

# NIOS I/O

Last updated 10/12/20

# NIOS I/O

These slides describe the development of a simple NIOS Processor with basic I/O

Upon completion: You should be able implement your own NIOS processor with basic I/O and write code for it

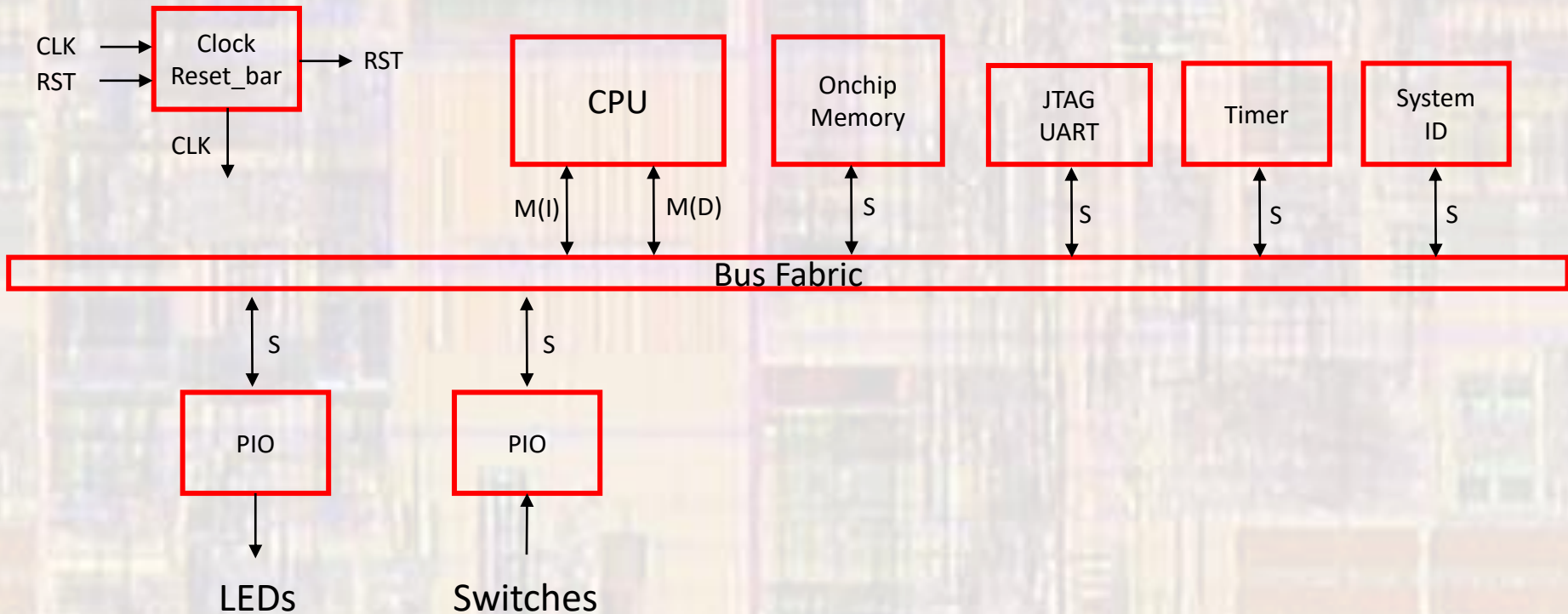
# NIOS I/O

- NIOS II Embedded Design Suite
  - Configurable Processor
  - Selection of Peripherals
  - Eclipse based Board Support Package (BSP) for SW development

# NIOS I/O

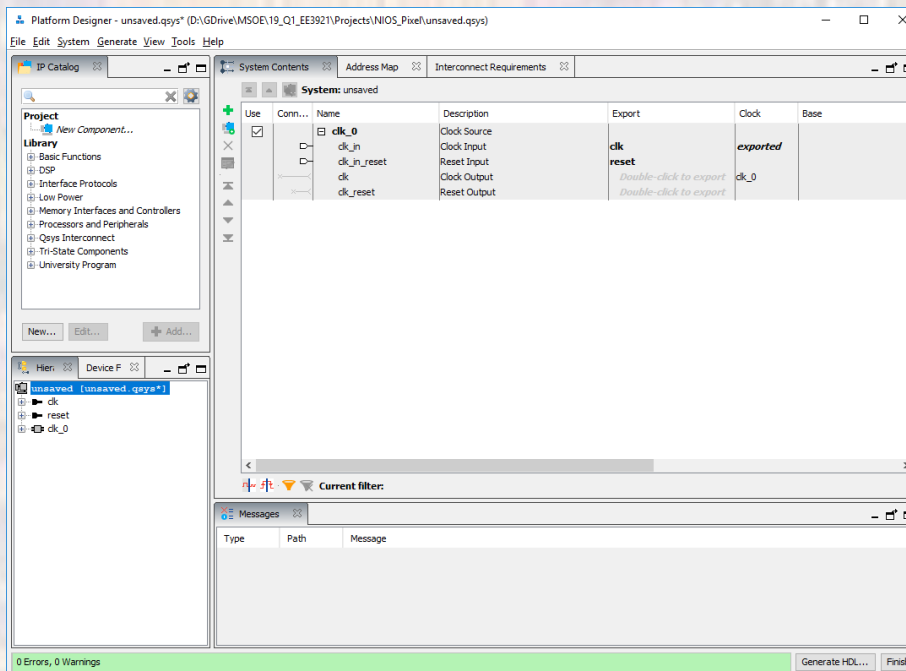
- NIOS I/O System

- Create a processor system to display a count on LEDs with switch control for up/dn, pause



# NIOS I/O

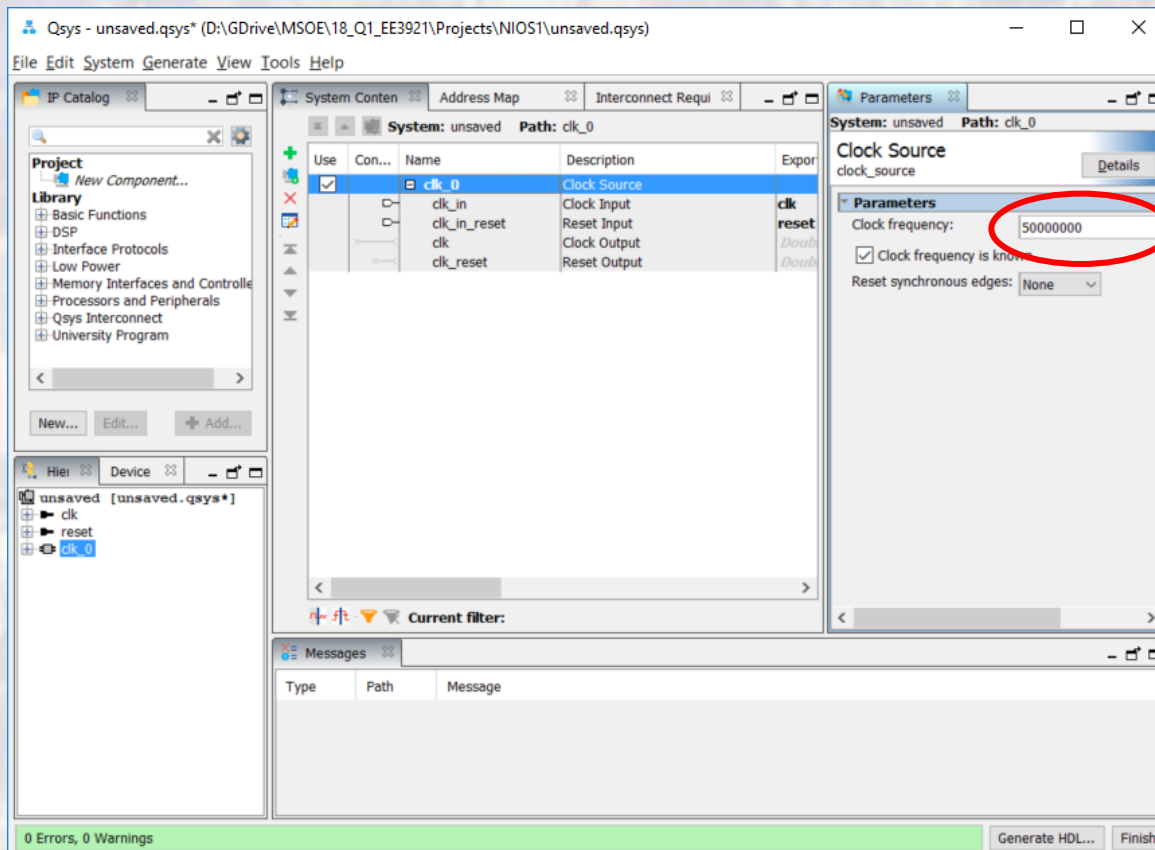
- Create a new Quartus project
  - **Do not** select a Simulation Tool in EDA Tool Settings
- Open **Tools** → **Platform Designer**





# NIOS I/O

- Create NIOS System
  - Double Click on clk\_0 - verify clk frequency = 50MHz



# NIOS I/O

- Add NIOS
  - Processors and Peripherals → Embedded Processors → NIOS II Processor
  - NIOS II/f

<input type="checkbox"/>	clk_reset	Reset Output	Double-click to export		
<input checked="" type="checkbox"/>	<b>nios2_gen2_0</b>	Nios II Processor			
<input type="checkbox"/>	clk	Clock Input	Double-click to export	unconnected	
<input type="checkbox"/>	reset	Reset Input	Double-click to export	[clk]	
<input type="checkbox"/>	data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
<input type="checkbox"/>	instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
<input type="checkbox"/>	irq	Interrupt Receiver	Double-click to export	[clk]	
<input type="checkbox"/>	debug_reset_request	Reset Output	Double-click to export	[clk]	
<input type="checkbox"/>	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x08
<input type="checkbox"/>	custom_instruction_m...	Custom Instruction Master	Double-click to export		

- Add On-chip Memory
  - Basic Functions → On Chip Memory → On Chip Memory (RAM or ROM)...

RAM

Size = 20,000 bytes

<input type="checkbox"/>	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x08
<input type="checkbox"/>	custom_instruction_m...	Custom Instruction Master	Double-click to export		
<input checked="" type="checkbox"/>	<b>onchip_memory2_0</b>	On-Chip Memory (RAM or ROM) Intel ...			
<input type="checkbox"/>	clk1	Clock Input	Double-click to export	unconnected	
<input type="checkbox"/>	s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	
<input type="checkbox"/>	reset1	Reset Input	Double-click to export	[clk1]	

# NIOS I/O

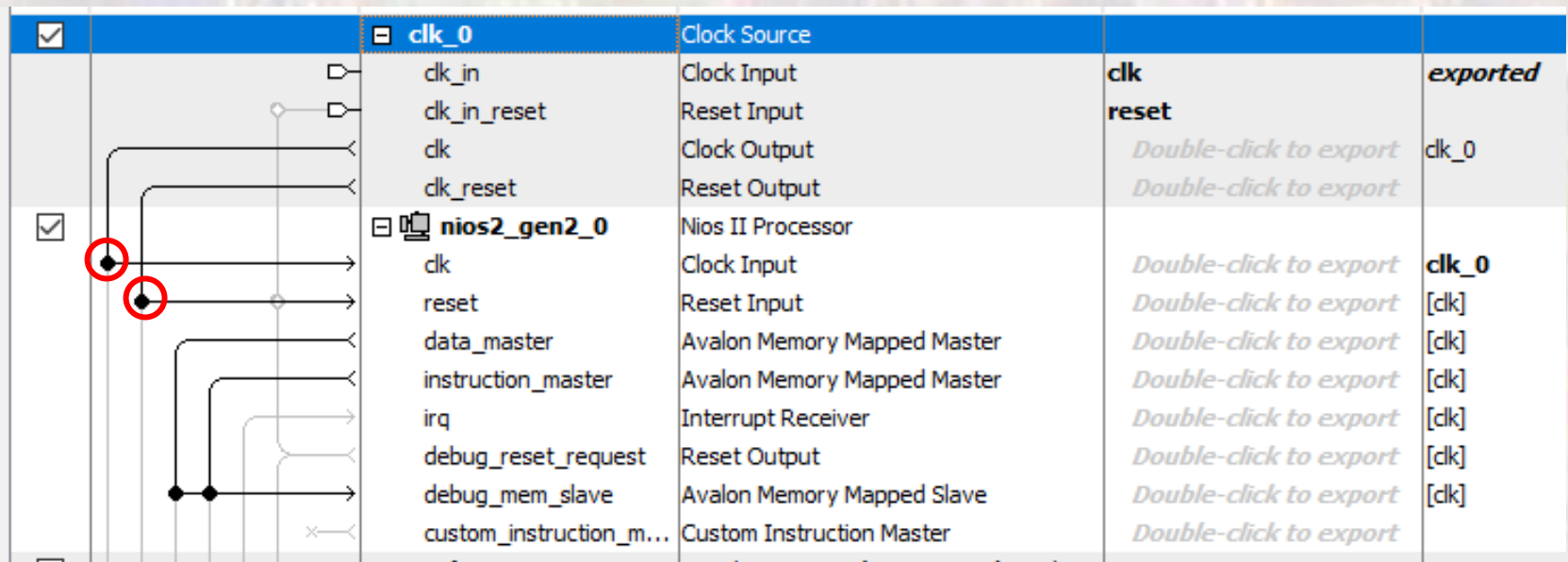
- Add JTAG
  - Interface Protocols → Serial → JTAG Uart Intel FPGA IP
- Add Timer
  - Processors and Peripherals → Peripherals → Interval Timer Intel FPGA IP
- Add System ID
  - Basic Functions → Simulation; Debug and Verification → Debug and Performance → System ID Peripheral Intel FPGA IP

<input checked="" type="checkbox"/>	reset1	Reset Input	<i>Double-click to export</i>	[clk1]
<input checked="" type="checkbox"/>	<ul style="list-style-type: none"> <li>[-] <b>jtag_uart_0</b></li> <li>clk</li> <li>reset</li> <li>avalon_jtag_slave</li> <li>irq</li> </ul>	<ul style="list-style-type: none"> <li>JTAG UART Intel FPGA IP</li> <li>Clock Input</li> <li>Reset Input</li> <li>Avalon Memory Mapped Slave</li> <li>Interrupt Sender</li> </ul>	<ul style="list-style-type: none"> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> </ul>	<ul style="list-style-type: none"> <li><b>unconnected</b></li> <li>[clk]</li> <li>[clk]</li> <li>[clk]</li> </ul>
<input checked="" type="checkbox"/>	<ul style="list-style-type: none"> <li>[-] <b>timer_0</b></li> <li>clk</li> <li>reset</li> <li>s1</li> <li>irq</li> </ul>	<ul style="list-style-type: none"> <li>Interval Timer Intel FPGA IP</li> <li>Clock Input</li> <li>Reset Input</li> <li>Avalon Memory Mapped Slave</li> <li>Interrupt Sender</li> </ul>	<ul style="list-style-type: none"> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> </ul>	<ul style="list-style-type: none"> <li><b>unconnected</b></li> <li>[clk]</li> <li>[clk]</li> <li>[clk]</li> </ul>
<input checked="" type="checkbox"/>	<ul style="list-style-type: none"> <li>[-] <b>sysid_qsyz_0</b></li> <li>clk</li> <li>reset</li> <li>control_slave</li> </ul>	<ul style="list-style-type: none"> <li>System ID Peripheral Intel FPGA IP</li> <li>Clock Input</li> <li>Reset Input</li> <li>Avalon Memory Mapped Slave</li> </ul>	<ul style="list-style-type: none"> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> <li><i>Double-click to export</i></li> </ul>	<ul style="list-style-type: none"> <li><b>unconnected</b></li> <li>[clk]</li> <li>[clk]</li> </ul>



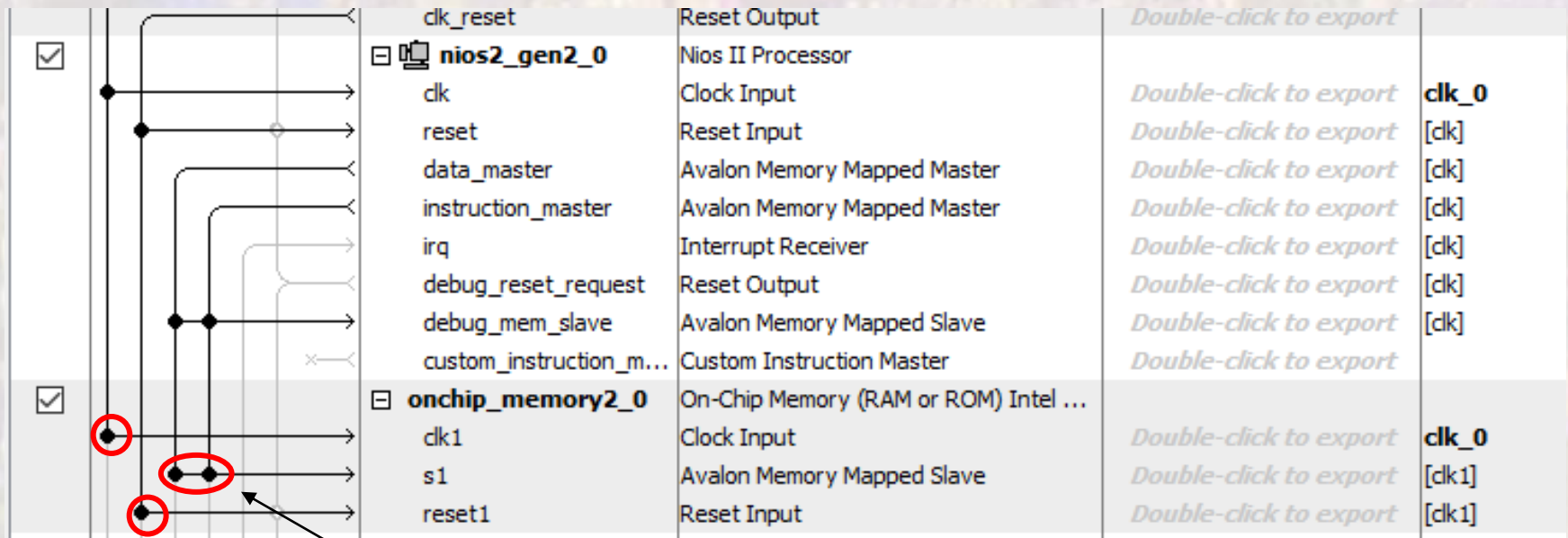
# NIOS I/O

- Connect up NIOS I/O
  - NIOS Inputs



# NIOS I/O

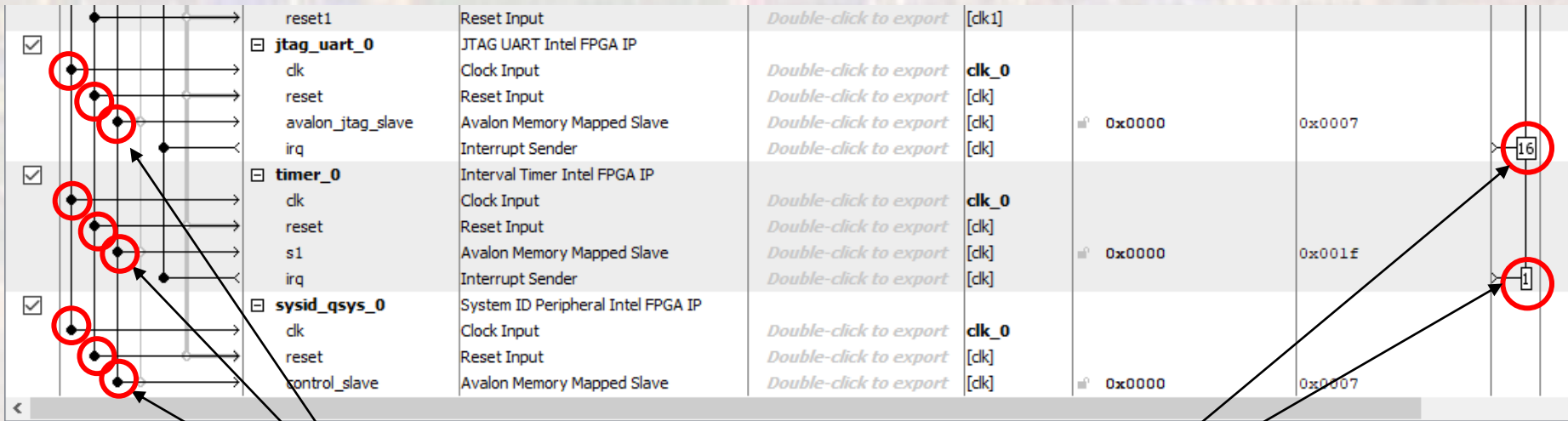
- Connect up NIOS I/O
  - On-chip Memory



Connect to data and instruction masters

# NIOS I/O

- Connect up NIOS I/O
  - JTAG, Timer, SysID

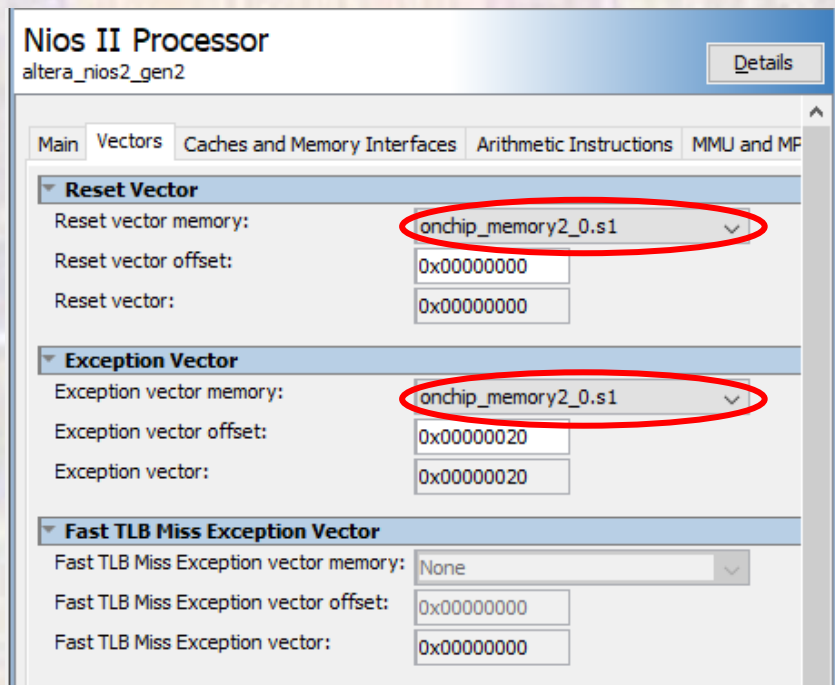


Connect to data master

Assign Priorities

# NIOS I/O

- Connect up NIOS I/O
  - Assign the NIOS II Reset and Exception vectors
    - Open the NIOS Processor
    - Select **Vectors**
    - Select on-chip memory for Reset and Exception



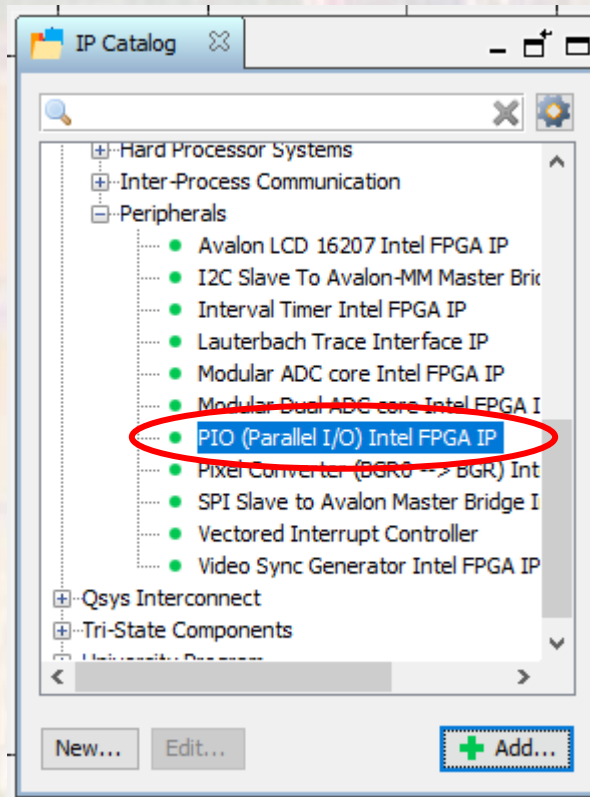


# NIOS I/O

- Customize System

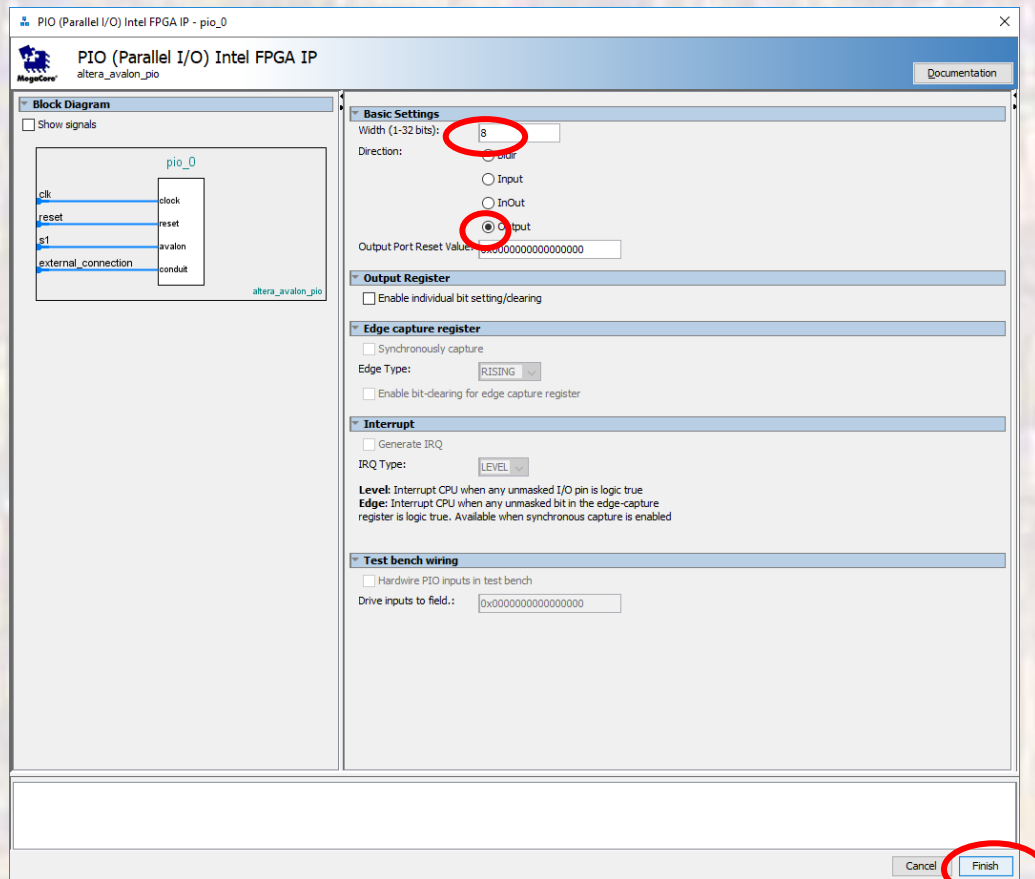
- Create LED output

- IP Catalog → Library → Processors and Peripherals → Peripherals → PIO (Parallel I/O)



# NIOS I/O

- Customize System
  - Create LED output
    - Set 8 bit, select output



# NIOS I/O

- Customize System
  - Hookup LED output

The screenshot shows the 'Connections' window in the NIOS IDE. It displays a list of components and their connections. The 'pio\_0' component is highlighted in blue. A red box highlights the 'led\_pio' component in the list, with a red circle around its name. A text box with red text says 'right click the PIO name and change it to led\_pio'.

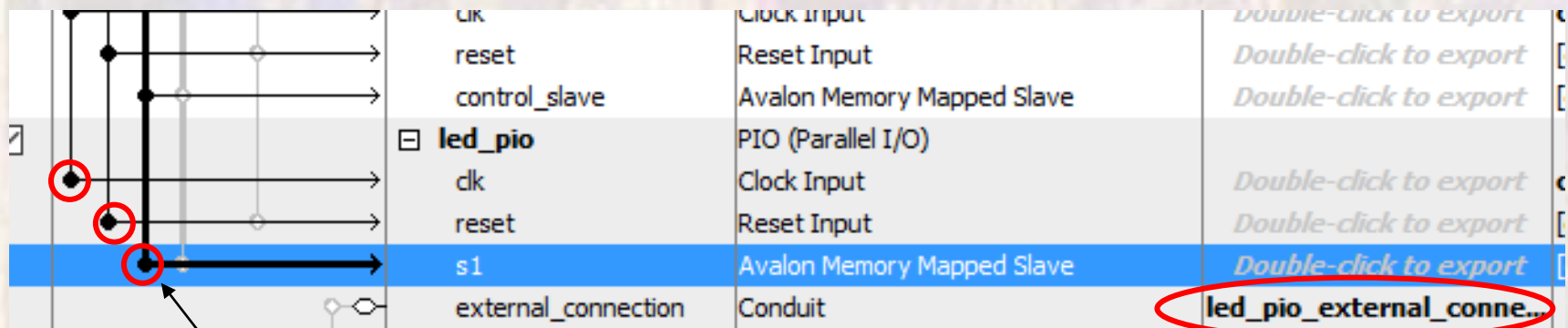
se	Connections	Name	Description
		debug_reset_request	Reset Output
		debug_mem_slave	Avalon Memory Mapped Slave
		custom_instruction_m...	Custom Instruction Master
<input checked="" type="checkbox"/>		<b>jtag_uart</b>	JTAG UART
		clk	Clock Input
		reset	Reset Input
		avalon_jtag_slave	Avalon Memory Mapped Slave
		irq	Interrupt Sender
<input checked="" type="checkbox"/>		<b>sys_clk_timer</b>	Interval Timer
		clk	Clock Input
		reset	Reset Input
		s1	Avalon Memory Mapped Slave
		irq	Interrupt Sender
<input checked="" type="checkbox"/>		<b>sysid</b>	System ID Peripheral
		clk	Clock Input
		reset	Reset Input
		control_slave	Avalon Memory Mapped Slave
<input checked="" type="checkbox"/>		<b>pio_0</b>	PIO (Parallel I/O)
		clk	Clock Input
		reset	Reset Input
		s1	Avalon Memory Mapped Slave
		external_connection	Conduit

right click the PIO name and change it to led\_pio

		control_slave	Avalon Memory Mapped Slave
		<b>led_pio</b>	PIO (Parallel I/O)
		clk	Clock Input
		reset	Reset Input
		s1	Avalon Memory Mapped Slave
		external_connection	Conduit

# NIOS I/O

- Customize System
  - Hookup LED output



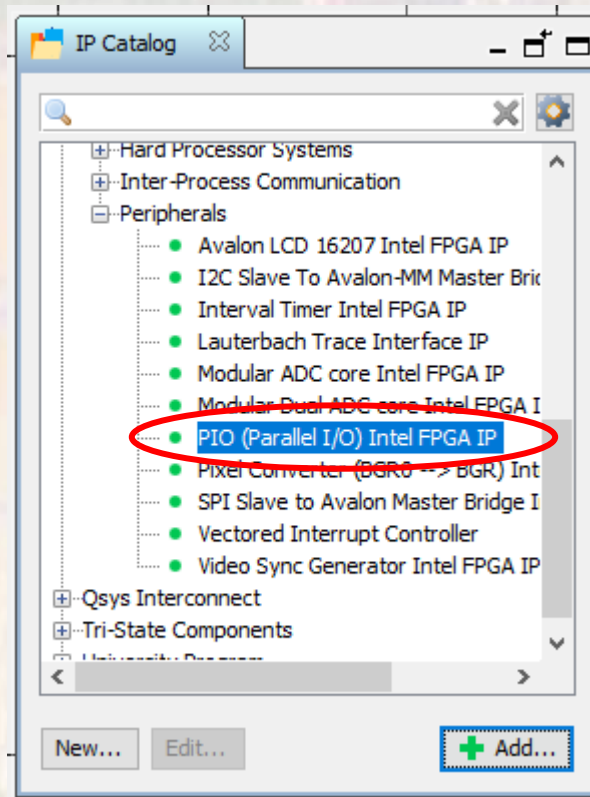
Connect to data master

Double click to export  
Makes the connection visible  
outside the system



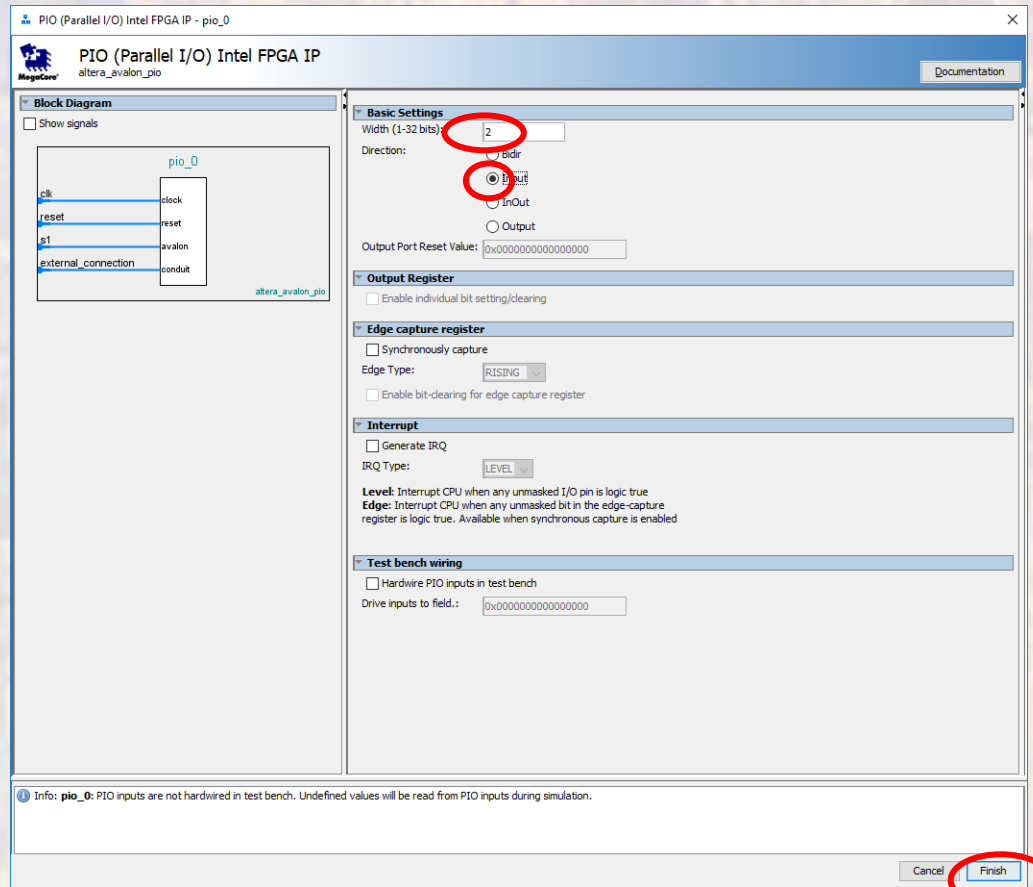
# NIOS I/O

- Customize System
  - Create switch inputs
    - IP Catalog → Library → Processors and Peripherals → Peripherals → PIO (Parallel I/O)



# NIOS I/O

- Customize System
  - Create switch inputs
    - Set 2 bit, select input



# NIOS I/O

- Customize System
  - Hookup switch input

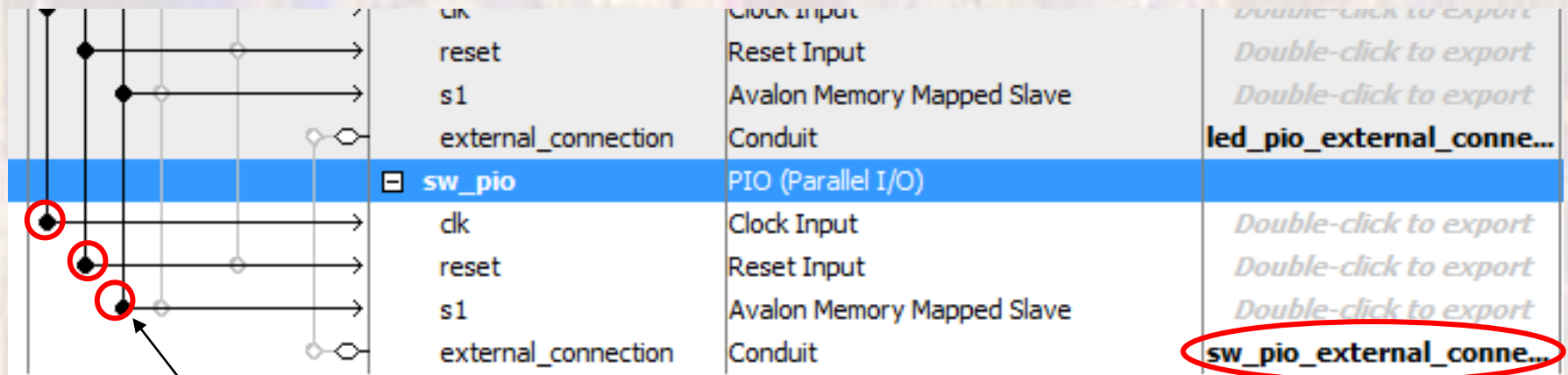
The screenshot shows the 'Connections' window in a NIOS IDE. It displays a list of components and their connections. The 'pio\_0' component is highlighted in blue. A red box highlights a sub-section of the list where 'sw\_pio' is being added to the 'pio\_0' component's connections.

Component	Connection	Description
reset	reset	Reset Input
	avalon_jtag_slave	Avalon Memory Mapped Slave
	irq	Interrupt Sender
sys_clk_timer	clk	Clock Input
	reset	Reset Input
	irq	Interrupt Sender
sysid	clk	Clock Input
	reset	Reset Input
	control_slave	Avalon Memory Mapped Slave
led_pio	clk	Clock Input
	reset	Reset Input
	s1	Avalon Memory Mapped Slave
pio_0	clk	Clock Input
	reset	Reset Input
	s1	Avalon Memory Mapped Slave
external_connection	external_connection	Conduit
	s1	Avalon Memory Mapped Slave
	external_connection	Conduit
sw_pio	clk	Clock Input
	reset	Reset Input
	s1	Avalon Memory Mapped Slave
external_connection	external_connection	Conduit

right click the PIO name and change it to sw\_pio

# NIOS I/O

- Customize System
  - Hookup switch input



Connect to data master

Double click to export

Makes the connection visible  
outside the system



# NIOS I/O

- Complete Custom System
  - Assign base addresses
- System → Assign Base Addresses

The image shows two windows from the Quartus II software. The left window displays the 'System Contents' table, and the right window displays the 'Address Map' table.

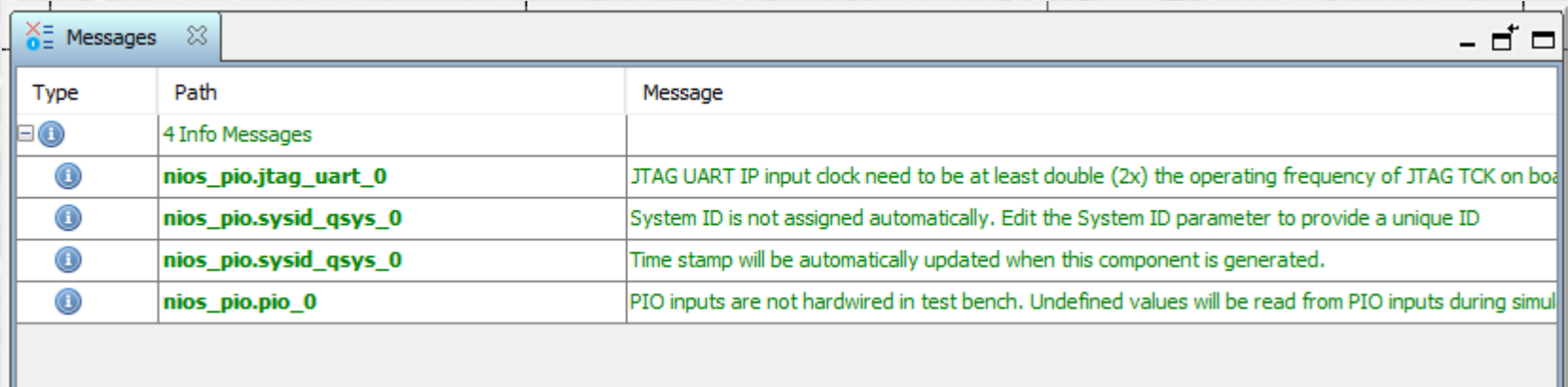
Base	End	IRQ	Tags
# 0x0800	0x0fff	IRQ 0	IRQ 31
# 0x0000	0x2edf		
# 0x0000	0x0007		
# 0x0000	0x001f		
# 0x0000	0x0007		
# 0x0000	0x000e		
# 0x0000	0x000e		

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
<input checked="" type="checkbox"/>		clk_0	Clock Source		exported				
<input checked="" type="checkbox"/>		clk_in	Clock Input	clk					
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input	reset					
<input checked="" type="checkbox"/>		clk	Clock Output	clk_0	clk_0				
<input checked="" type="checkbox"/>		clk_reset	Reset Output	reset					
<input checked="" type="checkbox"/>		nios2_gen2_0	Nios II Processor						
<input checked="" type="checkbox"/>		clk	Clock Input	clk_0	clk_0				
<input checked="" type="checkbox"/>		reset	Reset Input	reset					
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped Master	[clk]					
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master	[clk]					
<input checked="" type="checkbox"/>		irq	Interrupt Receiver	irq				IRQ 0	IRQ 31
<input checked="" type="checkbox"/>		debug_reset_request	Reset Output	reset					
<input checked="" type="checkbox"/>		debug_mem_slave	Avalon Memory Mapped Slave	[clk]					
<input checked="" type="checkbox"/>		custom_instruction_m...	Custom Instruction Master	[clk]					
<input checked="" type="checkbox"/>		onchip_memory2_0	On-Chip Memory (RAM or ROM) Intel ...						
<input checked="" type="checkbox"/>		clk1	Clock Input	clk_0	clk_0				
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	[clk-1]					
<input checked="" type="checkbox"/>		reset1	Reset Input	reset					
<input checked="" type="checkbox"/>		jtag_uart_0	JTAG UART Intel FPGA IP						
<input checked="" type="checkbox"/>		clk	Clock Input	clk_0	clk_0				
<input checked="" type="checkbox"/>		reset	Reset Input	reset					
<input checked="" type="checkbox"/>		avalon_jtag_slave	Avalon Memory Mapped Slave	[clk]					
<input checked="" type="checkbox"/>		irq	Interrupt Sender	irq					
<input checked="" type="checkbox"/>		timer_0	Interval Timer Intel FPGA IP						
<input checked="" type="checkbox"/>		clk	Clock Input	clk_0	clk_0				
<input checked="" type="checkbox"/>		reset	Reset Input	reset					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	[clk]					
<input checked="" type="checkbox"/>		irq	Interrupt Sender	irq					
<input checked="" type="checkbox"/>		sysid_qsys_0	System ID Peripheral Intel FPGA IP						
<input checked="" type="checkbox"/>		clk	Clock Input	clk_0	clk_0				
<input checked="" type="checkbox"/>		reset	Reset Input	reset					
<input checked="" type="checkbox"/>		control_slave	Avalon Memory Mapped Slave	[clk]					
<input checked="" type="checkbox"/>		led_pio	PIO (Parallel I/O) Intel FPGA IP						
<input checked="" type="checkbox"/>		clk	Clock Input	clk_0	clk_0				
<input checked="" type="checkbox"/>		reset	Reset Input	reset					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	[clk]					
<input checked="" type="checkbox"/>		external_connection	Conduit	led_pio_external_conne...					
<input checked="" type="checkbox"/>		sw_pio	PIO (Parallel I/O) Intel FPGA IP						
<input checked="" type="checkbox"/>		clk	Clock Input	clk_0	clk_0				
<input checked="" type="checkbox"/>		reset	Reset Input	reset					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave	[clk]					
<input checked="" type="checkbox"/>		external_connection	Conduit	sw_pio_external_conne...					

# NIOS I/O

- Create Custom System
  - Check for errors

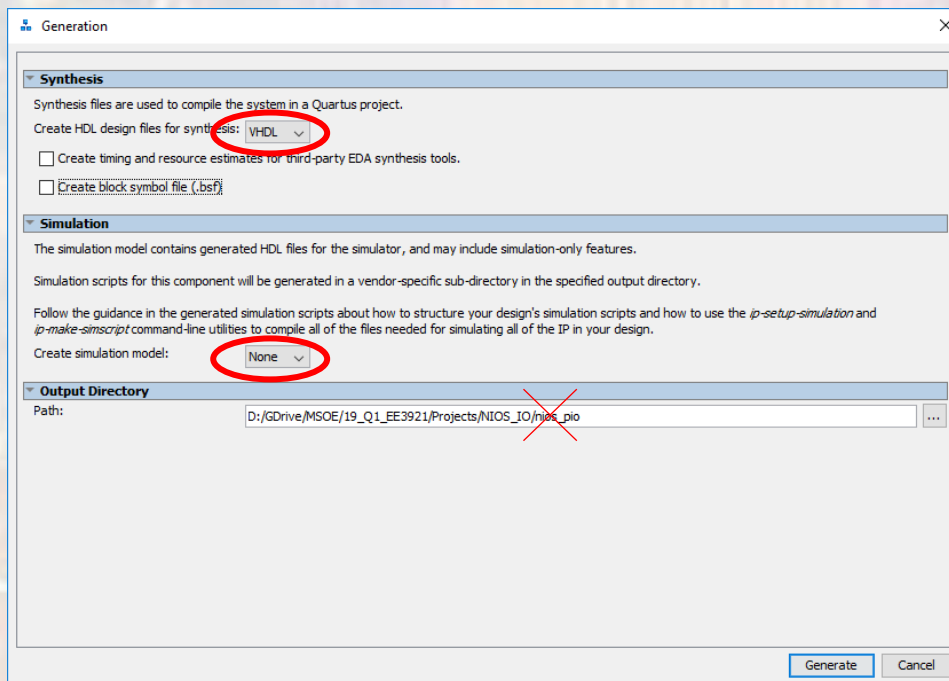


The screenshot shows a window titled "Messages" with a table of information messages. The table has three columns: "Type", "Path", and "Message". There are four rows of messages, all of which are information messages (indicated by an 'i' icon in the Type column).

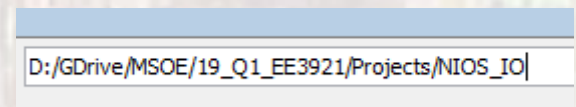
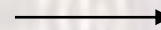
Type	Path	Message
4 Info Messages		
i	<b>nios_pio.jtag_uart_0</b>	JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board
i	<b>nios_pio.sysid_qsys_0</b>	System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID
i	<b>nios_pio.sysid_qsys_0</b>	Time stamp will be automatically updated when this component is generated.
i	<b>nios_pio.pio_0</b>	PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simul

# NIOS I/O

- Create Basic System
  - Save the Platform Designer system
  - Generate the Platform Designer system
    - **Generate** → **Generate HDL**
    - The first time you generate you must delete the last directory in the path – **don't use the '...'**

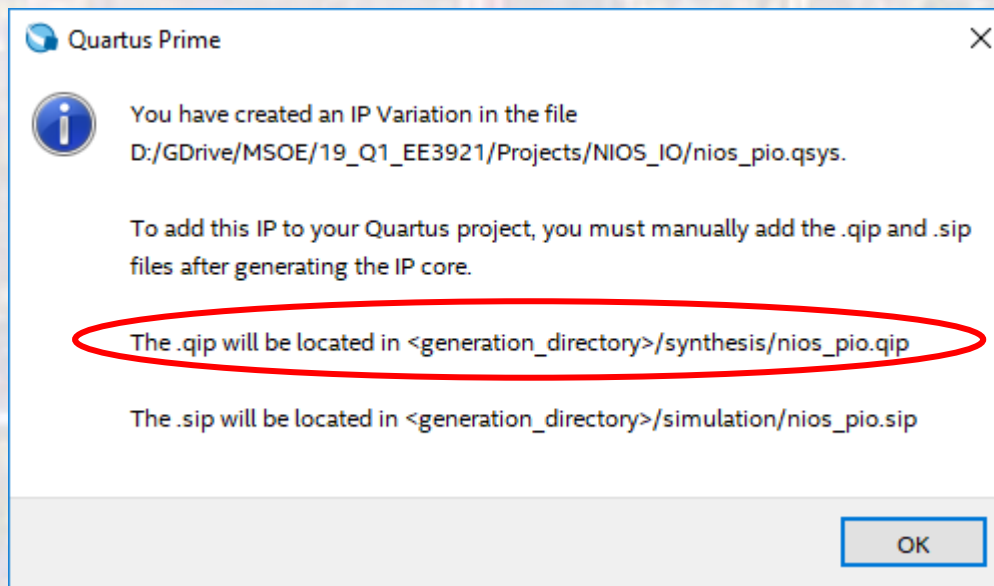


Should point to your project directory



# NIOS I/O

- Create Custom System
  - Add the .qip file to the project





# NIOS I/O

- Create DE10 Design
  - Instantiate into a VHDL file
    - Open a new VHDL design
    - In Platform Designer: **Generate** → **Show Instantiation Template**
    - Copy and Paste into the new design where appropriate

```
component nios_pio is
  port (
    clk_clk          : in std_logic          := 'X';          -- clk
    led_pio_external_connection_export : out std_logic_vector(7 downto 0);          -- export
    reset_reset_n    : in std_logic          := 'X';          -- reset_n
    sw_pio_external_connection_export : in std_logic_vector(1 downto 0) := (others => 'X') -- export
  );
end component nios_pio;

u0 : component nios_pio
  port map (
    clk_clk          => CONNECTED_TO_clk_clk,          --          clk.clk
    led_pio_external_connection_export => CONNECTED_TO_led_pio_external_connection_export, -- led_pio_external_connection.export
    reset_reset_n    => CONNECTED_TO_reset_reset_n,    --          reset.reset_n
    sw_pio_external_connection_export => CONNECTED_TO_sw_pio_external_connection_export -- sw_pio_external_connection.export
  );
```

# NIOS I/O

- Create DE10 Design
  - Instantiate into a VHDL file

```
-----  
-- nios_io_de10.vhd1  
--  
-- Created 9/18/18  
-- by: johnsontimoj  
-- rev: 0  
--  
-----  
-- Basic Nios system - with led/sw peripherals  
--  
-----  
  
library ieee;  
use ieee.std_logic_1164.all;  
use ieee.numeric_std.all;  
  
entity nios_io_de10 is  
  port(  
    CLOCK_50 : in std_logic;  
    SW : in std_logic_vector(9 downto 0);  
    LEDR : out std_logic_vector(9 downto 0)  
  );  
end entity;
```

# NIOS I/O

- Create DE10 Design
  - Instantiate into a VHDL file

```
architecture behavioral of nios_io_de10 is

    component nios_io is
        port (
            clk_clk           : in  std_logic           := 'X';           -- clk
            led_pio_external_connection_export : out std_logic_vector(7 downto 0); -- export
            reset_reset_n     : in  std_logic           := 'X';           -- reset_n
            sw_pio_external_connection_export : in  std_logic_vector(1 downto 0) := (others => 'X') -- export
        );
    end component nios_io;

begin

    u0 : component nios_io
        port map (
            clk_clk           => CLOCK_50,           -- clk.clk
            led_pio_external_connection_export => LEDR(7 downto 0), -- led_pio_external_connection.export
            reset_reset_n     => '1',               -- reset.reset_n
            sw_pio_external_connection_export => SW(1 downto 0) -- sw_pio_external_connection.export
        );
end architecture;
```

# NIOS I/O

- Create DE10 Design
- Prepare to synthesize
  - If you did not do these when you created the project be sure to do them now
    - assignments → device → device and Pin options
      - Single Uncompressed with memory initialization
    - Import the pin aliases (qsf file)
    - Setup the SDF file
- Be sure to set your top level entity
- Start Compilation



# NIOS I/O

- Create DE10 Design
  - Complete the HW setup
    - Download the HW project onto the board
    - **DO NOT CLOSE** either of these windows

The screenshot displays the Altera Programmer software interface. The main window shows the hardware setup as USB-Blaster [USB-0] in JTAG mode, with a progress bar indicating 100% success. Below this is a table of programming operations:

File	Device	Checksum	Usercode	Program/Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP
output_files/NIOS...	10M50DAF484	004673E6	004673E6	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table is a diagram of the device (10M50DAF484) with TDI and TDO connections. An 'OpenCore Plus Status' dialog box is overlaid on the bottom right, displaying the message: 'Click Cancel to stop using OpenCore Plus IP.' and 'Time remaining: unlimited' with a 'Cancel' button.

# NIOS I/O

- Create Eclipse System
  - Open NIOSII software
    - [Tools](#) → [NIOSII Software Build Tools for Eclipse](#)
    - Select the project directory for the workspace
  - Create the BSP
    - [File](#) → [New](#) → [NIOSII Application and BSP from template](#)
    - Select the SOPCinfo file in the project directory
    - Provide a name for the sw project (I use 'project\_name\_sw')
    - [Blank Project](#)
  - Edit the BSP
    - Right click on the BSP, [NIOS II](#) → [BSP Editor](#)
    - Change the properties for small systems
      - Small C library
      - Reduced device drivers
  - Generate the BSP (bottom of window)

# NIOS I/O

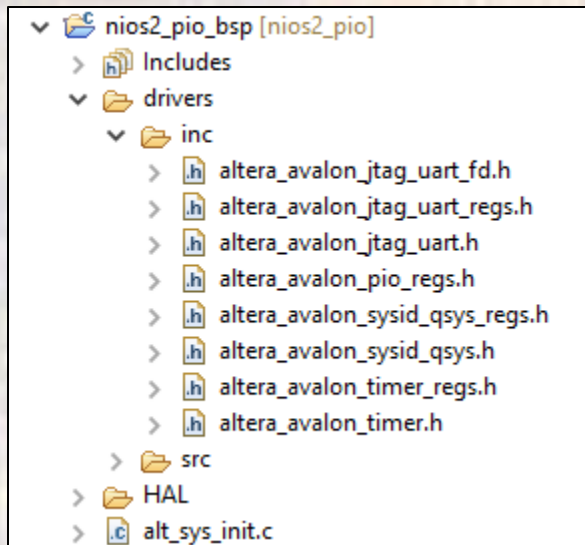
- Tutorial II
  - Open [system.h](#) in the bsp
    - scroll down to the PIOs
  - These define software parameters for the PIOs

```
221
222 #define ALT_MODULE_CLASS_led_pio altera_avalon_pio
223 #define LED_PIO_BASE 0x9030
224 #define LED_PIO_BIT_CLEARING_EDGE_REGISTER 0
225 #define LED_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
226 #define LED_PIO_CAPTURE 0
227 #define LED_PIO_DATA_WIDTH 8
228 #define LED_PIO_DO_TEST_BENCH_WIRING 0
229 #define LED_PIO_DRIVEN_SIM_VALUE 0
230 #define LED_PIO_EDGE_TYPE "NONE"
231 #define LED_PIO_FREQ 50000000
232 #define LED_PIO_HAS_IN 0
233 #define LED_PIO_HAS_OUT 1
234 #define LED_PIO_HAS_TRI 0
235 #define LED_PIO_IRQ -1
236 #define LED_PIO_IRQ_INTERRUPT_CONTROLLER_ID -1
237 #define LED_PIO_IRQ_TYPE "NONE"
238 #define LED_PIO_NAME "/dev/led_pio"
239 #define LED_PIO_RESET_VALUE 0
240 #define LED_PIO_SPAN 16
241 #define LED_PIO_TYPE "altera_avalon_pio"
242
```

```
277
278 #define ALT_MODULE_CLASS_sw_pio altera_avalon_pio
279 #define SW_PIO_BASE 0x9020
280 #define SW_PIO_BIT_CLEARING_EDGE_REGISTER 0
281 #define SW_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
282 #define SW_PIO_CAPTURE 0
283 #define SW_PIO_DATA_WIDTH 2
284 #define SW_PIO_DO_TEST_BENCH_WIRING 0
285 #define SW_PIO_DRIVEN_SIM_VALUE 0
286 #define SW_PIO_EDGE_TYPE "NONE"
287 #define SW_PIO_FREQ 50000000
288 #define SW_PIO_HAS_IN 1
289 #define SW_PIO_HAS_OUT 0
290 #define SW_PIO_HAS_TRI 0
291 #define SW_PIO_IRQ -1
292 #define SW_PIO_IRQ_INTERRUPT_CONTROLLER_ID -1
293 #define SW_PIO_IRQ_TYPE "NONE"
294 #define SW_PIO_NAME "/dev/sw_pio"
295 #define SW_PIO_RESET_VALUE 0
296 #define SW_PIO_SPAN 16
297 #define SW_PIO_TYPE "altera_avalon_pio"
298
```

# NIOS I/O

- Tutorial II
  - Expand **drivers** → **inc** in the bsp
    - Open **altera\_Avalon\_pio\_regs.h**
  - PIO command are defined here

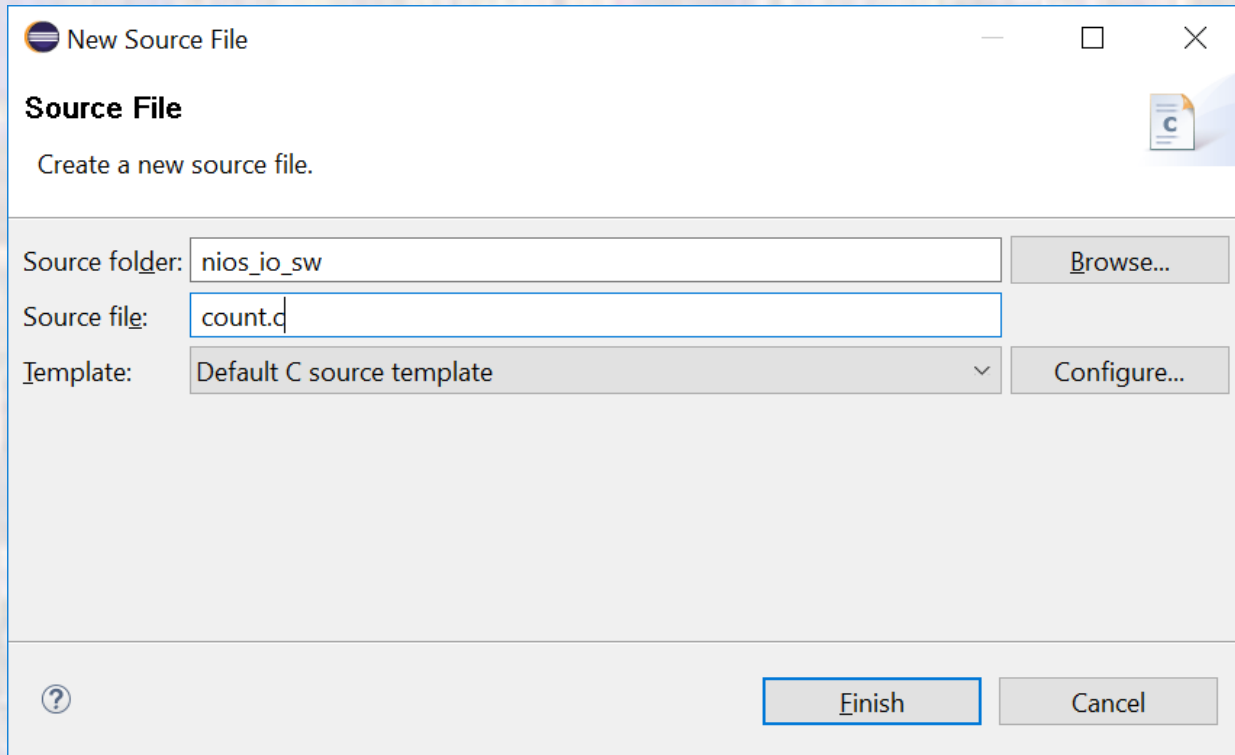


```
30
31 #ifndef __ALTERA_AVALON_PIO_REGS_H__
32 #define __ALTERA_AVALON_PIO_REGS_H__
33
34 #include <io.h>
35
36 #define IORD_ALTERA_AVALON_PIO_DATA(base)      __IO_CALC_ADDRESS_NATIVE(base, 0)
37 #define IORD_ALTERA_AVALON_PIO_DATA(base)      IORD(base, 0)
38 #define IOWR_ALTERA_AVALON_PIO_DATA(base, data) IOWR(base, 0, data)
39
40 #define IOADDR_ALTERA_AVALON_PIO_DIRECTION(base) __IO_CALC_ADDRESS_NATIVE(base, 1)
41 #define IORD_ALTERA_AVALON_PIO_DIRECTION(base)   IORD(base, 1)
42 #define IOWR_ALTERA_AVALON_PIO_DIRECTION(base, data) IOWR(base, 1, data)
43
44 #define IOADDR_ALTERA_AVALON_PIO_IRQ_MASK(base) __IO_CALC_ADDRESS_NATIVE(base, 2)
45 #define IORD_ALTERA_AVALON_PIO_IRQ_MASK(base)   IORD(base, 2)
46 #define IOWR_ALTERA_AVALON_PIO_IRQ_MASK(base, data) IOWR(base, 2, data)
47
48 #define IOADDR_ALTERA_AVALON_PIO_EDGE_CAP(base) __IO_CALC_ADDRESS_NATIVE(base, 3)
49 #define IORD_ALTERA_AVALON_PIO_EDGE_CAP(base)   IORD(base, 3)
50 #define IOWR_ALTERA_AVALON_PIO_EDGE_CAP(base, data) IOWR(base, 3, data)
51
52
53 #define IOADDR_ALTERA_AVALON_PIO_SET_BIT(base) __IO_CALC_ADDRESS_NATIVE(base, 4)
54 #define IORD_ALTERA_AVALON_PIO_SET_BITS(base)  IORD(base, 4)
55 #define IOWR_ALTERA_AVALON_PIO_SET_BITS(base, data) IOWR(base, 4, data)
56
57 #define IOADDR_ALTERA_AVALON_PIO_CLEAR_BITS(base) __IO_CALC_ADDRESS_NATIVE(base, 5)
58 #define IORD_ALTERA_AVALON_PIO_CLEAR_BITS(base)  IORD(base, 5)
59 #define IOWR_ALTERA_AVALON_PIO_CLEAR_BITS(base, data) IOWR(base, 5, data)
60
61
62
63 /* Definitions for direction-register operation with Bi-directional Pios */
64 #define ALTERA_AVALON_PIO_DIRECTION_INPUT 0
65 #define ALTERA_AVALON_PIO_DIRECTION_OUTPUT 1
66
67 #endif /* __ALTERA_AVALON_PIO_REGS_H__ */
68
```



# NIOS I/O

- Tutorial II
  - Create a new c file in the nios\_io\_sw project



# NIOS I/O

- Tutorial II
  - Write a program to read the switches and display a count to the LEDs

```
/*
 *
 * count.c
 *
 * Created on: Sep 19, 2018
 * Author: johnsontimoj
 */
//
// Example file using
// NIOS with LEDs and switches
//
//
//
//
#include "altera_avalon_pio_regs.h"
#include "system.h"
#include <stdio.h>
#include <unistd.h>

int main(){
    printf("My count program!\n");
    alt_u8 count = 0;
    alt_u8 sw;

    while(1){
        // output the count to the LEDs
        IOWR_ALTERA_AVALON_PIO_DATA(LED_PIO_BASE, count);

        // read the switches
        sw = IORD_ALTERA_AVALON_PIO_DATA(SW_PIO_BASE);

        // count up/dn/pause
        if(sw & 0x01){ // pause
            if(!(sw & 0x02)) // up/dn
                count++;
            else
                count--;
            printf("%02x, ", count);
            usleep(500000);
        } // end if

        // delay before restarting
        if( count == 0xff ){
            printf("\nWaiting...");
            int i;
            for (i = 0; i<6; ++i){
                usleep(500000); /* Sleep for 3s. */
            }
        } // end if
    } // end while
    return 0;
} // end main
```

# NIOS I/O

- Tutorial II
  - Compile the software
    - Select the code file (count.c)
    - Project → Build Project
  - Right Click on the project → run as → Nios II Hardware