



DesignNews

Field-Programmable Gate Array (FPGA) Primer

Day 5:

Hardware and Software Design with Vivado and Vitis

Sponsored by



Webinar Logistics

- Turn on your system sound to hear the streaming presentation.
- If you have technical problems, click “Help” or submit a question asking for assistance.
- Participate in ‘Attendee Chat’ by maximizing the chat widget in your dock.

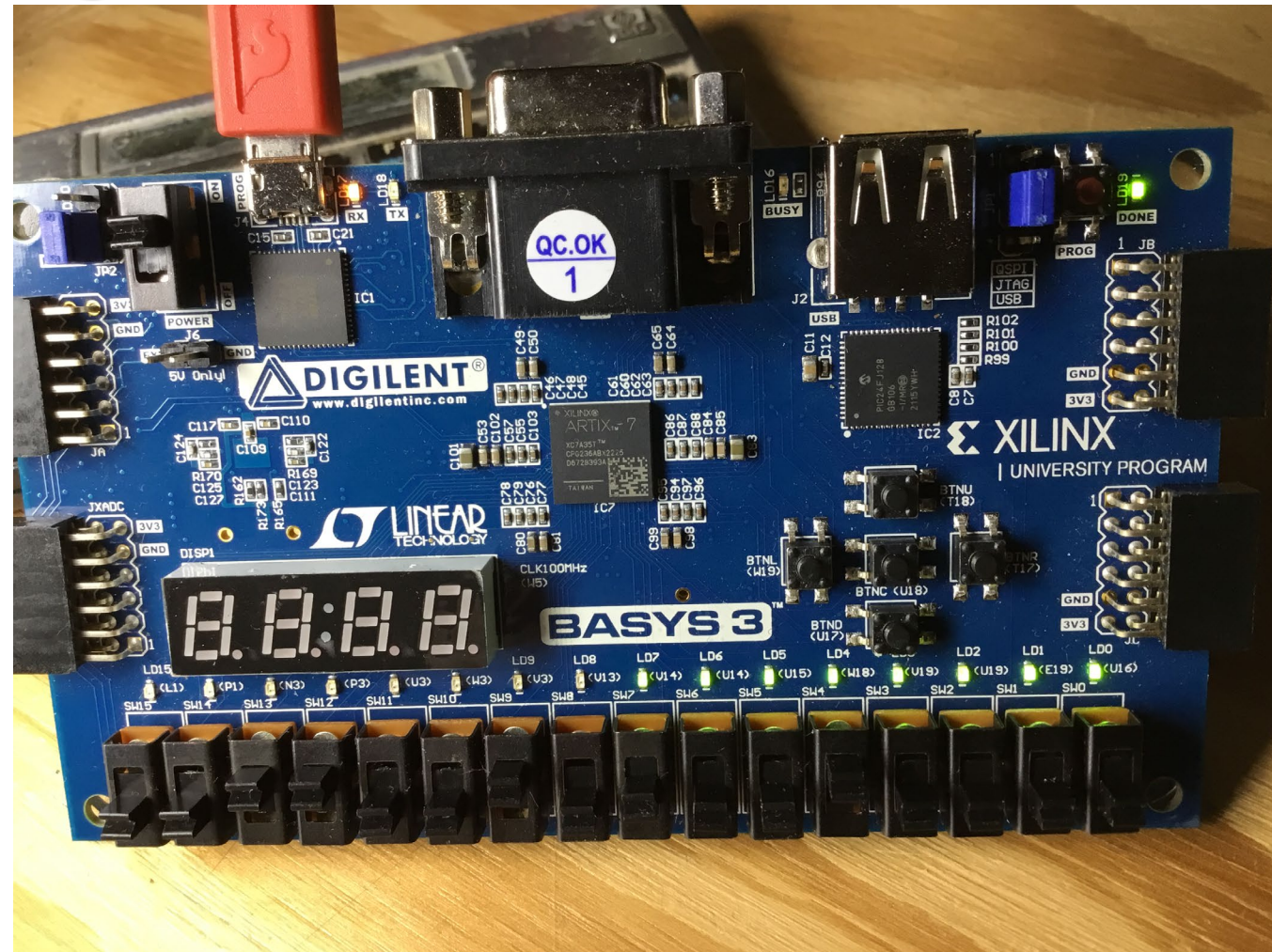


Fred Eady

Visit 'Lecturer Profile' in your console for more details.

AGENDA

- **Create a Block Design with Vivado**
- **Finish the Design with Vitis**



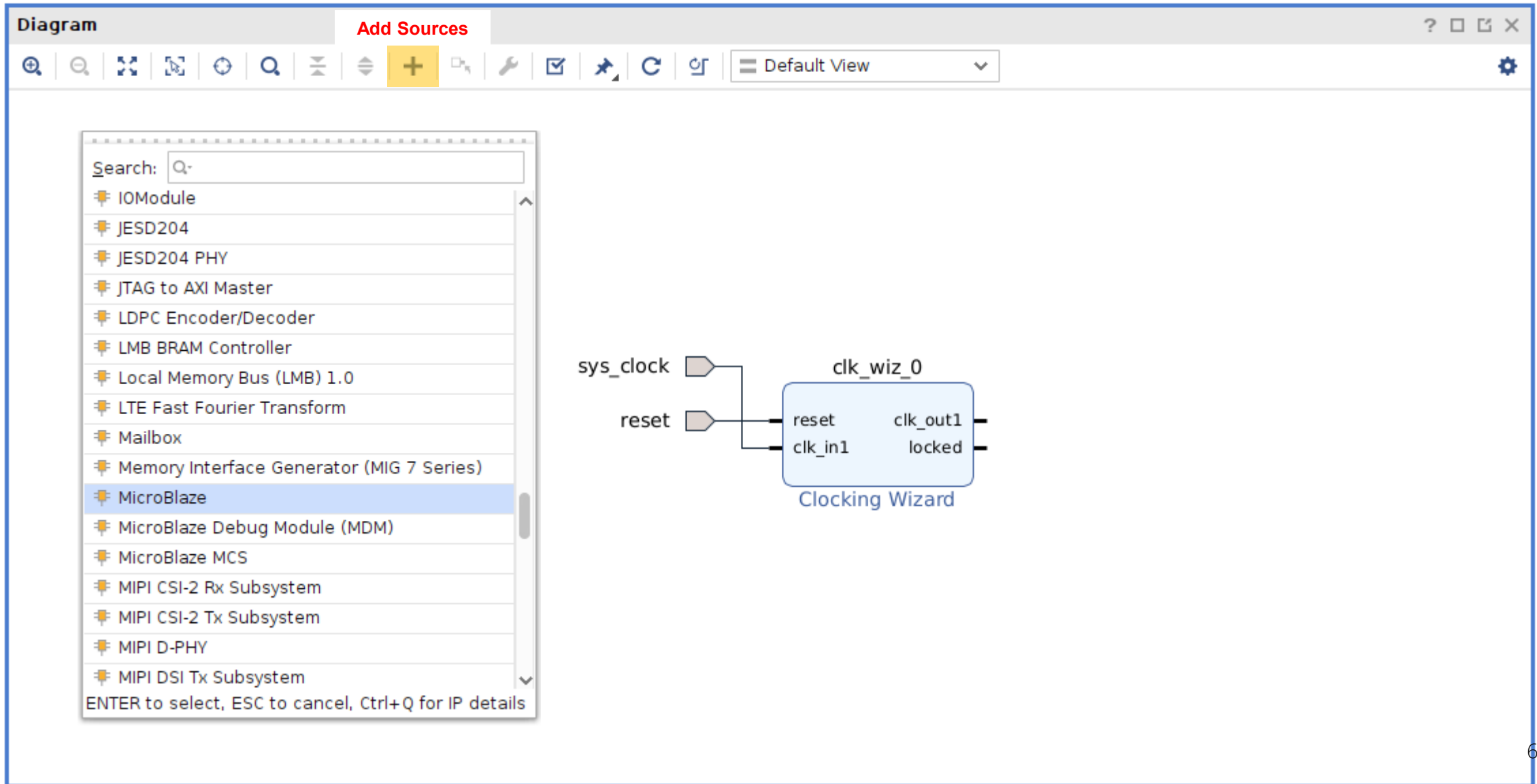
Create Block Design and Add System Clock

The screenshot shows the Vivado 2023.1 interface for creating a block design. The 'Flow Navigator' on the left shows the 'IP INTEGRATOR' section with 'Create Block Design' selected. The 'Sources' tab is active, showing a tree view of components including 'System Clock'. A context menu is open over 'System Clock', with 'Connect Board Component...' selected. A blue arrow points from this menu to the 'Connect Board Component' dialog box on the right. The dialog box prompts the user to 'Select one or more pins to connect board component 'System Clock''. It contains a table with the following data:

Name	VLNV
Create new IP	
+ Clocking Wizard	xilinx.com:ip:clk_wiz:6.0
<input checked="" type="checkbox"/> clock_CLK_IN1	
<input type="checkbox"/> clock_CLK_IN2	

At the bottom of the dialog box are 'OK' and 'Cancel' buttons.

Add MicroBlaze



The screenshot shows the Vivado IP Catalog window with the 'Add Sources' tab selected. The search bar is empty, and the list of IP blocks is displayed. The 'MicroBlaze' IP block is highlighted. To the right, a block diagram shows the 'clk_wiz_0' (Clocking Wizard) block. The 'sys_clock' and 'reset' signals are connected to the 'reset' and 'clk_in1' inputs of the wizard. The 'clk_out1' and 'locked' signals are shown as outputs of the wizard.

Diagram

Add Sources

Search: Q-

- IOModule
- JESD204
- JESD204 PHY
- JTAG to AXI Master
- LDPC Encoder/Decoder
- LMB BRAM Controller
- Local Memory Bus (LMB) 1.0
- LTE Fast Fourier Transform
- Mailbox
- Memory Interface Generator (MIG 7 Series)
- MicroBlaze**
- MicroBlaze Debug Module (MDM)
- MicroBlaze MCS
- MIPI CSI-2 Rx Subsystem
- MIPI CSI-2 Tx Subsystem
- MIPI D-PHY
- MIPI DSI Tx Subsystem

ENTER to select, ESC to cancel, Ctrl+Q for IP details

sys_clock

reset

clk_wiz_0

reset

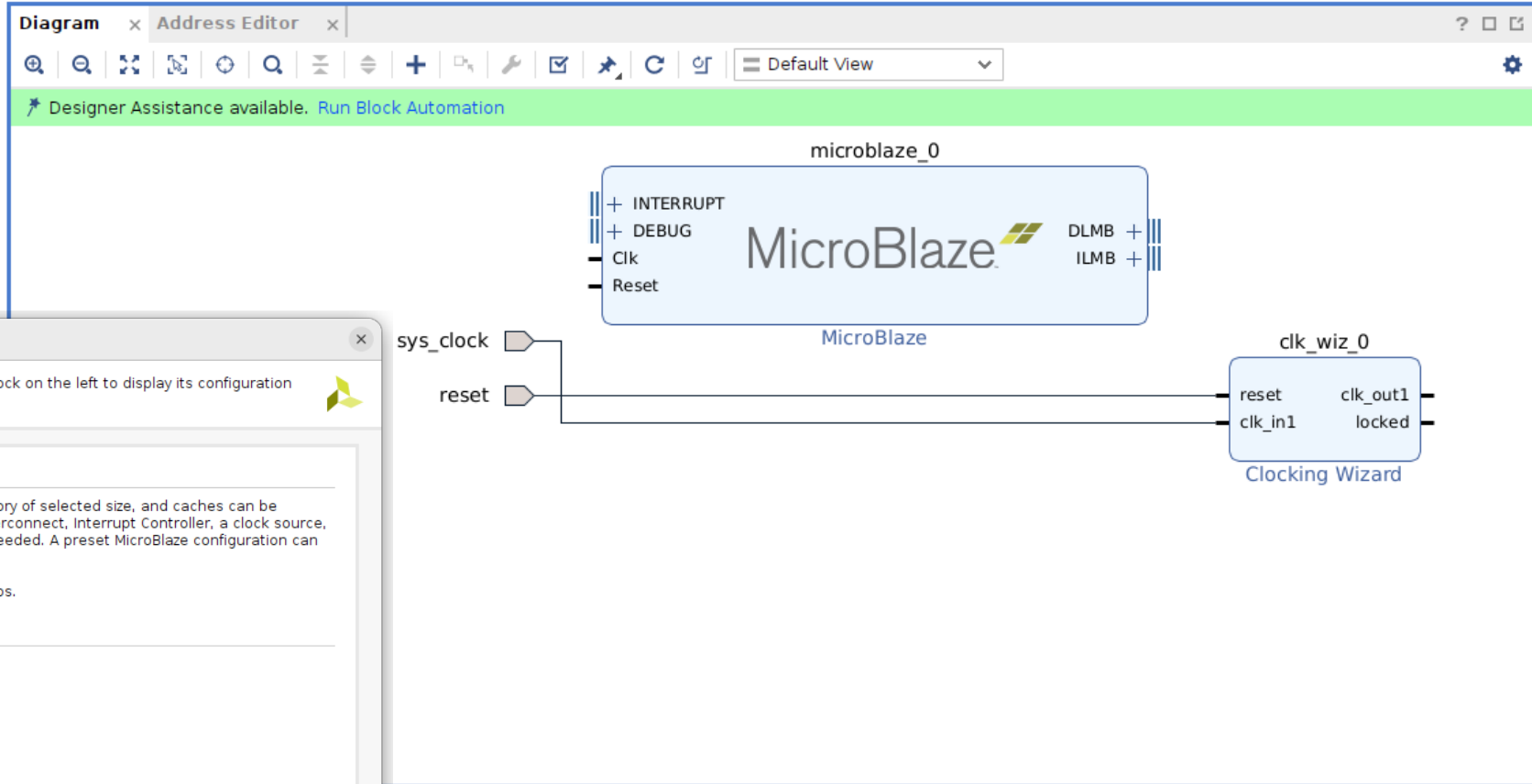
clk_in1

clk_out1

locked

Clocking Wizard

Specify 32KB of Local Memory



Run Block Automation

Automatically make connections in your design by checking the boxes of the blocks to connect. Select a block on the left to display its configuration options on the right.

All Automation (1 out of 1 selected)
 microblaze_0

Description

MicroBlaze connection automation generates local memory of selected size, and caches can be configured. MicroBlaze Debug Module, Peripheral AXI Interconnect, Interrupt Controller, a clock source, Processor System Reset are added and connected as needed. A preset MicroBlaze configuration can also be selected.

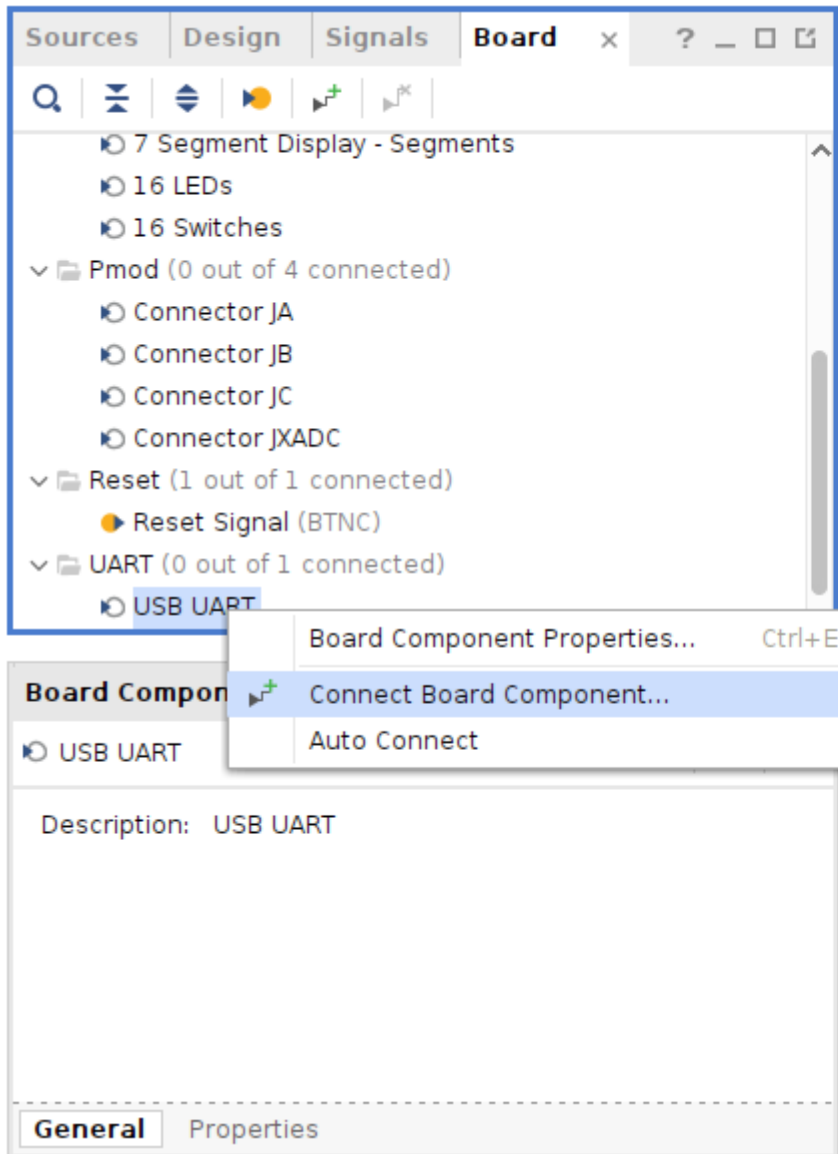
Information about the options can be found in the tooltips.

Options

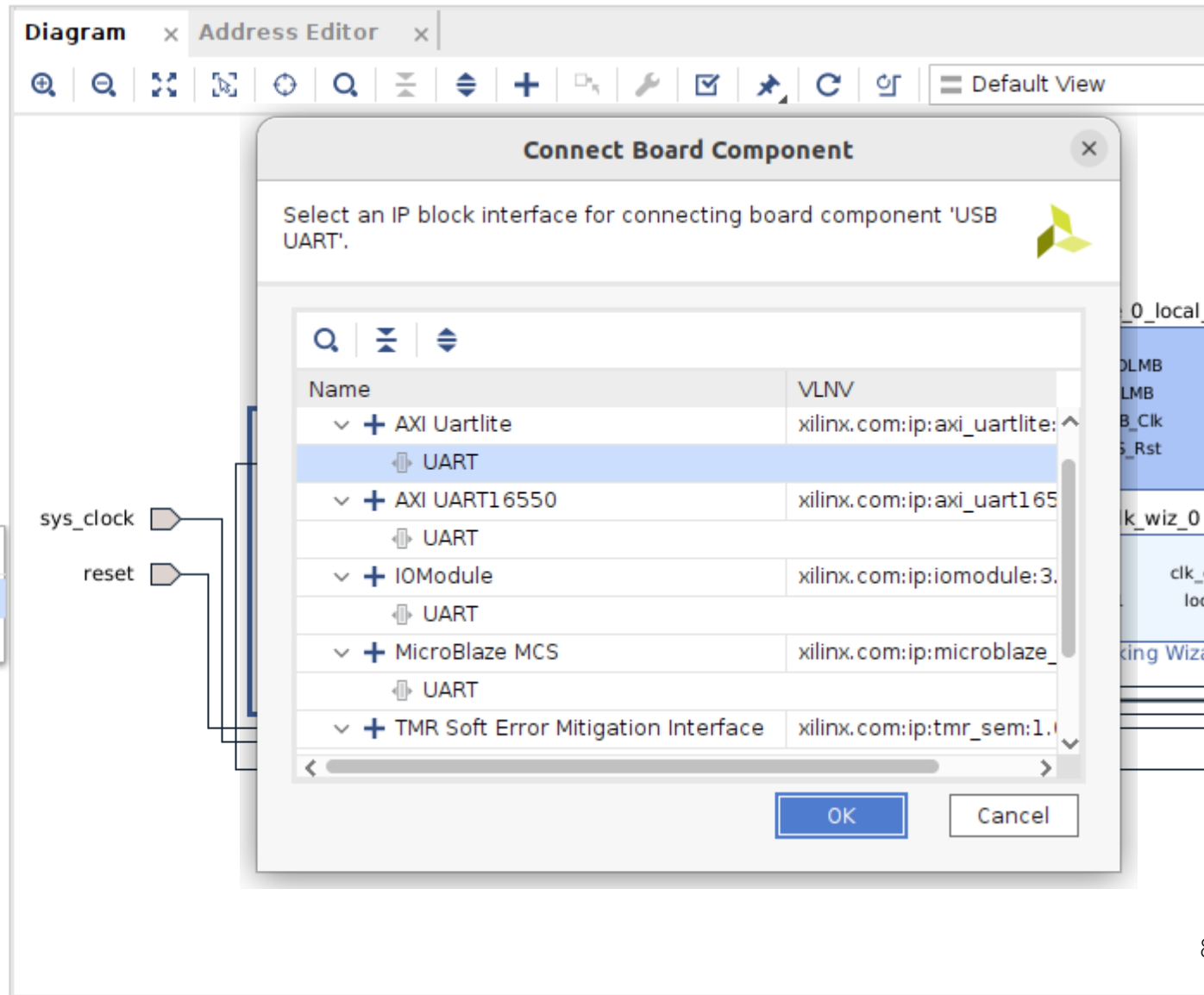
Preset	None
Local Memory	32KB
Local Memory ECC	None
Cache Configuration	None
Debug Module	Debug Only
Peripheral AXI Port	Enabled
<input type="checkbox"/> Interrupt Controller	
Clock Connection	/clk_wiz_0/clk_out1 (100 MHz)

OK Cancel

Add USB UART



The screenshot shows the Board Component Manager in Vivado. The 'Board' tab is active, displaying a tree view of components. The 'USB UART' component is selected. A context menu is open over the 'USB UART' component, showing options: 'Board Component Properties...' (Ctrl+E), 'Connect Board Component...' (highlighted), and 'Auto Connect'. Below the tree view, the 'Board Component' section shows 'USB UART' with a description: 'Description: USB UART'. The 'General' tab is selected in the Properties section.

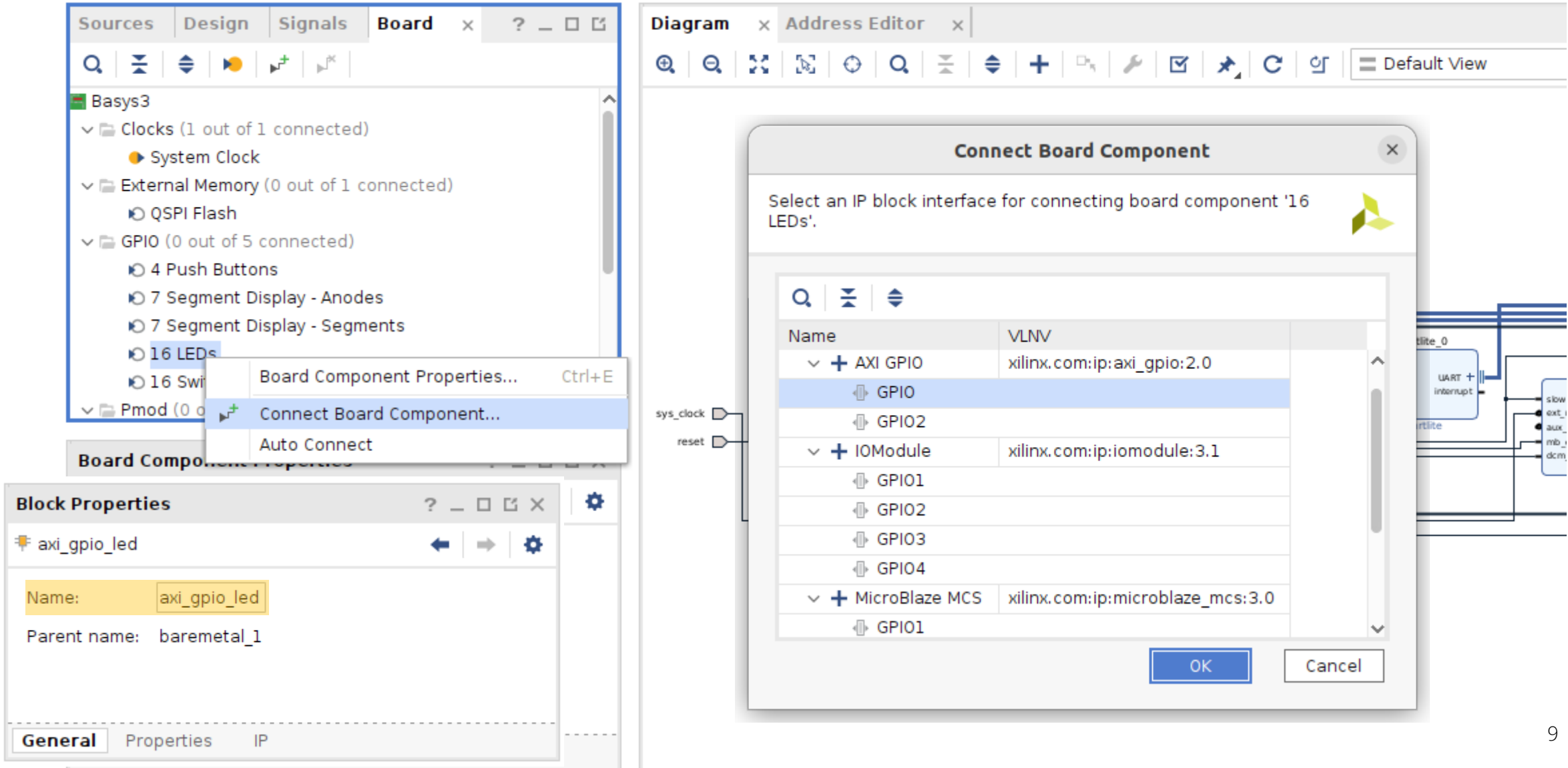


The screenshot shows the 'Connect Board Component' dialog box in Vivado. The dialog prompts the user to 'Select an IP block interface for connecting board component 'USB UART''. A list of IP blocks is displayed, with 'UART' selected. The list includes:

Name	VLNV
+ AXI Uartlite	xilinx.com:ip:axi_uartlite:1.0
UART	xilinx.com:ip:uart:1.0
+ AXI UART16550	xilinx.com:ip:axi_uart16550:1.0
UART	xilinx.com:ip:uart:1.0
+ IOModule	xilinx.com:ip:iomodule:3.0
UART	xilinx.com:ip:uart:1.0
+ MicroBlaze MCS	xilinx.com:ip:microblaze_mcs:1.0
UART	xilinx.com:ip:uart:1.0
+ TMR Soft Error Mitigation Interface	xilinx.com:ip:tmr_sem:1.0

The 'OK' button is highlighted. In the background, a diagram shows 'sys_clock' and 'reset' signals connected to the board component.

Add Bank of 16 LEDs – Rename LED Block



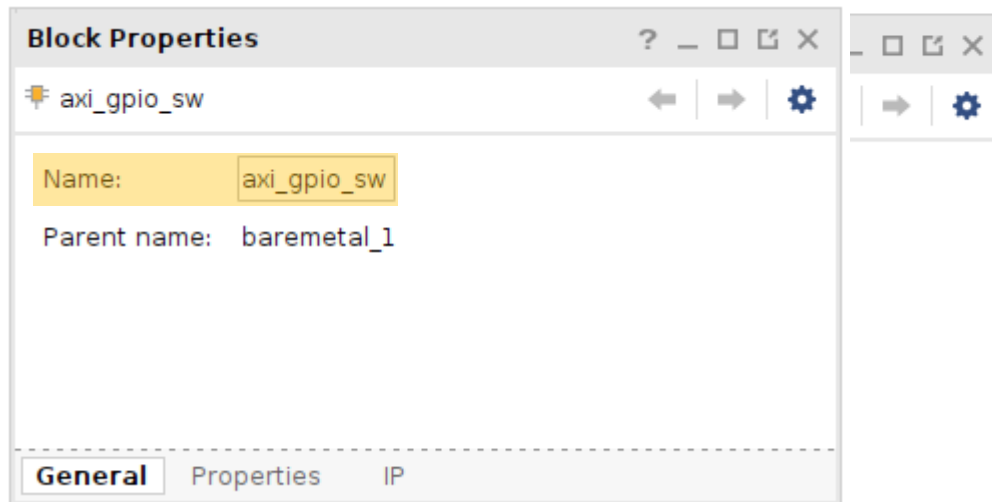
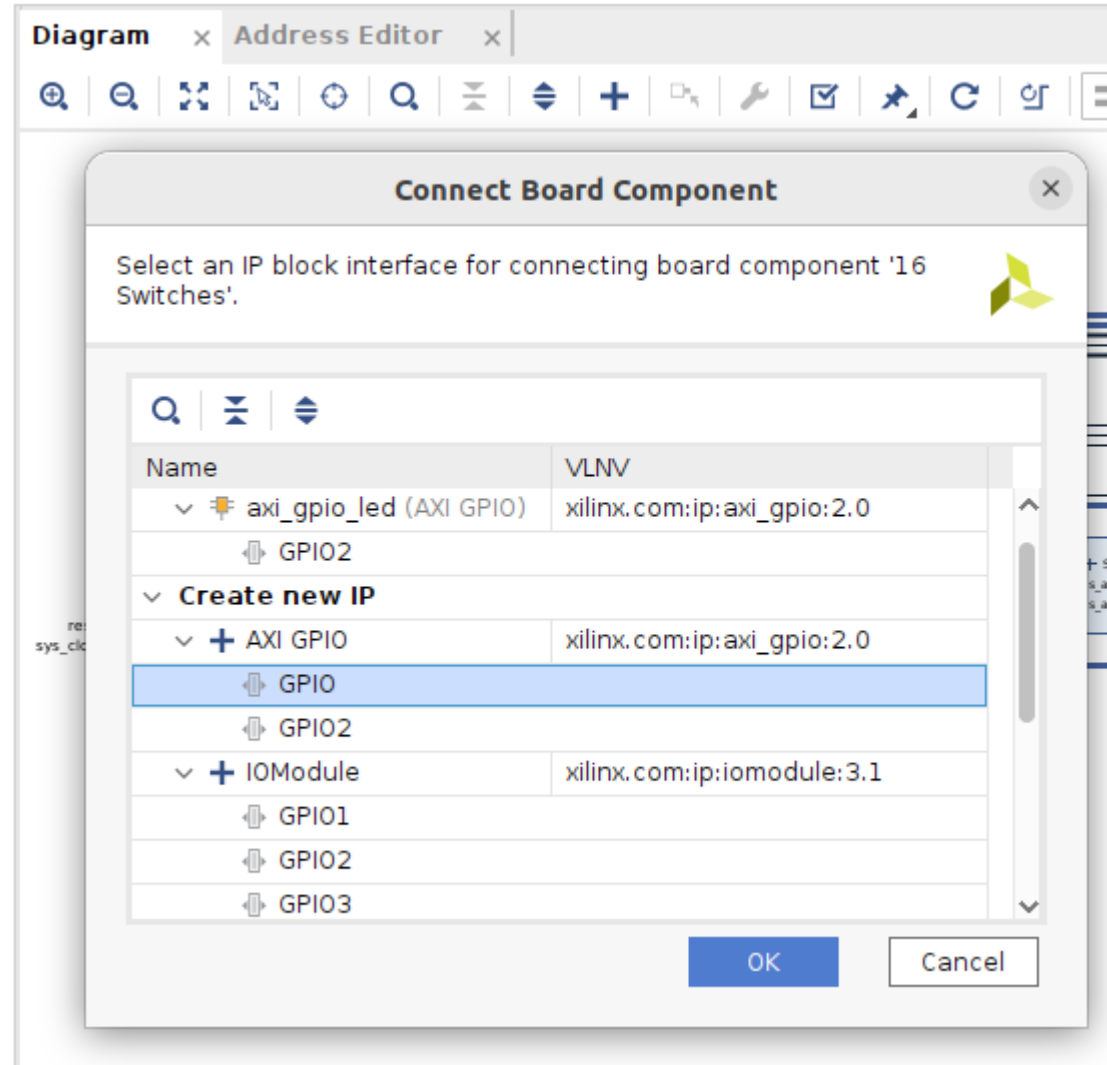
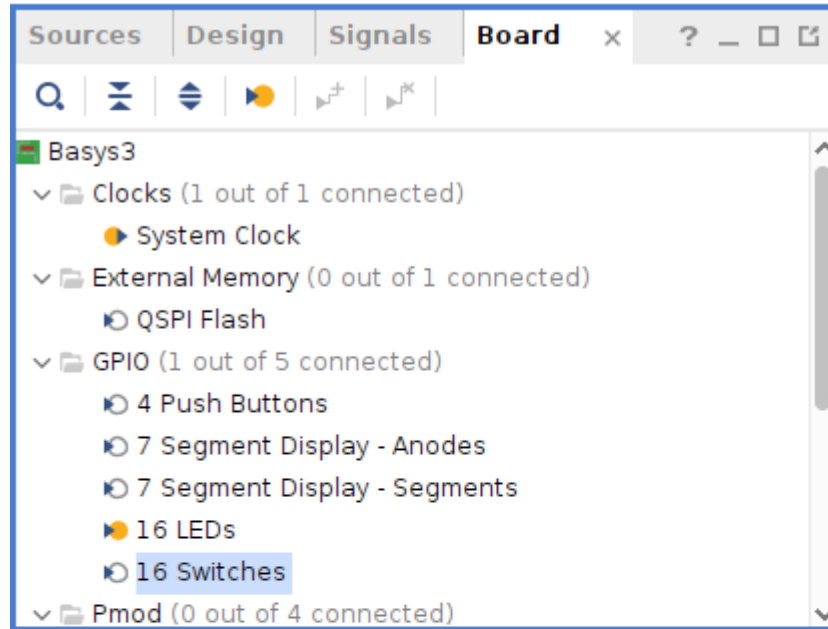
The screenshot displays the Vivado IDE interface with several windows open:

- Sources**: Shows the project hierarchy for 'Basys3'. Under 'GPIO (0 out of 5 connected)', the '16 LEDs' component is selected. A context menu is open over it, with 'Connect Board Component...' highlighted.
- Block Properties**: Shows the properties for the selected 'axi_gpio_led' block. The 'Name' field is highlighted and contains the text 'axi_gpio_led'. The 'Parent name' is 'baremetal_1'.
- Diagram**: Shows a partial circuit diagram with 'sys_clock' and 'reset' inputs.
- Connect Board Component**: A dialog box is open, prompting the user to 'Select an IP block interface for connecting board component '16 LEDs''. The dialog contains a search bar and a list of IP blocks:

Name	VLNV
AXI GPIO	xilinx.com:ip:axi_gpio:2.0
GPIO	
GPIO2	
IOModule	xilinx.com:ip:iomodule:3.1
GPIO1	
GPIO2	
GPIO3	
GPIO4	
MicroBlaze MCS	xilinx.com:ip:microblaze_mcs:3.0
GPIO1	

The 'GPIO' block is currently selected in the list. The dialog has 'OK' and 'Cancel' buttons at the bottom.

Add Bank of 16 Switches – Rename Switch Block



Add Basys 3 Constraint File

Sources x Design Signals Board ? - □ ↗

🔍 | ⚙️ | ⚙️ | + | ? | ● 0 | ⚙️

- Design Sources (1)
 - baremetal_1 (baremetal_1.bd) (12)
- Constraints (1)
 - constrs_1 (1)
 - Basys-3-Master.xdc
- Simulation Sources (1)
- Utility Sources

Hierarchy IP Sources Libraries Compile Order

Source File Properties ? - □ ↗

Basys-3-Master.xdc ⏪ ⏩ ⚙️

Enabled

Location: /home/fred/baremetal_1

Type: XDC ...

Size: 10.1 KB

Modified: Thursday 08/31/23 03:59:32 PM

General Properties

Diagram x Address Editor x Basys-3-Master.xdc * x

/home/fred/baremetal_1/Basys-3-Master.xdc

🔍 | 📄 | ⏪ | ⏩ | ✂️ | 📄 | ✖️ | // | 📄 | 💡

```

6  ## Clock signal
7  set_property -dict { PACKAGE_PIN W5   IOSTANDARD LVCMOS33 } [get_ports clk]
8  create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5} [get_ports clk]
9
10 ## LEDs
11 set_property -dict { PACKAGE_PIN U16  IOSTANDARD LVCMOS33 } [get_ports {led[0]}]
12 set_property -dict { PACKAGE_PIN E19  IOSTANDARD LVCMOS33 } [get_ports {led[1]}]
13 set_property -dict { PACKAGE_PIN U19  IOSTANDARD LVCMOS33 } [get_ports {led[2]}]
14 set_property -dict { PACKAGE_PIN V19  IOSTANDARD LVCMOS33 } [get_ports {led[3]}]
15 set_property -dict { PACKAGE_PIN W18  IOSTANDARD LVCMOS33 } [get_ports {led[4]}]
16 set_property -dict { PACKAGE_PIN U15  IOSTANDARD LVCMOS33 } [get_ports {led[5]}]
17 set_property -dict { PACKAGE_PIN U14  IOSTANDARD LVCMOS33 } [get_ports {led[6]}]
18 set_property -dict { PACKAGE_PIN V14  IOSTANDARD LVCMOS33 } [get_ports {led[7]}]
19 set_property -dict { PACKAGE_PIN V13  IOSTANDARD LVCMOS33 } [get_ports {led[8]}]
20 set_property -dict { PACKAGE_PIN V3   IOSTANDARD LVCMOS33 } [get_ports {led[9]}]
21 set_property -dict { PACKAGE_PIN W3   IOSTANDARD LVCMOS33 } [get_ports {led[10]}]
22 set_property -dict { PACKAGE_PIN U3   IOSTANDARD LVCMOS33 } [get_ports {led[11]}]
23 set_property -dict { PACKAGE_PIN P3   IOSTANDARD LVCMOS33 } [get_ports {led[12]}]
24 set_property -dict { PACKAGE_PIN N3   IOSTANDARD LVCMOS33 } [get_ports {led[13]}]
25 set_property -dict { PACKAGE_PIN P1   IOSTANDARD LVCMOS33 } [get_ports {led[14]}]
26 set_property -dict { PACKAGE_PIN L1   IOSTANDARD LVCMOS33 } [get_ports {led[15]}]
27
28
29 ##7 Segment Display
30 #set_property -dict { PACKAGE_PIN W7   IOSTANDARD LVCMOS33 } [get_ports {seg[0]}]
31 #set_property -dict { PACKAGE_PIN W6   IOSTANDARD LVCMOS33 } [get_ports {seg[1]}]
32 #set_property -dict { PACKAGE_PIN U8   IOSTANDARD LVCMOS33 } [get_ports {seg[2]}]
33 #set_property -dict { PACKAGE_PIN V8   IOSTANDARD LVCMOS33 } [get_ports {seg[3]}]
34 #set_property -dict { PACKAGE_PIN U5   IOSTANDARD LVCMOS33 } [get_ports {seg[4]}]
35 #set_property -dict { PACKAGE_PIN V5   IOSTANDARD LVCMOS33 } [get_ports {seg[5]}]
36 #set_property -dict { PACKAGE_PIN U7   IOSTANDARD LVCMOS33 } [get_ports {seg[6]}]
  
```

11

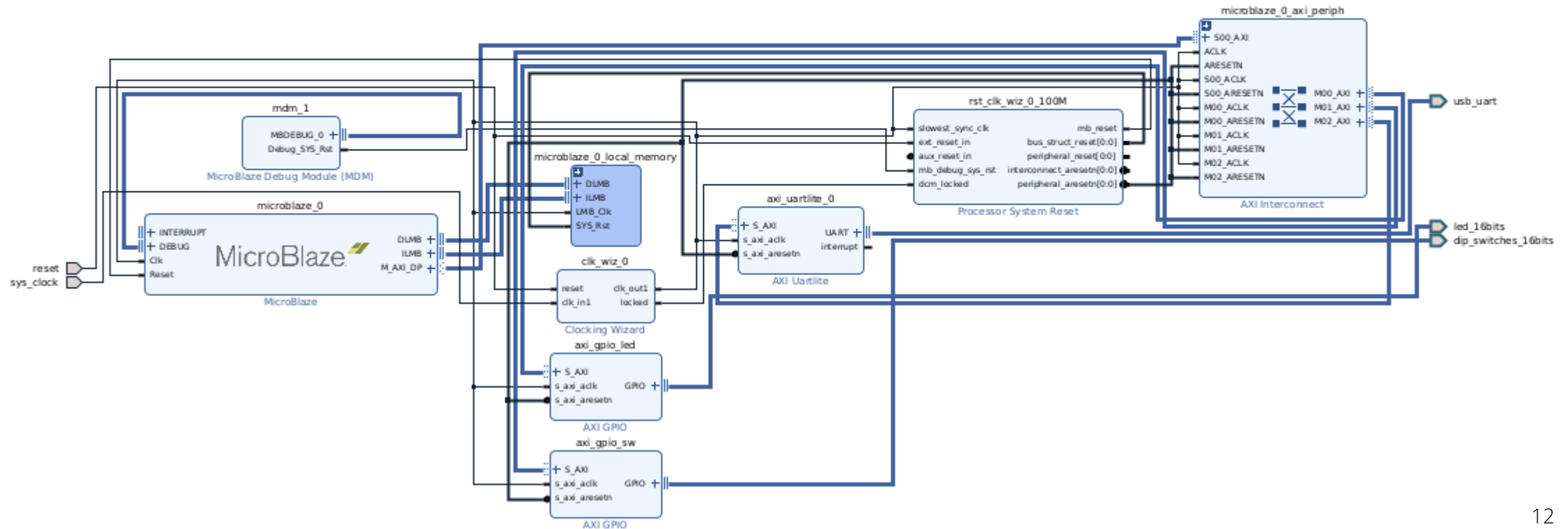
Validate the Design



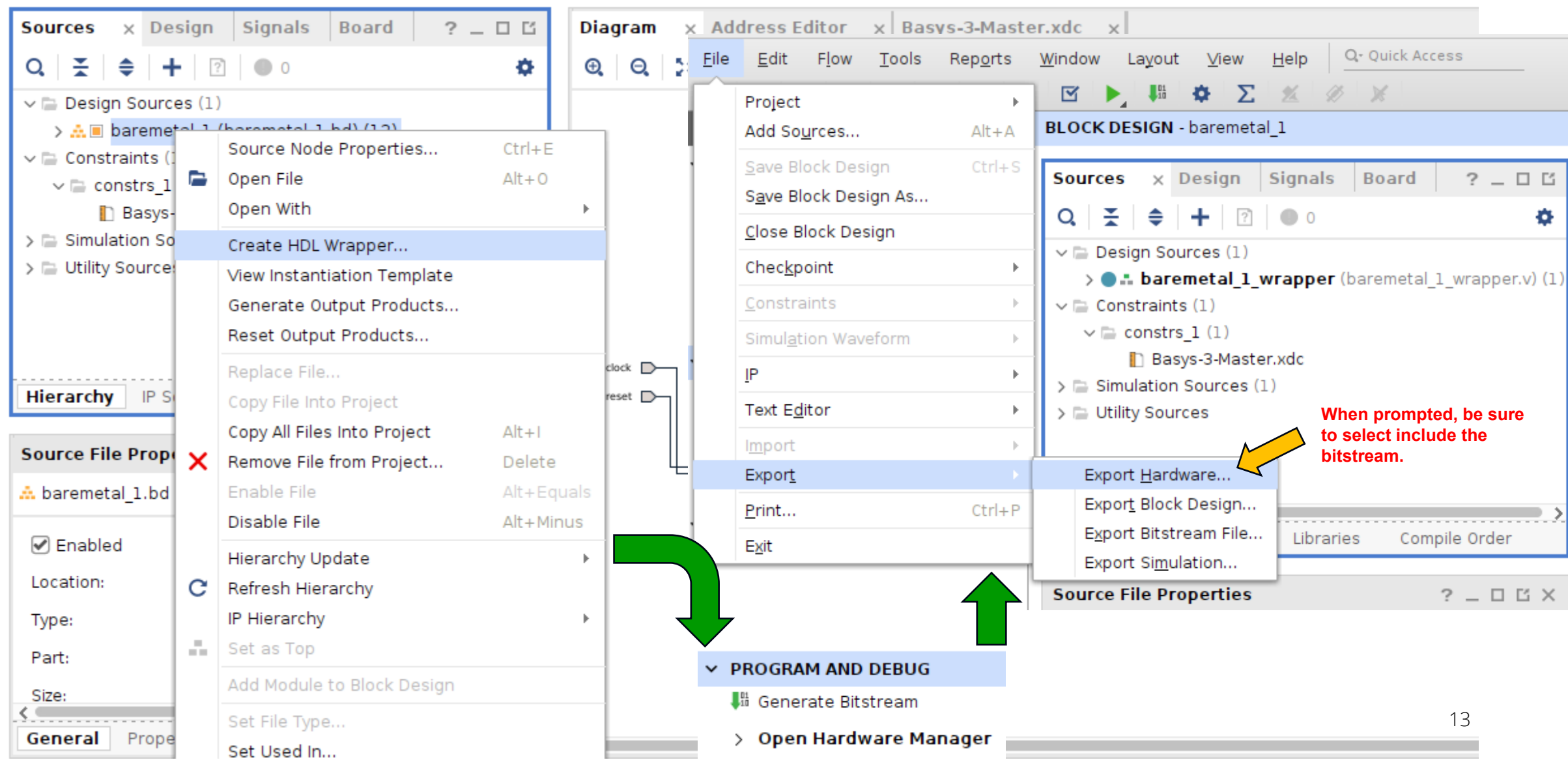
Validate Design

Validation successful. There are no errors or critical warnings in this design.

OK

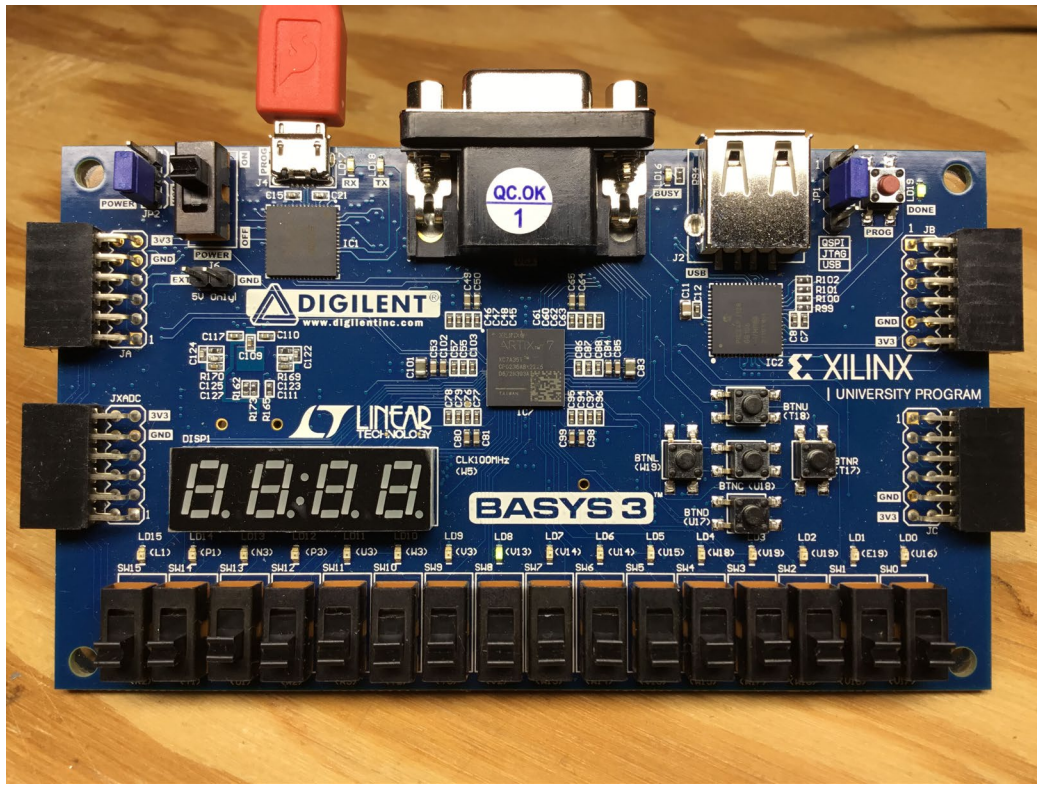


Create the HDL Wrapper – Generate Bitstream - Export



The screenshot shows the Vivado IDE interface with several windows and menus open. The 'Sources' window on the left shows a project named 'baremetal_1' with a source file 'baremetal_1_wrapper.v'. The 'Diagram' window in the center shows a block design with 'clock' and 'reset' inputs. The 'File' menu is open, and the 'Export' option is selected, leading to a sub-menu where 'Export Hardware...' is highlighted. A yellow arrow points to this option with the text: "When prompted, be sure to select include the bitstream." Below the 'Export Hardware...' option, the 'PROGRAM AND DEBUG' section is expanded, showing 'Generate Bitstream' and 'Open Hardware Manager'. Green arrows indicate the flow from the 'Export Hardware...' option to the 'PROGRAM AND DEBUG' section. The 'Source File Properties' window is also visible at the bottom right.

Activate Hardware Manager



File Edit Flow Tools Reports Window Layout View Help | Q Quick Access

Flow Navigator

HARDWARE MANAGER - localhost/xilinx_tcf/Digilent/210183B7AFD3A

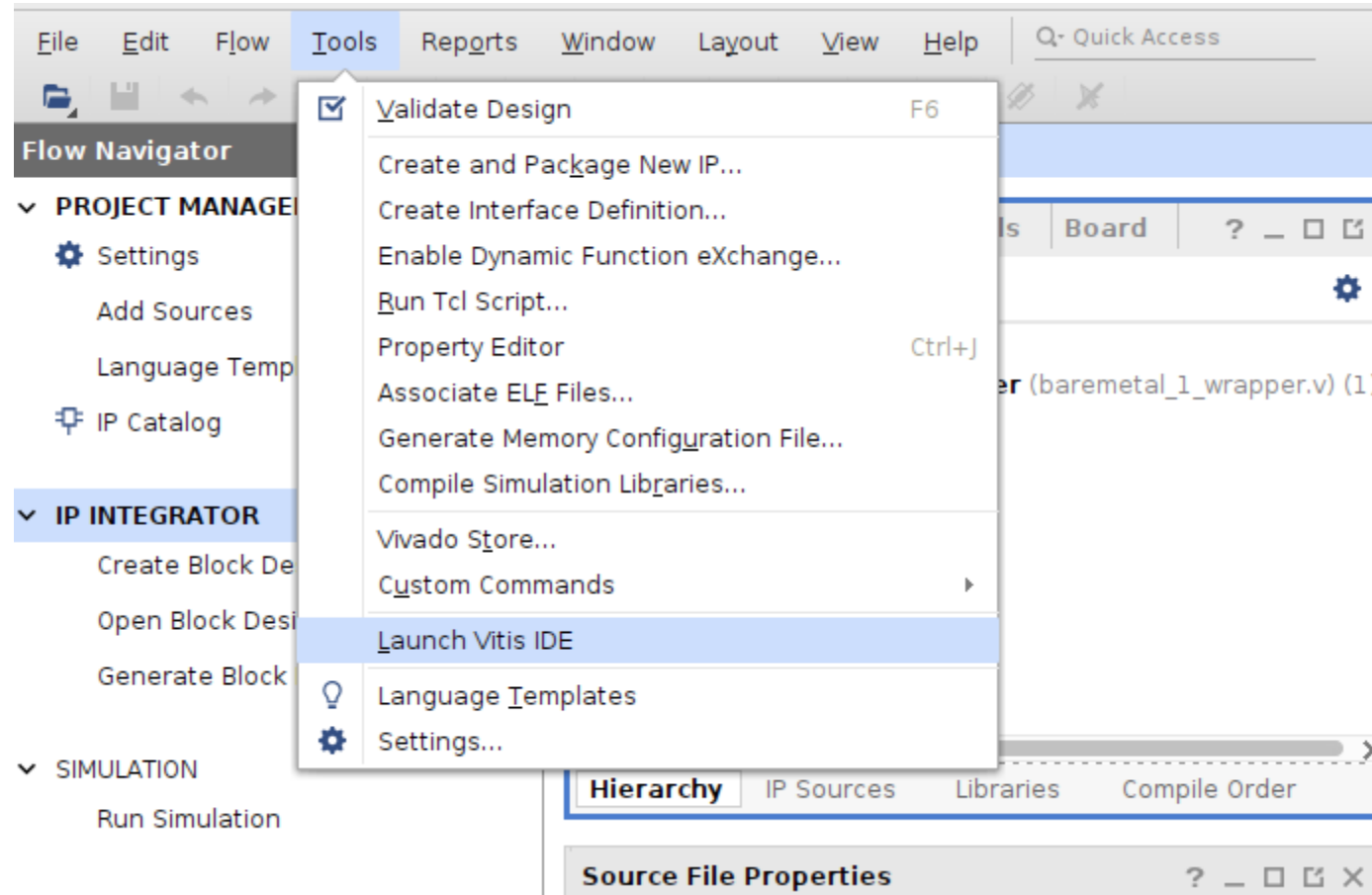
There are no debug cores. [Program device](#) [Refresh device](#)

Hardware	
Name	Status
localhost (1)	Connected
xilinx_tcf/Digilent/210183B7AFD3A (1)	Open
xc7a35t_0 (1)	Programmed
XADC (System Monitor)	

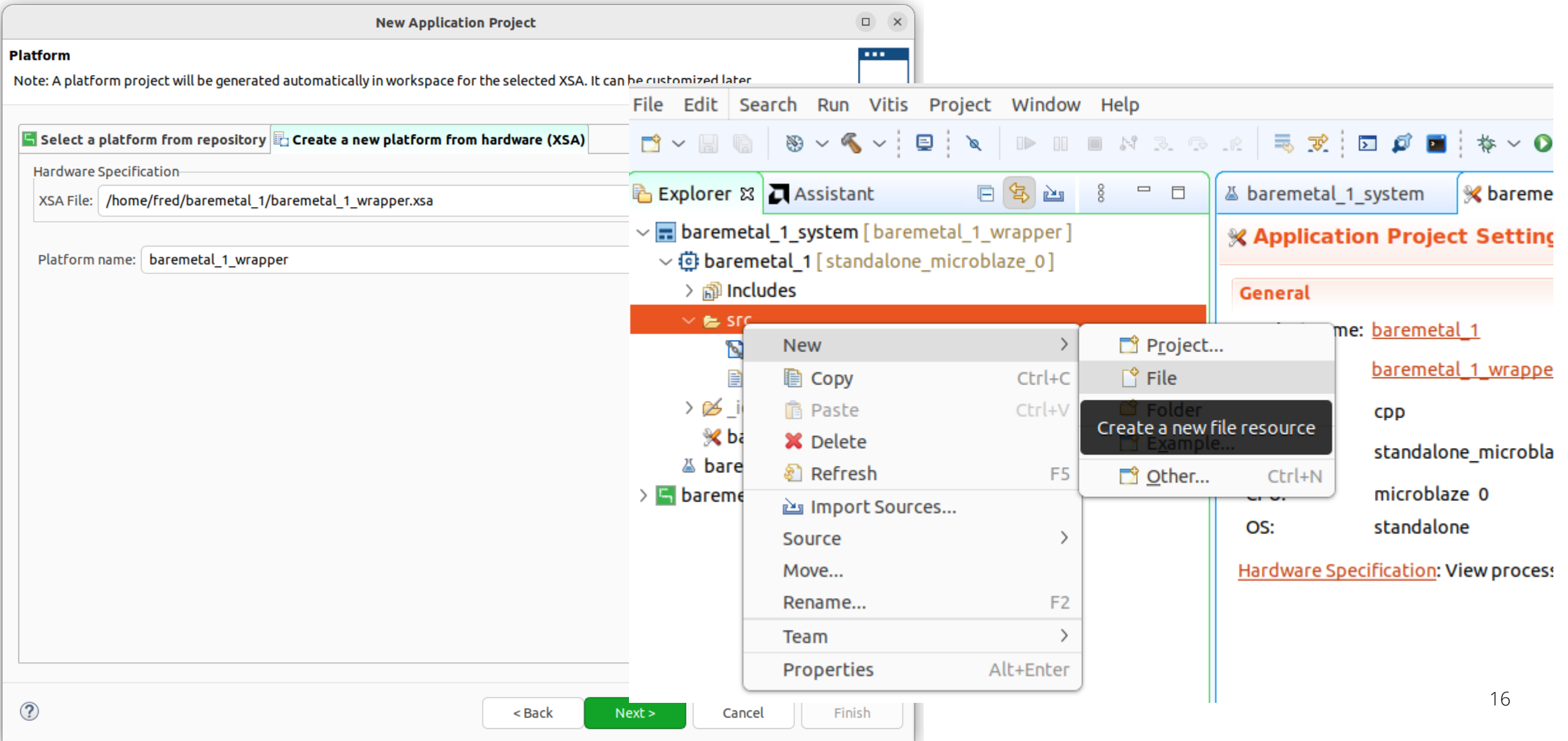
```

6  ## Clock
7  set_pro
8  create_
9
10
11 ## Switch
12 #set_pi
13 #set_pi
14 #set_pi
15 #set_pi
16 #set_pi
17 #set_pi
18 #set_pi
19 #set_pi
20 #set_pi
  
```


Enter... Vitis



Create main.c



New Application Project

Platform
Note: A platform project will be generated automatically in workspace for the selected XSA. It can be customized later.

Select a platform from repository | Create a new platform from hardware (XSA)

Hardware Specification
XSA File: /home/fred/baremetal_1/baremetal_1_wrapper.xsa
Platform name: baremetal_1_wrapper

File Edit Search Run Vitis Project Window Help

Explorer Assistant

- baremetal_1_system [baremetal_1_wrapper]
 - baremetal_1 [standalone_microblaze_0]
 - Includes
 - src

Project...
File
Folder
Create a new file resource
Example...
Other... Ctrl+N

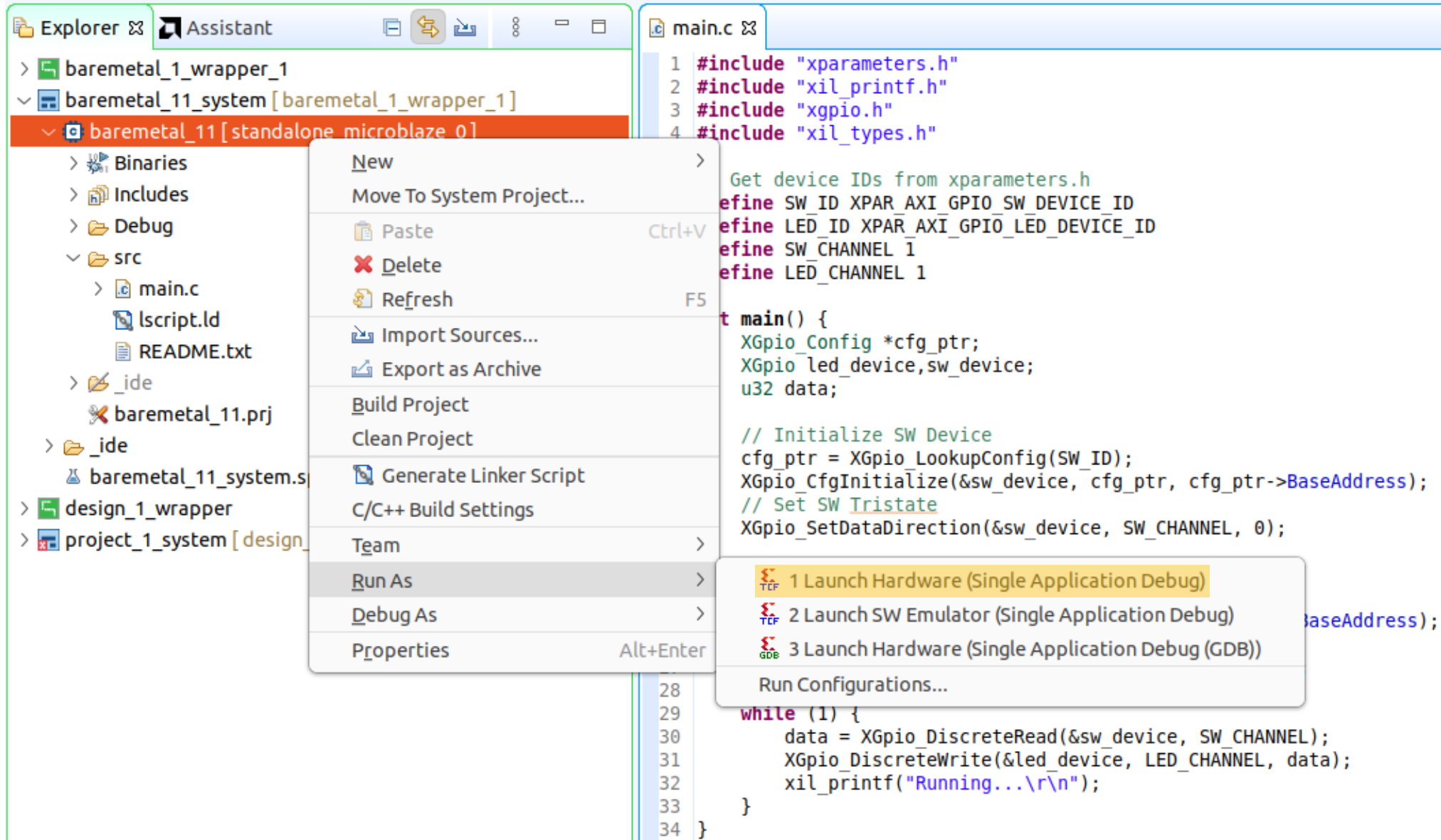
Application Project Settings
General
Name: baremetal_1
baremetal_1_wrapper
cpp
standalone_microblaze_0
microblaze_0
OS: standalone
Hardware Specification: View process...

< Back Next > Cancel Finish

Vitis Application Code

```
main.c ✖
1 #include "xparameters.h"
2 #include "xil_printf.h"
3 #include "xgpio.h"
4 #include "xil_types.h"
5
6 // Get device IDs from xparameters.h
7 #define SW_ID XPAR_AXI_GPIO_SW_DEVICE_ID
8 #define LED_ID XPAR_AXI_GPIO_LED_DEVICE_ID
9 #define SW_CHANNEL 1
10 #define LED_CHANNEL 1
11
12 int main() {
13     XGpio_Config *cfg_ptr;
14     XGpio led_device, sw_device;
15     u32 data;
16
17     // Initialize SW Device
18     cfg_ptr = XGpio_LookupConfig(SW_ID);
19     XGpio_CfgInitialize(&sw_device, cfg_ptr, cfg_ptr->BaseAddress);
20     // Set SW Tristate
21     XGpio_SetDataDirection(&sw_device, SW_CHANNEL, 0);
22
23     // Initialize LED Device
24     cfg_ptr = XGpio_LookupConfig(LED_ID);
25     XGpio_CfgInitialize(&led_device, cfg_ptr, cfg_ptr->BaseAddress);
26     // Set Led Tristate
27     XGpio_SetDataDirection(&led_device, LED_CHANNEL, 0);
28
29     while (1) {
30         data = XGpio_DiscreteRead(&sw_device, SW_CHANNEL);
31         XGpio_DiscreteWrite(&led_device, LED_CHANNEL, data);
32         xil_printf("Running...\r\n");
33     }
34 }
```


Run it



The screenshot shows the Vitis IDE interface. The Explorer pane on the left displays a project structure with the following folders and files:

- baremetal_1_wrapper_1
 - baremetal_11_system [baremetal_1_wrapper_1]
 - baremetal_11 [standalone_microblaze_0]
 - Binaries
 - Includes
 - Debug
 - src
 - main.c
 - lscript.ld
 - README.txt
 - _ide
 - baremetal_11.prj
 - _ide
 - baremetal_11_system.s
 - design_1_wrapper
 - project_1_system [design_1_wrapper]

The main editor window shows the code in `main.c`:

```

1 #include "xparameters.h"
2 #include "xil_printf.h"
3 #include "xgpio.h"
4 #include "xil_types.h"

Get device IDs from xparameters.h
#define SW_ID XPAR_AXI_GPIO_SW_DEVICE_ID
#define LED_ID XPAR_AXI_GPIO_LED_DEVICE_ID
#define SW_CHANNEL 1
#define LED_CHANNEL 1

int main() {
    XGpio_Config *cfg_ptr;
    XGpio_led_device,sw_device;
    u32 data;

    // Initialize SW Device
    cfg_ptr = XGpio_LookupConfig(SW_ID);
    XGpio_CfgInitialize(&sw_device, cfg_ptr, cfg_ptr->BaseAddress);
    // Set SW Tristate
    XGpio_SetDataDirection(&sw_device, SW_CHANNEL, 0);

    while (1) {
        data = XGpio_DiscreteRead(&sw_device, SW_CHANNEL);
        XGpio_DiscreteWrite(&led_device, LED_CHANNEL, data);
        xil_printf("Running...\r\n");
    }
}

```

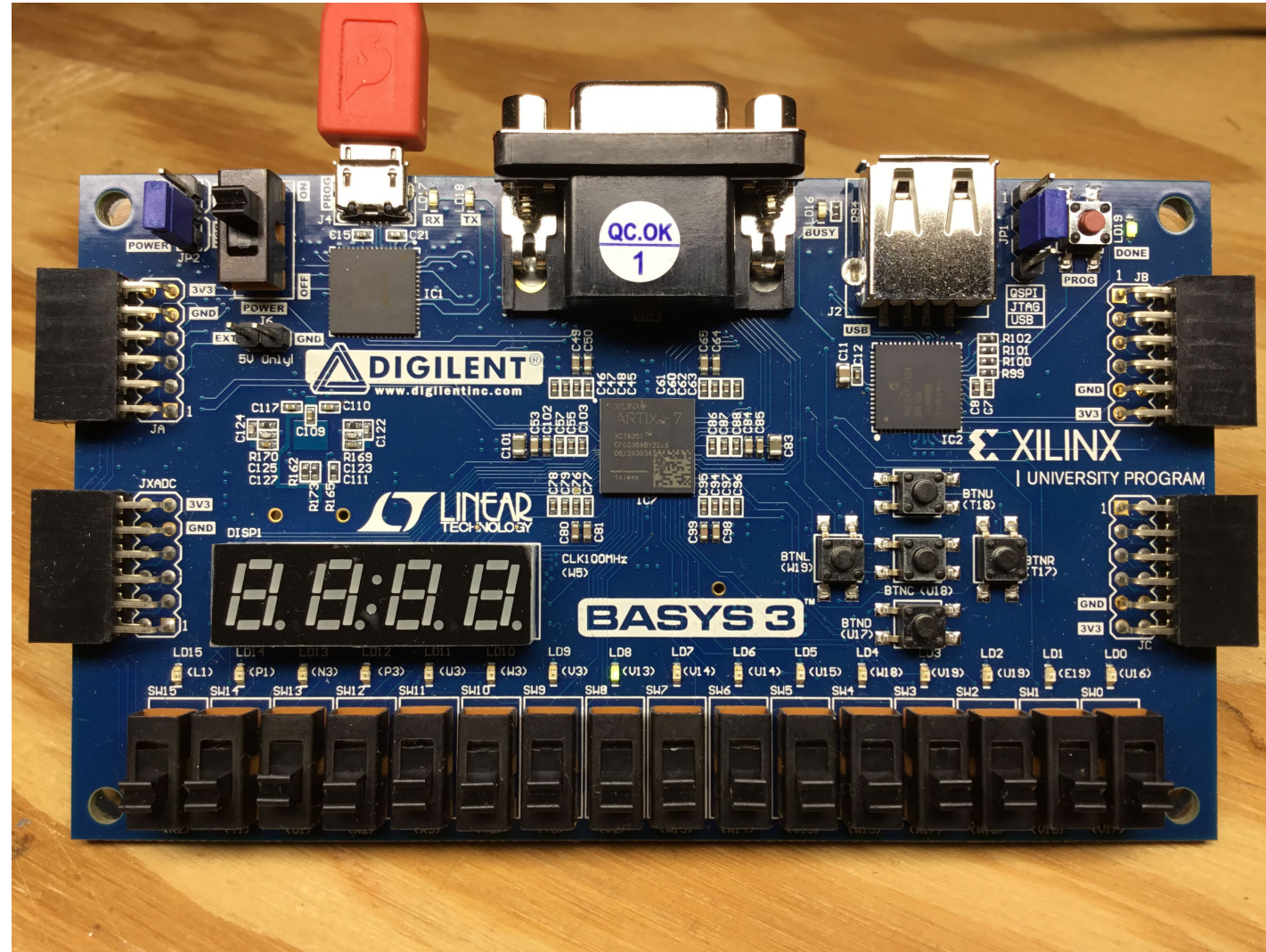
The 'Run As' menu is open, showing the following options:

- 1 Launch Hardware (Single Application Debug)
- 2 Launch SW Emulator (Single Application Debug)
- 3 Launch Hardware (Single Application Debug (GDB))
- Run Configurations...

Thank you for attending!!!

Please consider the resources below:

- xilinx.com
- digilent.com
- [Basys 3 Reference Manual](#)





DesignNews

Thank You

Sponsored by

