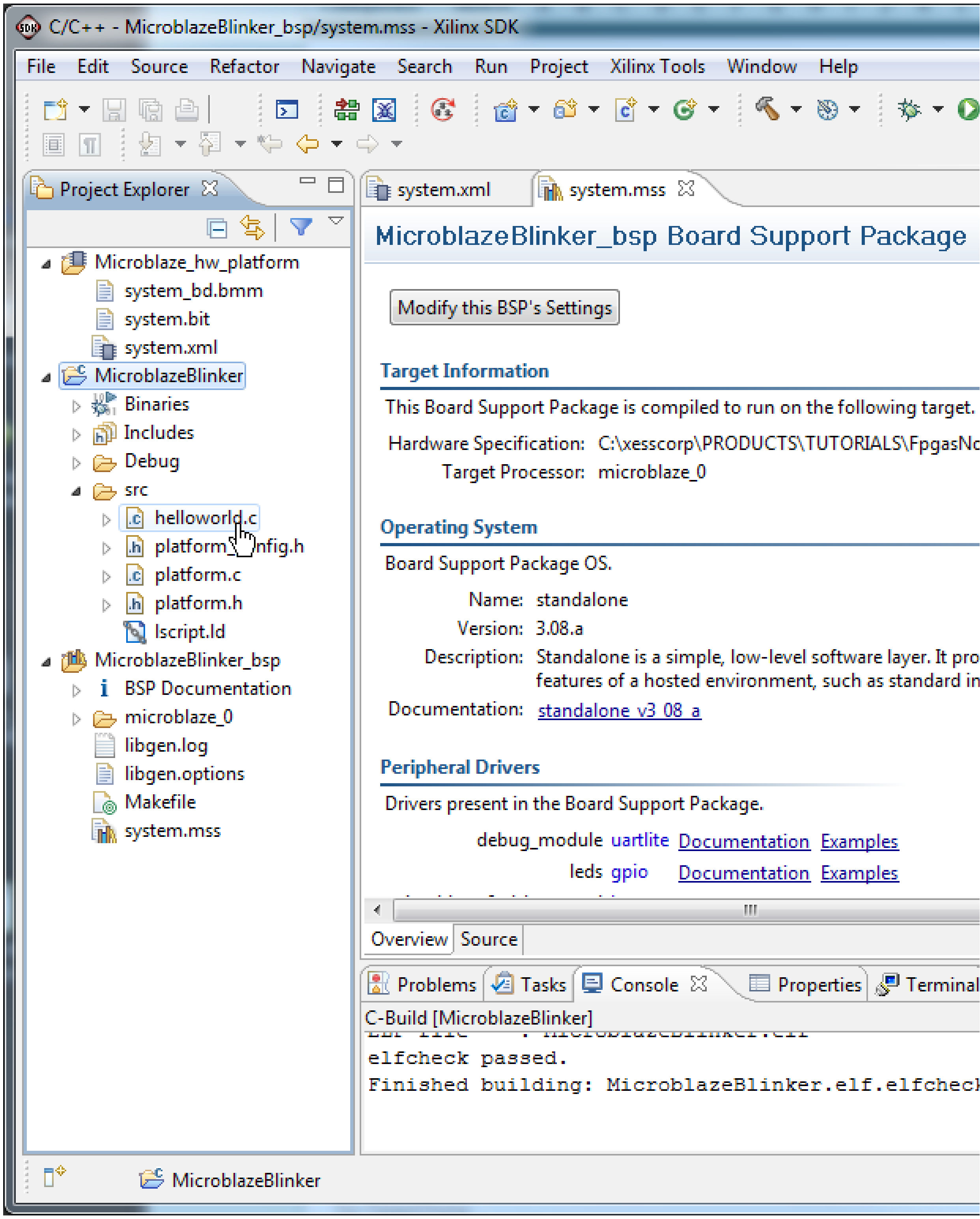
debug_module uartlite Documentation Examples

leds gpio Documentation Examples

C-Build [MicroblazeBlinker]

```
elfcheck passed.
Finished building: MicroblazeBlinker.elf.elfcheck
```

Expanding the MicroblazeBlinker software project, I can double-click on the helloworld.c source file to see what the default program does.



MicroblazeBlinker_bsp Board Support Package

Target Information

This Board Support Package is compiled to run on the following target.

Hardware Specification: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNo

Target Processor: microblaze_0

Operating System

Board Support Package OS.

Name: standalone
Version: 3.08.a
Description: Standalone is a simple, low-level software layer. It pro
features of a hosted environment, such as standard in
Documentation: standalone_v3_08_a

Peripheral Drivers

Drivers present in the Board Support Package.

debug_module	uartlite	Documentation	Examples
leds	gpio	Documentation	Examples

C-Build [MicroblazeBlinker]

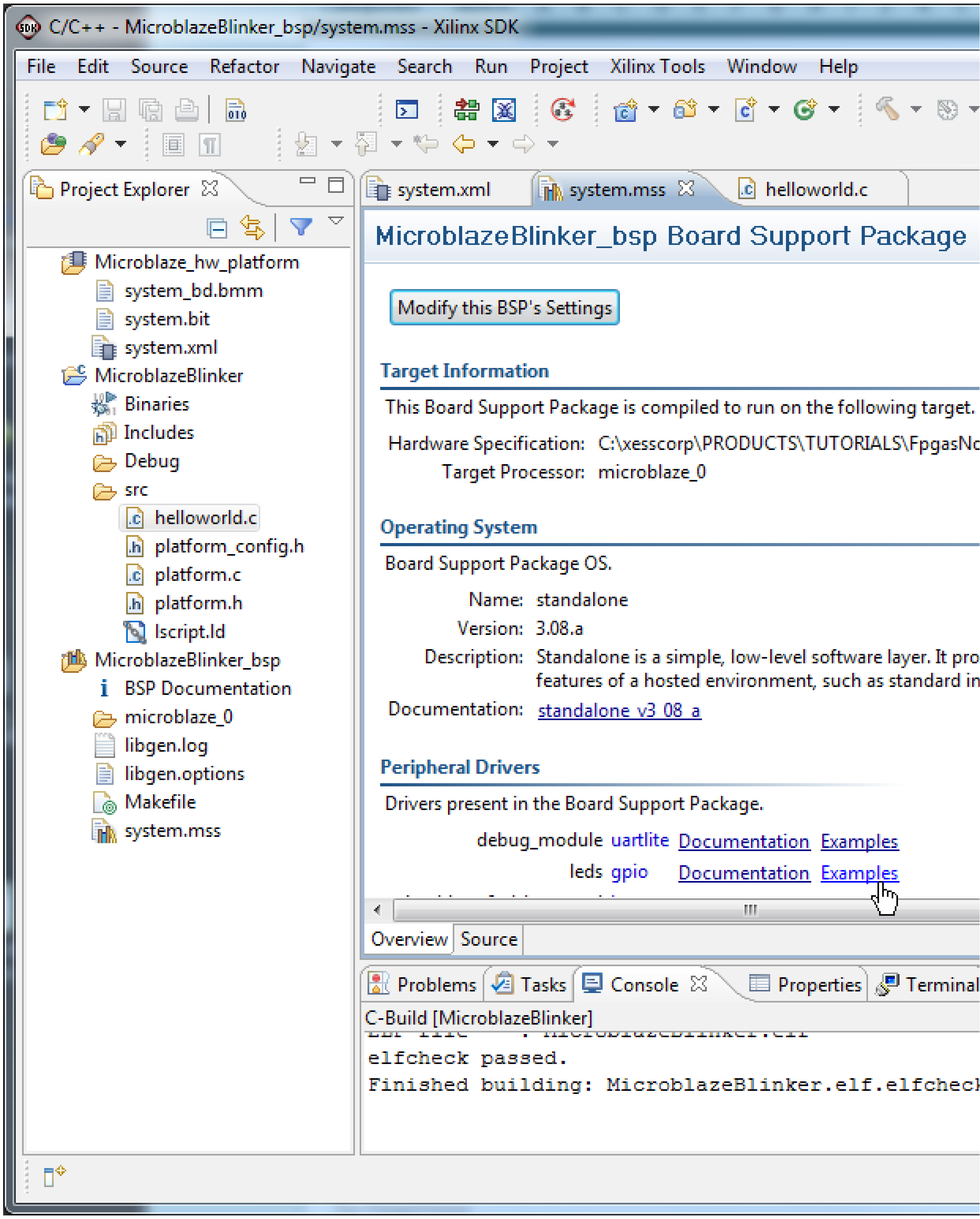
```
elfcheck passed.
Finished building: MicroblazeBlinker.elf.elfcheck
```

The default program is certainly simple enough, but it doesn't do anything I want. There's nothing about blinking an LED in there, and I don't even have a way to display anything that might be printed. Obviously, there's some work to be done here.

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

```
#include "platform.h"

void print( char *str);

int main()
{
  init_platform();

  print("Hello World\n\r");

  return 0;
}
```

Looking in the system.mss file in the BSP, I see that the I/O driver for the LED has some links for documentation and example code. Since "monkey-see, monkey-do" has always worked for me, I decide to click on the Examples link.



MicroblazeBlinker_bsp Board Support Package

Modify this BSP's Settings

Target Information

This Board Support Package is compiled to run on the following target.

Hardware Specification: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNc

Target Processor: microblaze_0

Operating System

Board Support Package OS.

Name: standalone

Version: 3.08.a

Description: Standalone is a simple, low-level software layer. It pro features of a hosted environment, such as standard in

Documentation: standalone v3 08 a

Peripheral Drivers

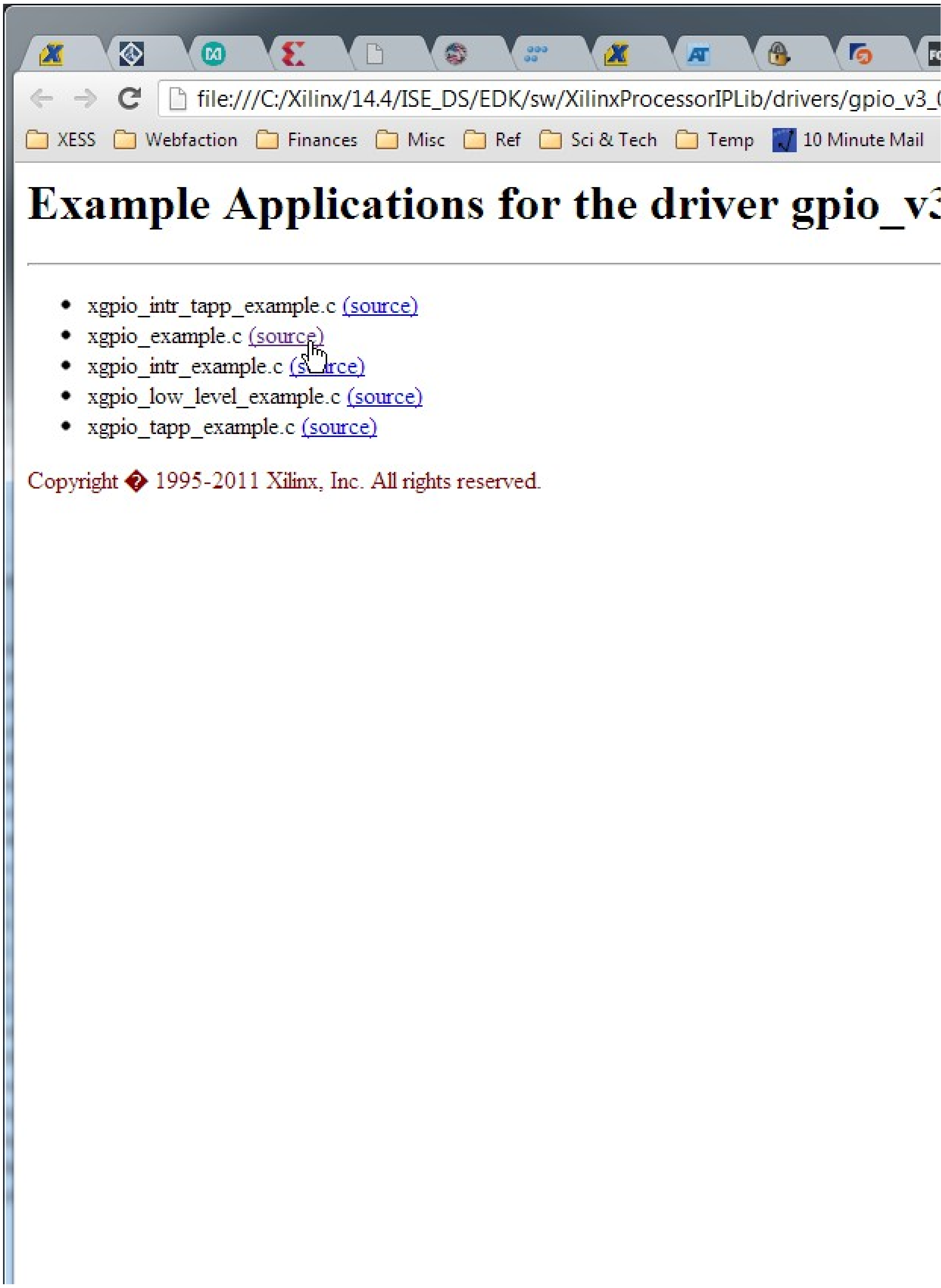
Drivers present in the Board Support Package.

debug_module	uartlite	Documentation	Examples
leds	gpio	Documentation	Examples

C-Build [MicroblazeBlinker]

```
elfcheck passed.
Finished building: MicroblazeBlinker.elf.elfcheck
```

On the webpage that comes up, I select the xgpio_example link since it doesn't use interrupts (which I don't even have in my system) and it's not too low-level (I hope).



file:///C:/Xilinx/14.4/ISE_DS/EDK/sw/XilinxProcessorIPLib/drivers/gpio_v3_0
XESS
Webfaction
Finances
Misc
Ref
Sci & Tech
Temp
10 Minute Mail

Example Applications for the driver gpio_v3

- xgpio_intr_tapp_example.c (source)
- xgpio_example.c (source)
- xgpio_intr_example.c (source)
- xgpio_low_level_example.c (source)
- xgpio_tapp_example.c (source)

Copyright � 1995-2011 Xilinx, Inc. All rights reserved.

The meat of the code in the example is shown below (minus an big licensing header that I'm sure nobody reads). This code actually shows two ways to blink an LED!

```
#include "xparameters.h"

#include "xgpio.h"



/************************** Constant Definitions
*****************************/


#define LED 0x01   /* Assumes bit 0 of GPIO is connected to an LED  */


/*

 * The following constant maps to the name of the hardware instances that

 * were created in the EDK XPS system.

 */

#define GPIO_EXAMPLE_DEVICE_ID  XPAR_LEDS_POSITIONS_DEVICE_ID


/*

 * The following constant is used to wait after an LED is turned on to make

 * sure that it is visible to the human eye.  This constant might need to be

 * tuned for faster or slower processor speeds.

 */

#define LED_DELAY     1000000


/*

 * The following constant is used to determine which channel of the GPIO is

 * used for the LED if there are 2 channels supported.

 */

#define LED_CHANNEL 1


/************************** Variable Definitions
*****************************/


/*

 * The following are declared globally so they are zeroed and so they are

 * easily accessible from a debugger

 */
```

```
XGpio Gpio; /* The Instance of the GPIO Driver */

/
**********************************************************************
******/
/**
*
* The purpose of this function is to illustrate how to use the GPIO level 1
* driver to turn on and off an LED.
*
* @param        None
*
* @return       XST_FAILURE to indicate that the GPIO Intialisation had failed.
*
* @note         This function will not return if the test is running.
*
**********************************************************************
*******/
int main(void)
{
	u32 Data;
	int Status;
	volatile int Delay;

	/*
	 * Initialize the GPIO driver
	 */
	Status = XGpio_Initialize(&Gpio, GPIO_EXAMPLE_DEVICE_ID);
	if (Status != XST_SUCCESS) {
		return XST_FAILURE;
	}

	/*
	 * Set the direction for all signals to be inputs except the
	 * LED output
	 */
	XGpio_SetDataDirection(&Gpio, LED_CHANNEL, ~LED);
```

```
/* Loop forever blinking the LED */

while (1) {
        /*
         * Read the state of the data so that only the LED state can be
         * modified
         */
        Data = XGpio_DiscreteRead(&Gpio, LED_CHANNEL);

        /*
         * Set the LED to the opposite state such that it blinks using
         * the first method, two methods are used for illustration
         * purposes only
         */
        if (Data & LED) {
                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, Data & ~LED);
        } else {
                XGpio_DiscreteWrite(&Gpio, LED_CHANNEL, Data | LED);
        }

        /* Wait a small amount of time so the LED is visible */

        for (Delay = 0; Delay < LED_DELAY; Delay++);

        /*
         * Read the state of the data so that only the LED state can be
         * modified
         */
        Data = XGpio_DiscreteRead(&Gpio, LED_CHANNEL);

        /*
         * Set the LED to the opposite state such that it blinks using
         * the other API functions
         */
        if (Data & LED) {
                XGpio_DiscreteClear(&Gpio, LED_CHANNEL, LED);
        } else {
                XGpio_DiscreteSet(&Gpio, LED_CHANNEL, LED);
            }

            /* Wait a small amount of time so the LED is visible */

            for (Delay = 0; Delay < LED_DELAY; Delay++);
        }

        return XST_SUCCESS;
}
```

I extracted the code from the example, transferred it to the helloworld.c file, and then cut it down and simplified the main loop until it looks like this:

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

#include "xparameters.h"
#include "xgpio.h"


/************************** Constant Definitions
*****************************/

#define LED 0x01   /* Assumes bit 0 of GPIO is connected to an LED
*/

/*
 * The following constant maps to the name of the hardware instances
that
 * were created in the EDK XPS system.
 */
#define GPIO_EXAMPLE_DEVICE_ID XPAR_LEDS_POSITIONS_DEVICE_ID

/*
 * The following constant is used to wait after an LED is turned on
to make
 * sure that it is visible to the human eye.  This constant might
need to be
 * tuned for faster or slower processor speeds.
 */
#define LED_DELAY     1000000

/*
 * The following constant is used to determine which channel of the
GPIO is
 * used for the LED if there are 2 channels supported.
 */
#define LED_CHANNEL 1

/************************** Variable Definitions
*****************************/

/*
 * The following are declared globally so they are zeroed and so
they are
 * easily accessible from a debugger
 */

XGpio Gpio; /* The Instance of the GPIO Driver */

/
**************************************************************************
******/
/**
*
* The purpose of this function is to illustrate how to use the GPIO level 1
* driver to turn on and off an LED.
*
* @param       None
*
* @return      XST_FAILURE to indicate that the GPIO Intialisation had failed.
*
* @note        This function will not return if the test is running.
*
**************************************************************************
*******/
int main( void)
{
```

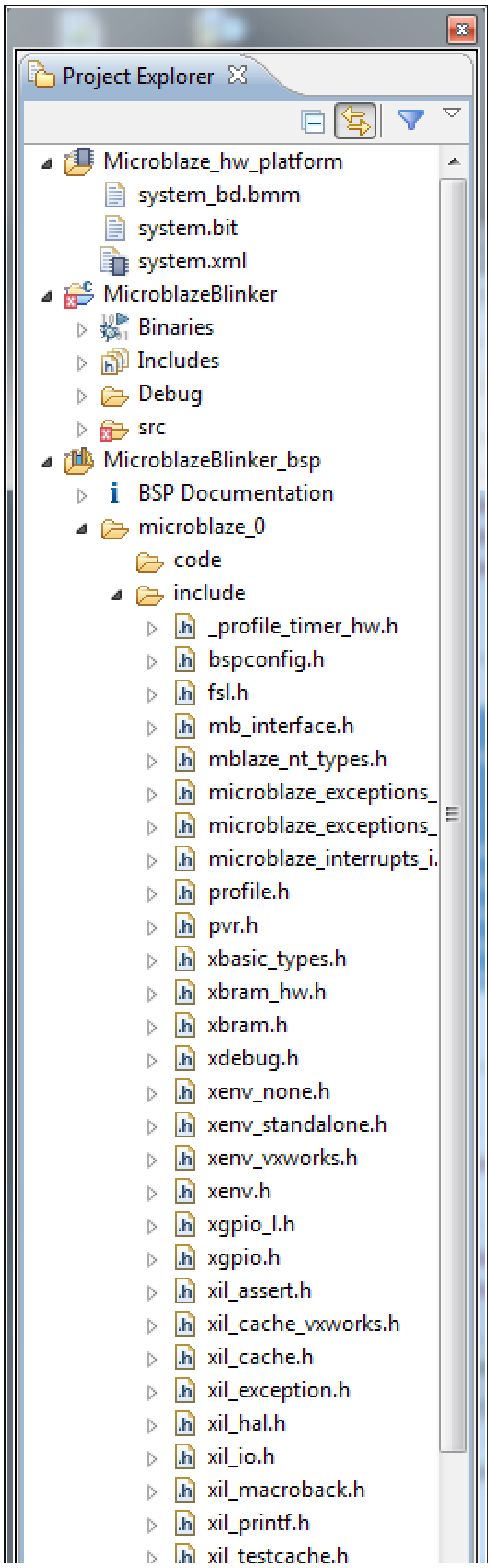
```
u32 Data;
int Status;
volatile int Delay;

/*
 * Initialize the GPIO driver
 */
Status = XGpio_Initialize(&Gpio, GPIO_EXAMPLE_DEVICE_ID);
if (Status != XST_SUCCESS) {
    return XST_FAILURE;
}

/*
 * Set the direction for all signals to be inputs except the
 * LED output
 */
XGpio_SetDataDirection(&Gpio, LED_CHANNEL, ~LED);

/* Loop forever blinking the LED */
while (1) {

      /* Turn off the LED by clearing the bit. */
    XGpio_DiscreteClear(&Gpio, LED_CHANNEL, LED);

    /* Wait a small amount of time so the LED is visible */
    for (Delay = 0; Delay < LED_DELAY; Delay++);

      /* Turn on the LED by setting the bit. */
    XGpio_DiscreteSet(&Gpio, LED_CHANNEL, LED);

    /* Wait a small amount of time so the LED is visible */
    for (Delay = 0; Delay < LED_DELAY; Delay++);
}

return XST_SUCCESS;
```

```
}
```

As soon as I saved the modified helloworld.c file, the C compiler kicked off and flagged this error:

```
../src/helloworld.c:84:42: error: 'XPAR_LEDS_POSITIONS_DEVICE_ID' undeclared (first use in this function)
```

So obviously I'm missing some definition in a header file. Since it's related to the LED driver, possibly it's in the BSP. Expanding the include folder in the BSP, I find a file called "xparameters.h" That seems like a likely candidate for what I need.



```
◢ Microblaze_hw_platform
      system_bd.bmm
      system.bit
      system.xml
◢ MicroblazeBlinker
   ▷ Binaries
   ▷ Includes
   ▷ Debug
   ▷ src
◢ MicroblazeBlinker_bsp
   ▷ BSP Documentation
   ◢ microblaze_0
         code
      ◢ include
         ▷ _profile_timer_hw.h
         ▷ bspconfig.h
         ▷ fsl.h
         ▷ mb_interface.h
         ▷ mblaze_nt_types.h
         ▷ microblaze_exceptions_
         ▷ microblaze_exceptions_
         ▷ microblaze_interrupts_i.
         ▷ profile.h
         ▷ pvr.h
         ▷ xbasic_types.h
         ▷ xbram_hw.h
         ▷ xbram.h
         ▷ xdebug.h
         ▷ xenv_none.h
         ▷ xenv_standalone.h
         ▷ xenv_vxworks.h
         ▷ xenv.h
         ▷ xgpio_l.h
         ▷ xgpio.h
         ▷ xil_assert.h
         ▷ xil_cache_vxworks.h
         ▷ xil_cache.h
         ▷ xil_exception.h
         ▷ xil_hal.h
         ▷ xil_io.h
         ▷ xil_macroback.h
         ▷ xil_printf.h
         ▷ xil_testcache.h
```

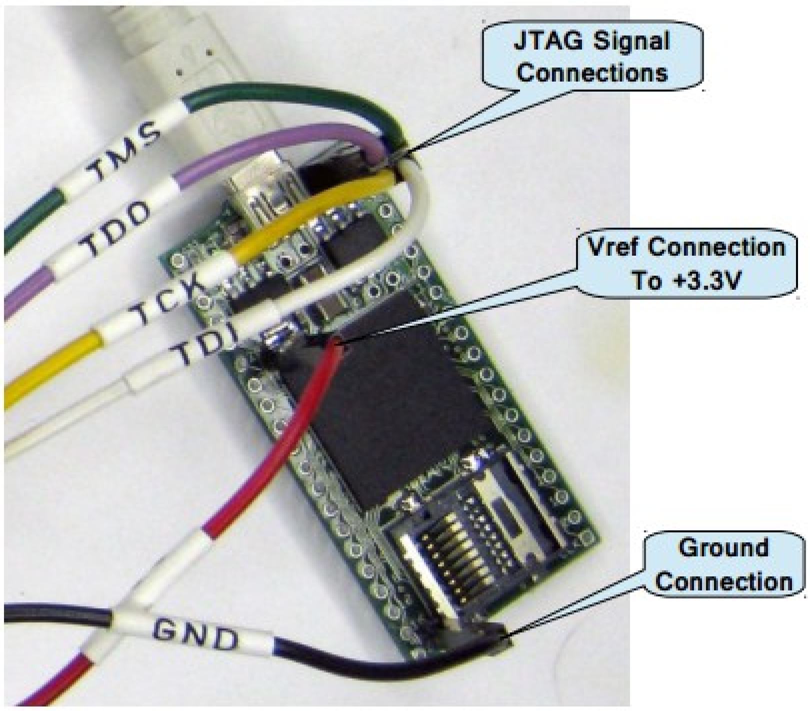
Opening the xparameters.h file, I find the following definitions inside:

```
/* Definitions for driver GPIO */
#define XPAR_XGPIO_NUM_INSTANCES 1


/* Definitions for peripheral LEDS */
#define XPAR_LEDS_BASEADDR 0x40000000
#define XPAR_LEDS_HIGHADDR 0x4000FFFF
#define XPAR_LEDS_DEVICE_ID 0
#define XPAR_LEDS_INTERRUPT_PRESENT 0
#define XPAR_LEDS_IS_DUAL 0
```

Going back to the helloworld.c file, I substitute `XPAR_LEDS_DEVICE_ID` in place of XPAR_LEDS_POSITIONS_DEVICE_ID. After saving the modified file, the project compiles successfully and an ELF file is produced.

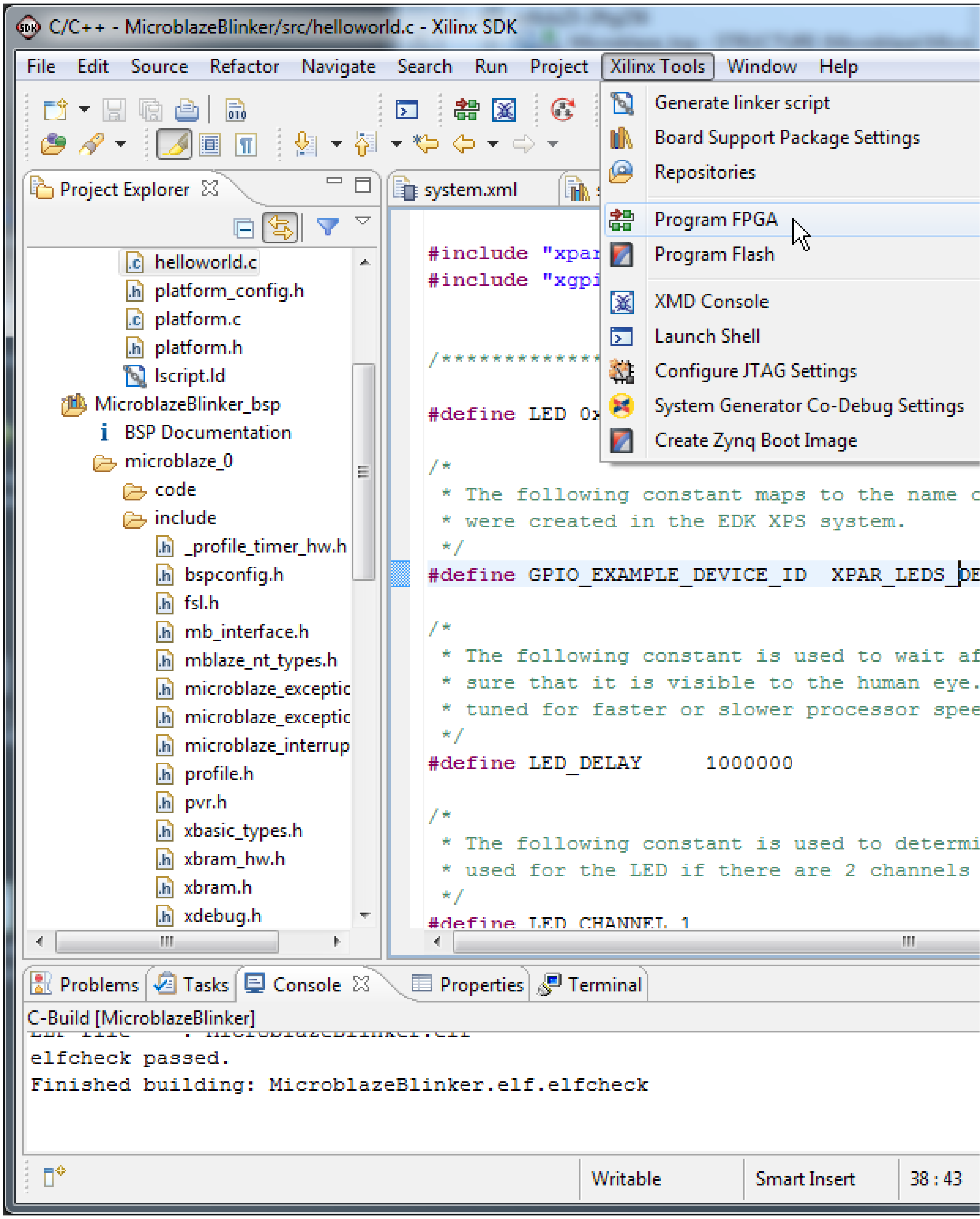
At this point, I setup a XuLA2 board to run the LED blinker project as follows:

1. I connect the XuLA2 to a USB port and use the GXSFLAGS utility to enable the auxiliary JTAG port.
2. I connect the flying leads of a Xilinx USB cable to the XuLA2 board as follows:



3. I connect an LED + resistor from Channel 0 of the XuLA2 prototyping header to ground.

Once the hardware is ready, I click on the "Xilinx Tools => Program FPGA" menu item.



```
#include "xpar
#include "xgpi

/*************

#define LED 0x

/*
 * The following constant maps to the name c
 * were created in the EDK XPS system.
 */
#define GPIO_EXAMPLE_DEVICE_ID  XPAR_LEDS_DE

/*
 * The following constant is used to wait af
 * sure that it is visible to the human eye.
 * tuned for faster or slower processor spee
 */
#define LED_DELAY     1000000

/*
 * The following constant is used to determi
 * used for the LED if there are 2 channels
 */
#define LED_CHANNEL 1
```

```
elfcheck passed.
Finished building: MicroblazeBlinker.elf.elfcheck
```

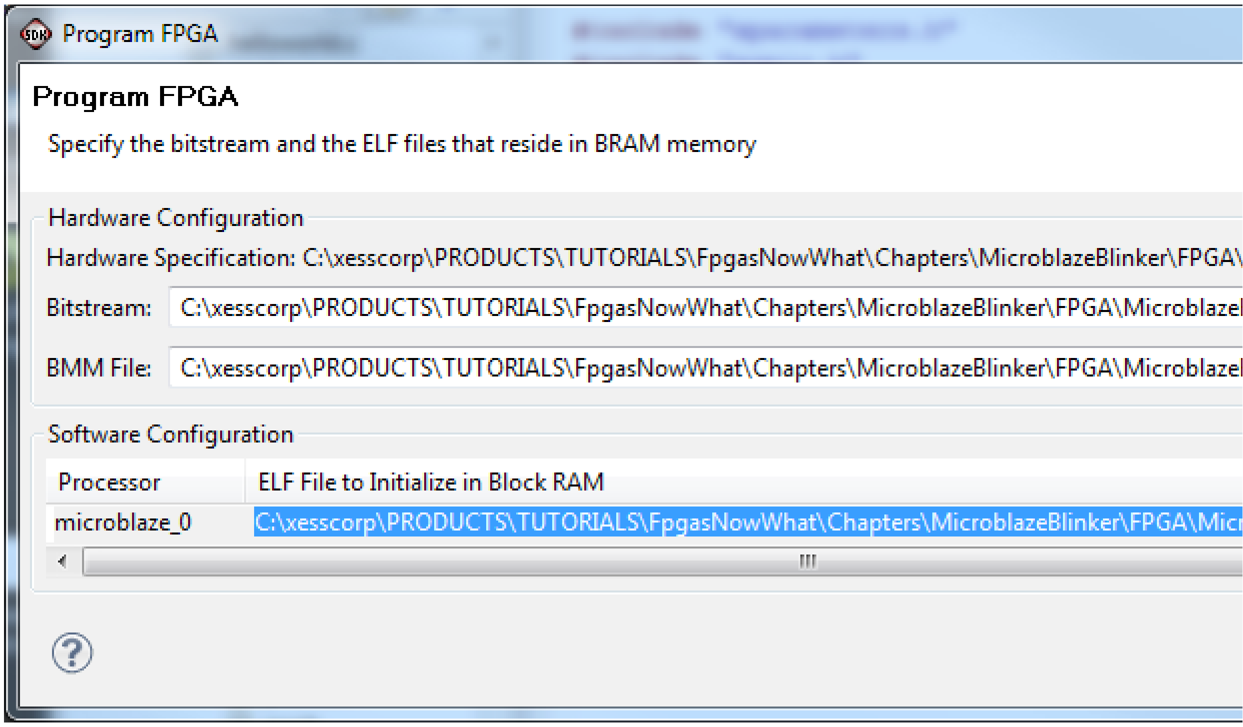
The Program FPGA window appears. There are three files that are needed to program the FPGA (and which should be automatically filled-in when the window appears):

1. The FPGA bitstream file for the unprogrammed Microblaze hardware (system.bit).

2. The block memory map (BMM) file that shows where the contents of the block RAMs are stored in the FPGA bitstream.
3. The ELF file containing the compiled machine code for the helloworld.c project.

After specifying these files, clicking the Program button sets in motion the following sequence of events:

1. The binary code in the ELF file is inserted into the FPGA bitstream under the direction of the BMM file.
2. The Xilinx USB cable is detected.
3. The combined FPGA bitstream and ELF file are downloaded into the FPGA on the XuLA2 board through the USB cable.


Program FPGA

Program FPGA

Specify the bitstream and the ELF files that reside in BRAM memory

Hardware Configuration

Hardware Specification: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\

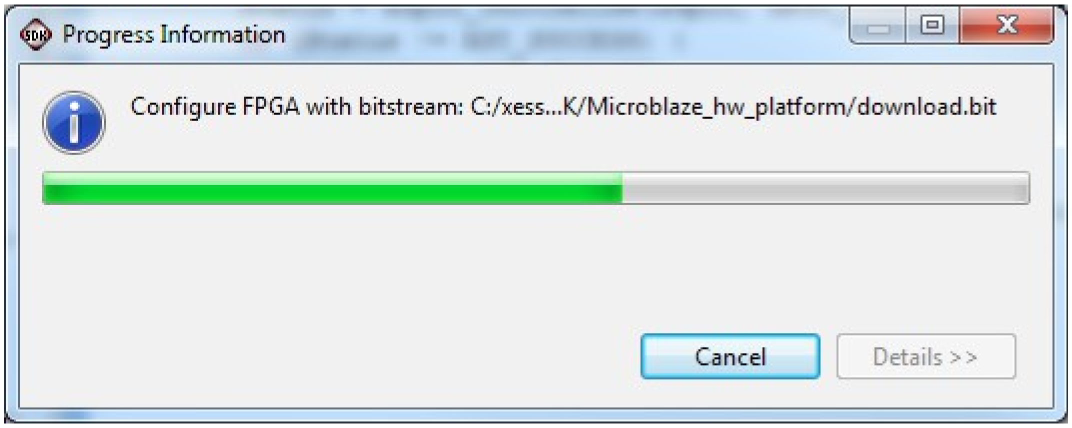
Bitstream: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\Microblaze

BMM File: C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\Microblaze

Software Configuration

Processor	ELF File to Initialize in Block RAM
microblaze_0	C:\xesscorp\PRODUCTS\TUTORIALS\FpgasNowWhat\Chapters\MicroblazeBlinker\FPGA\Mic

The download process should take a few seconds:


Progress Information

Configure FPGA with bitstream: C:/xess...K/Microblaze_hw_platform/download.bit

After the download completes, the LED should start blinking several times a second. At least it does for me. Hopefully you will get the same result!