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

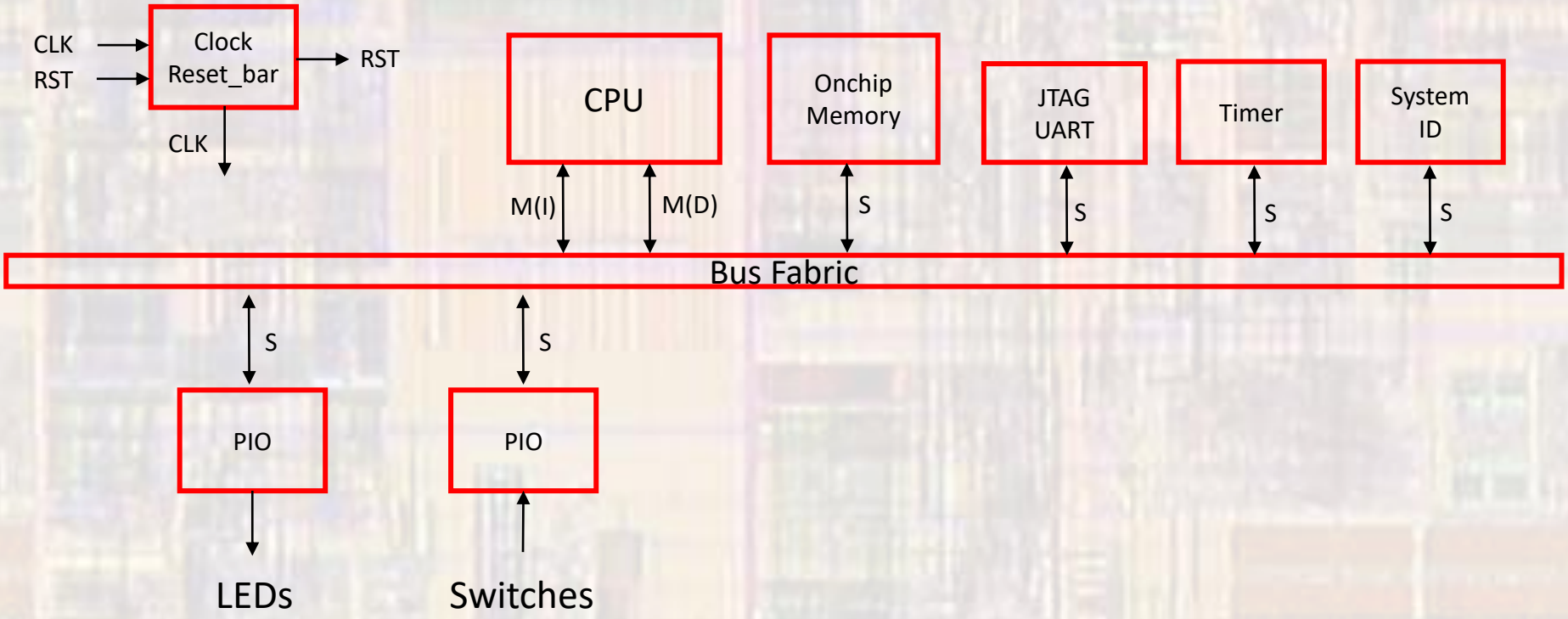
Upon completion: You should be able implement your own NIOS processor with basic I/O and write code for it

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

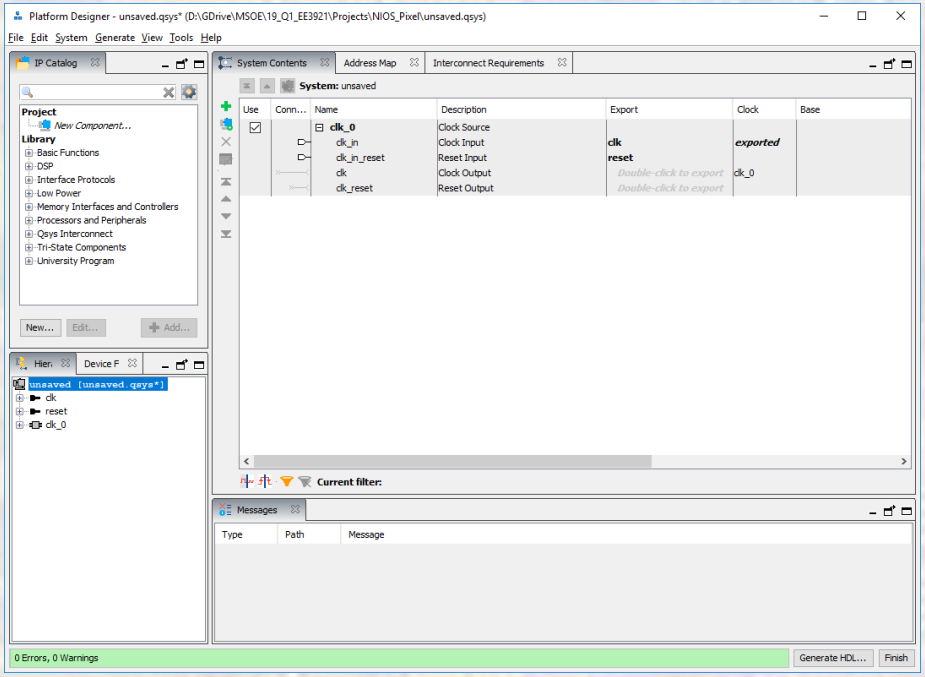
- NIOS I/O System
  - Create a processor system to display a count on LEDs with switch control for up/dn, pause



NIOS I/O

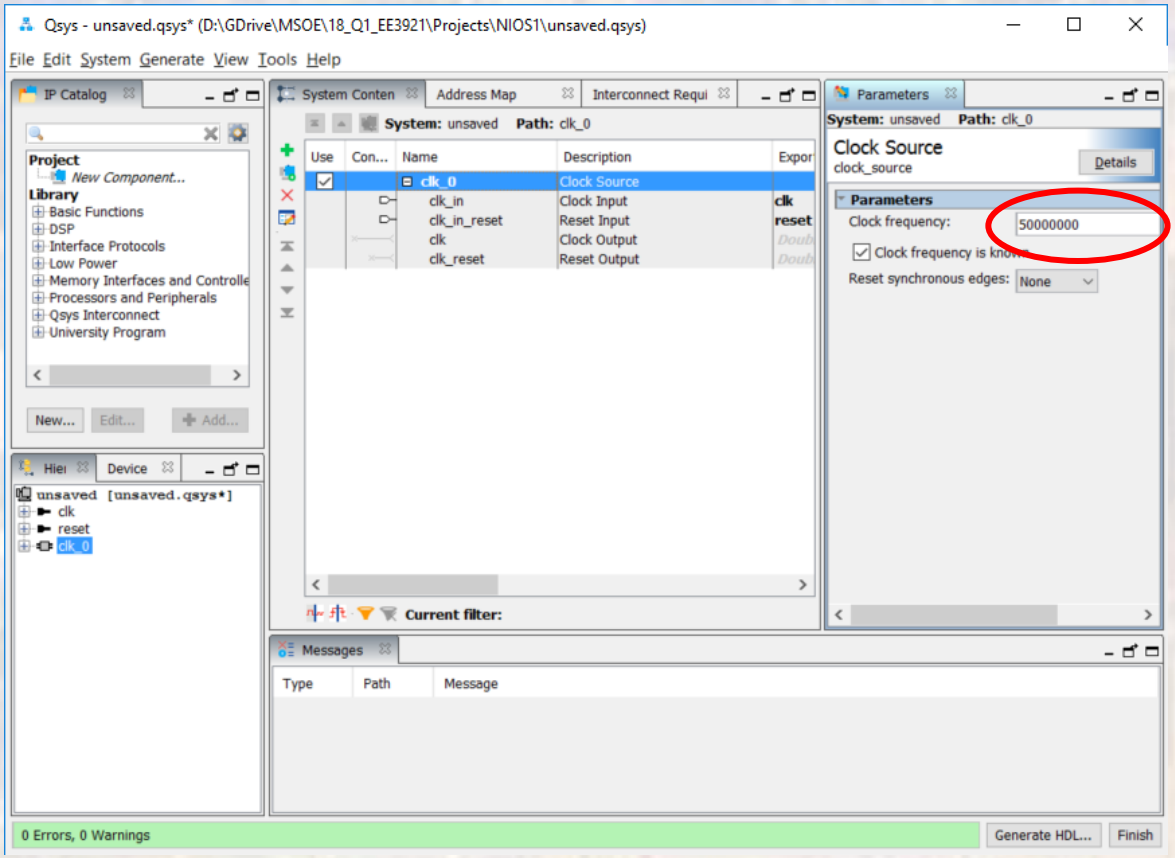
- Create a new Quartus project
  - Do not select a Simulation Tool in EDA Tool Settings

- Open Tools → Platform Designer



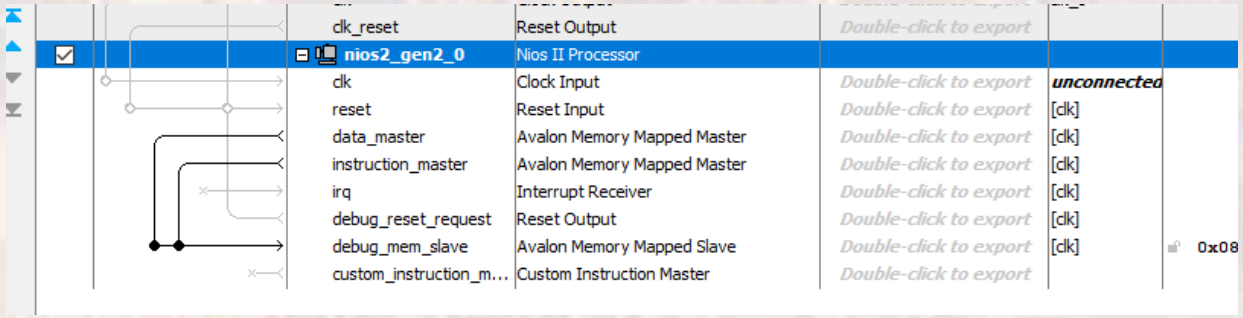
NIOS I/O

- Create NIOS System
  - Double Click on clk_0 - verify clk frequency = 50MHz



NIOS I/O

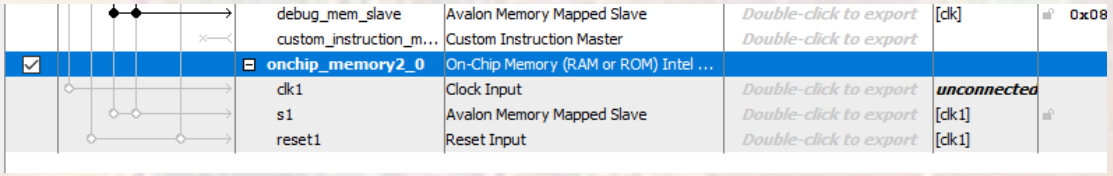
- Add NIOS
  - Processors and Peripherals → Embedded Processors → NIOS II Processor
  - NIOS II/f



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

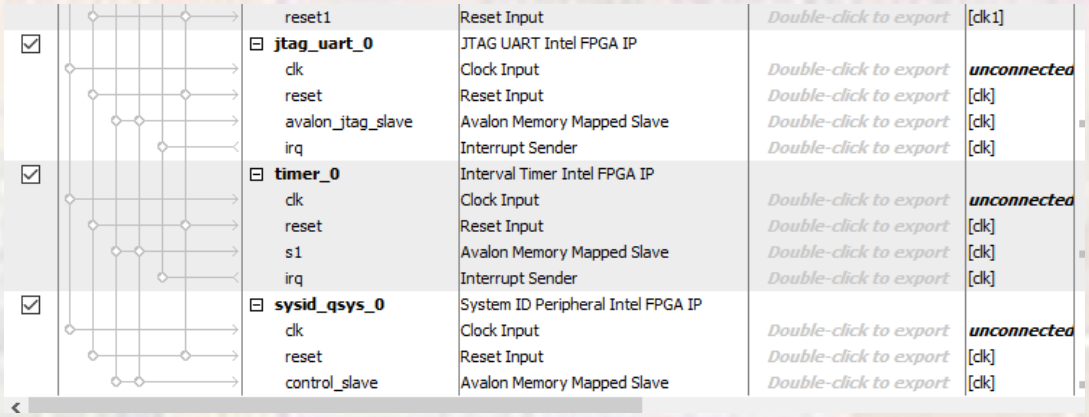
    RAM

    Size = 20,000 bytes



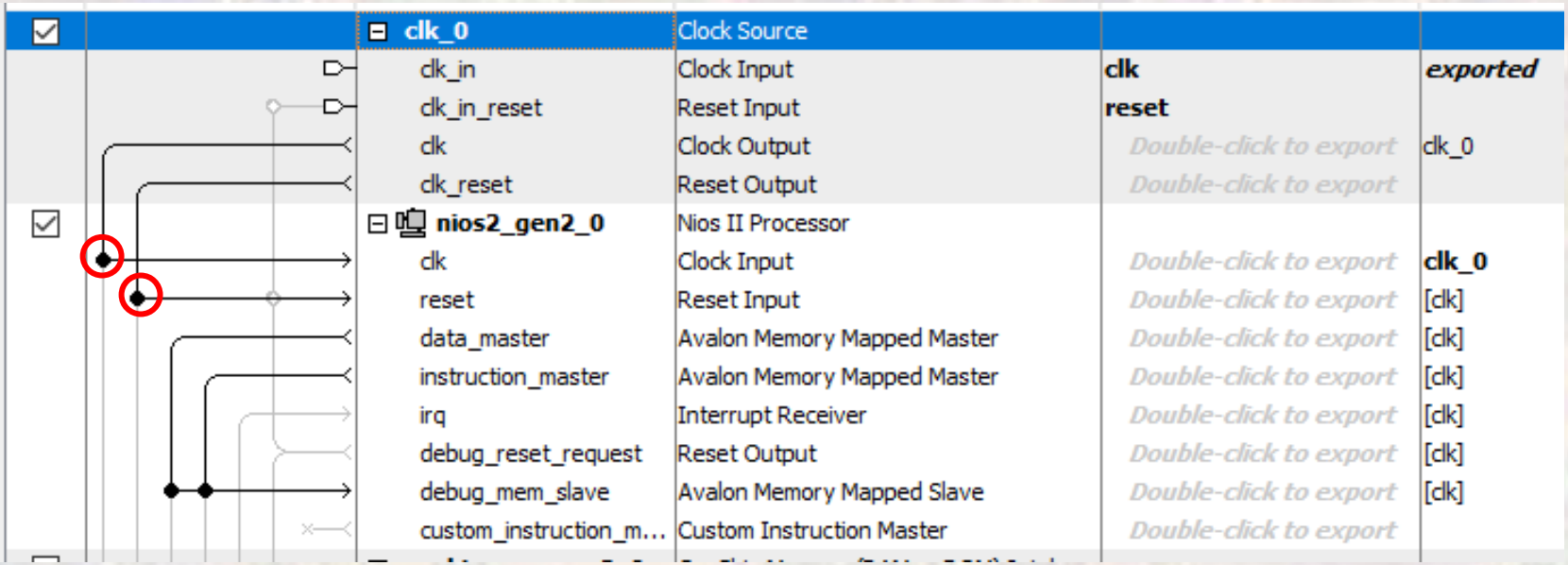
NIOS I/O

- 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



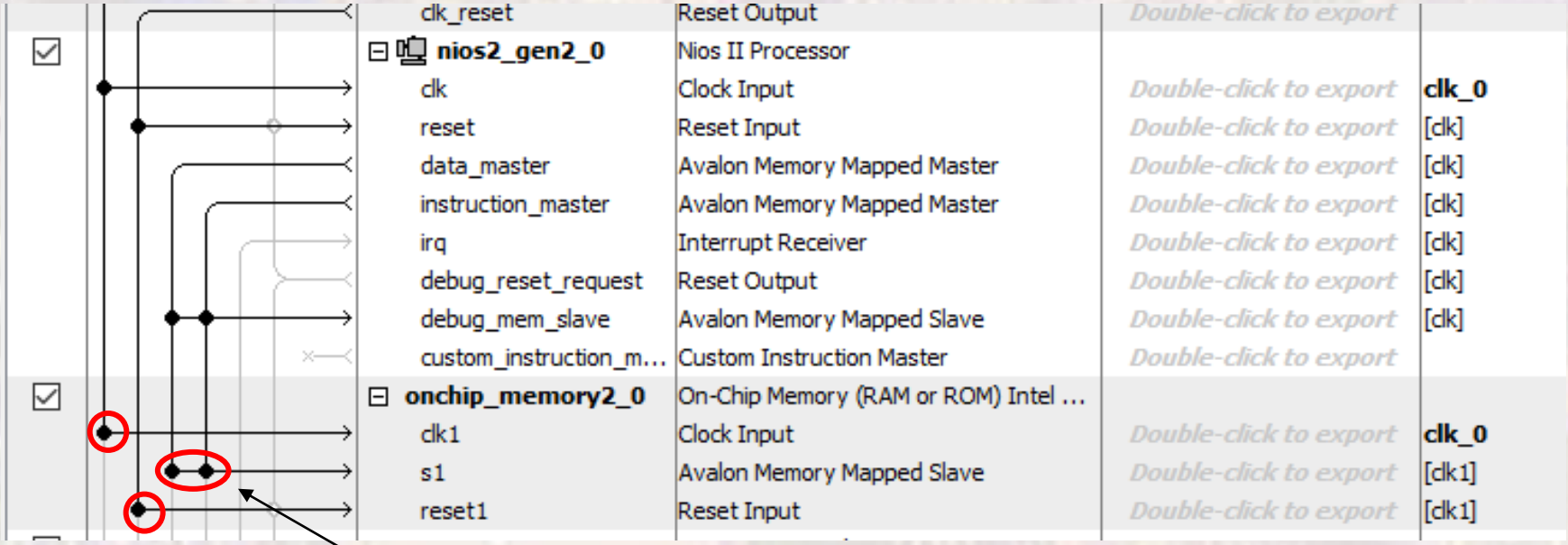
NIOS I/O

- Connect up NIOS I/O
  - NIOS Inputs



NIOS I/O

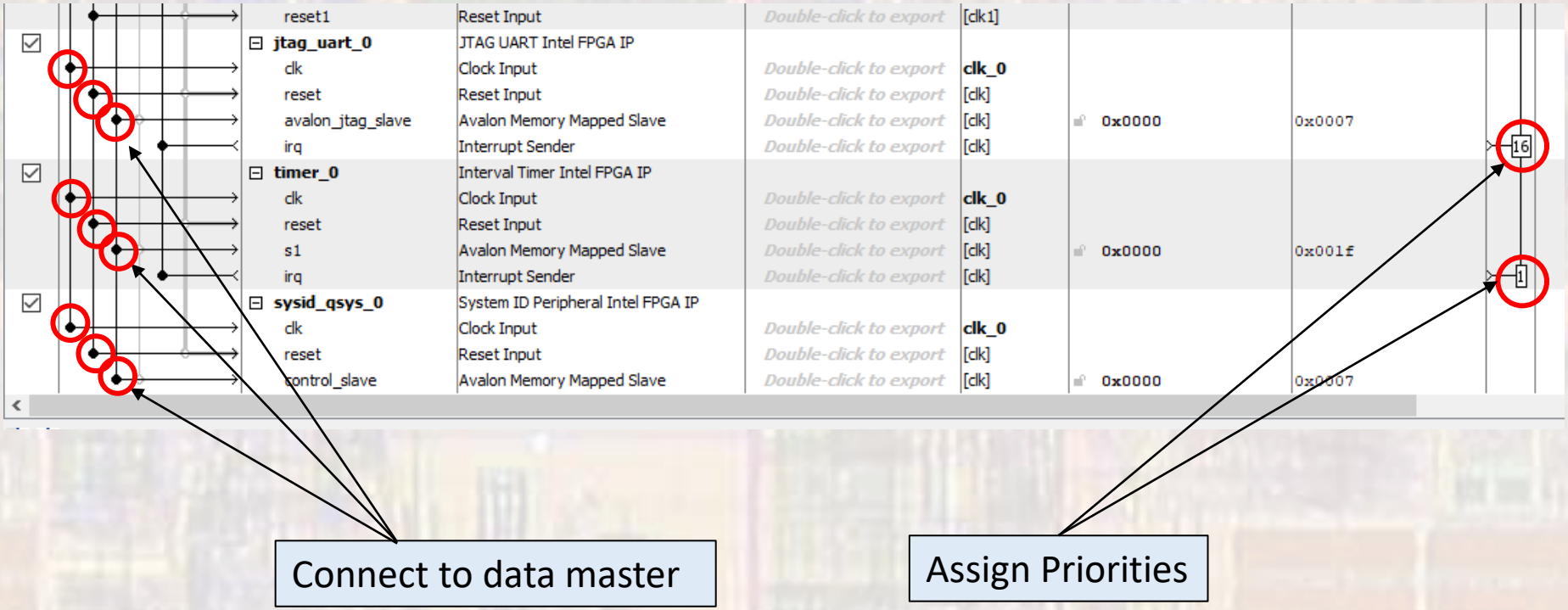
- Connect up NIOS I/O
  - On-chip Memory



Connect to data and instruction masters

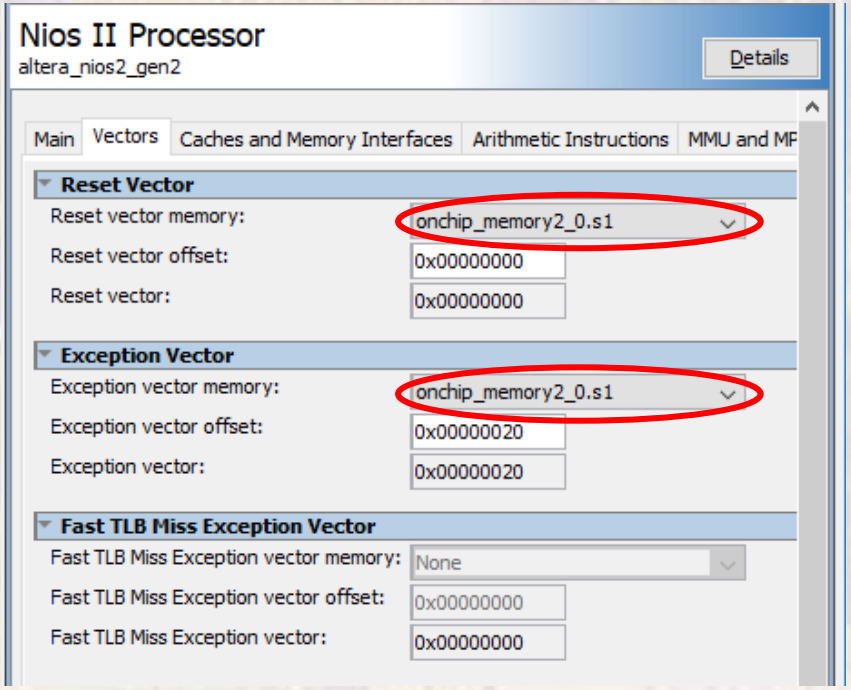
NIOS I/O

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



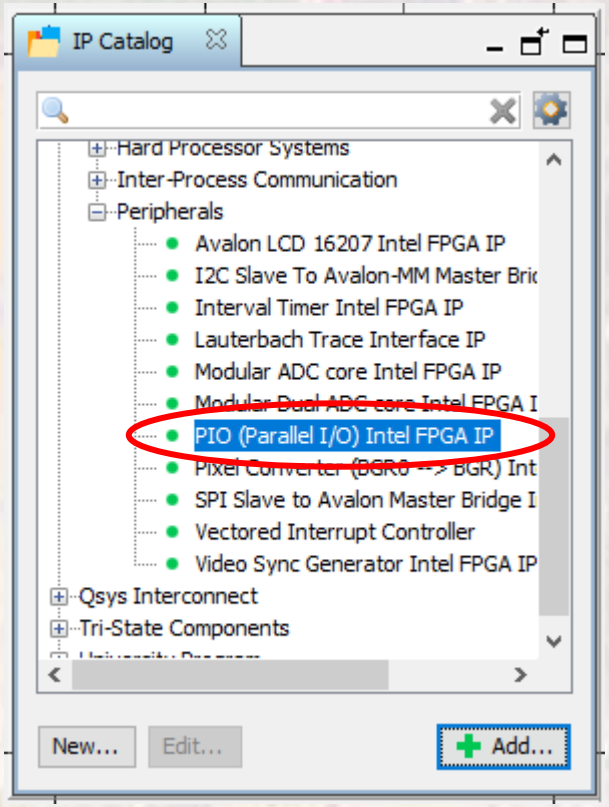
NIOS I/O

- 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



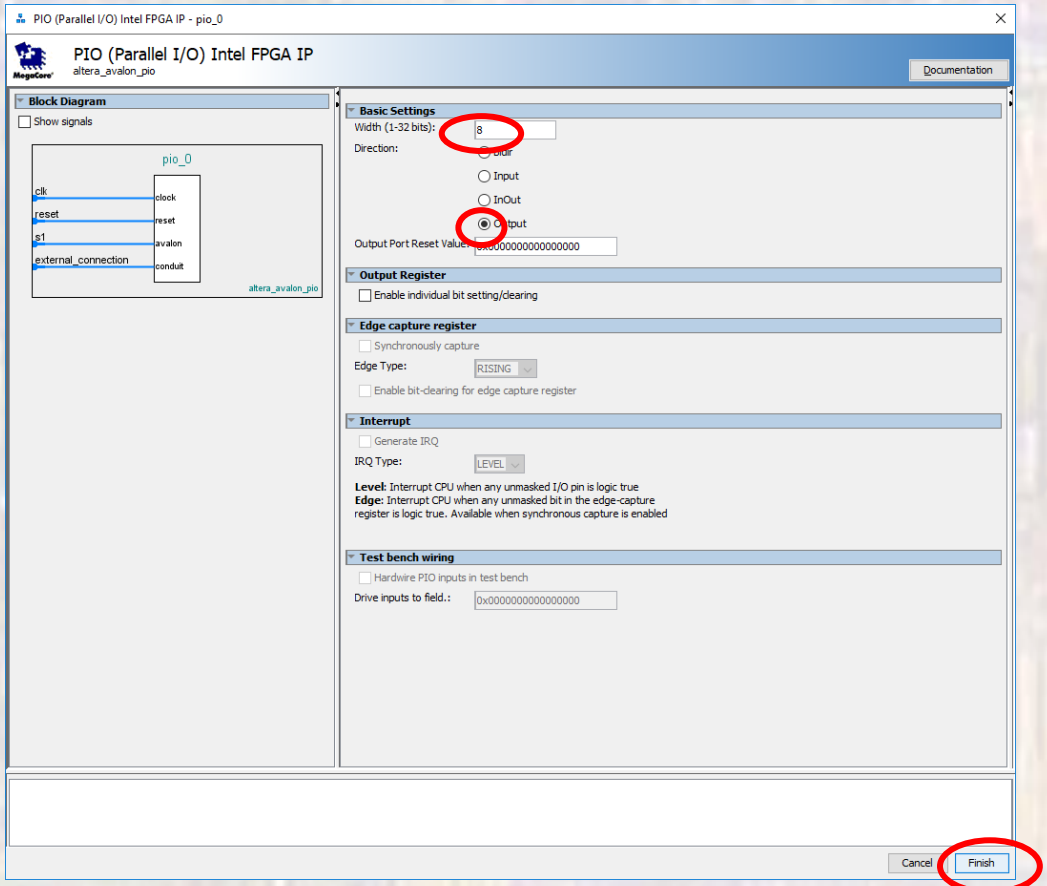
NIOS I/O

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



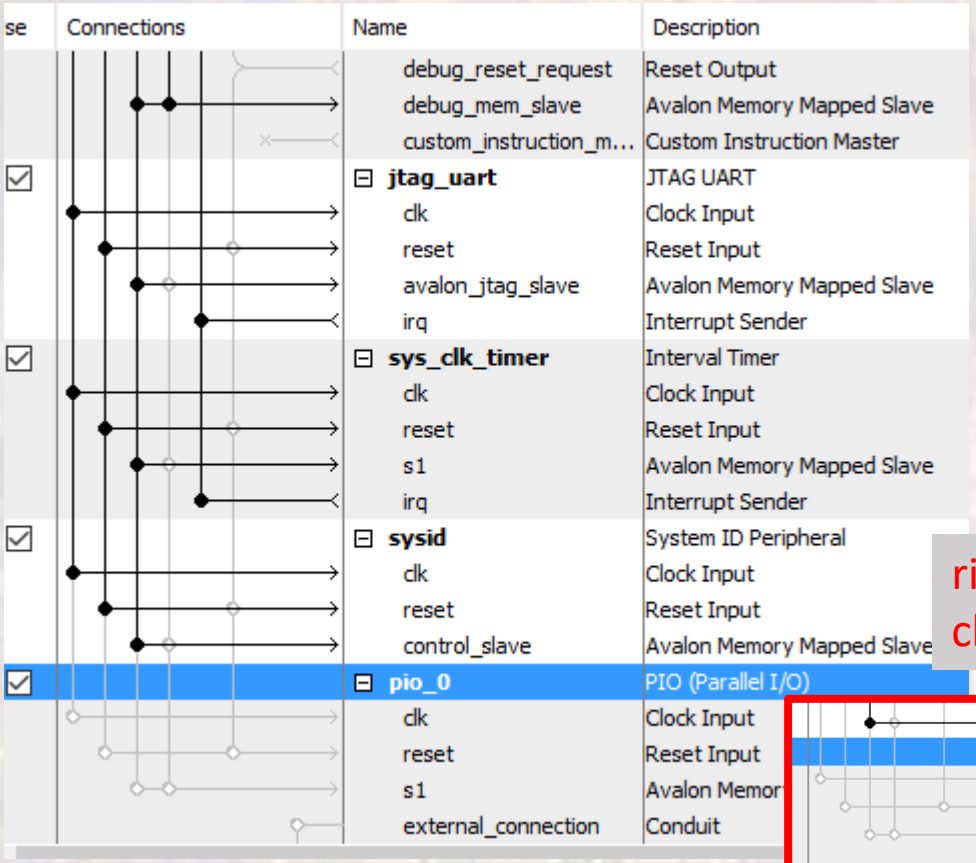
NIOS I/O

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

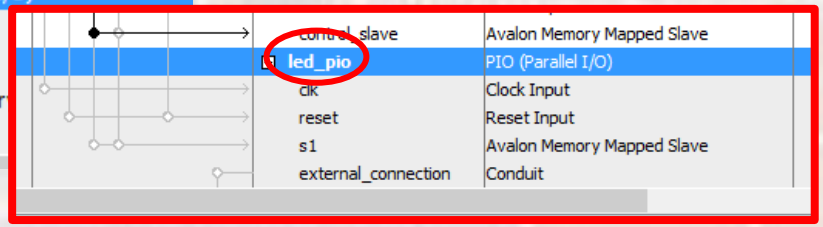


NIOS I/O

- Customize System
  - Hookup LED output

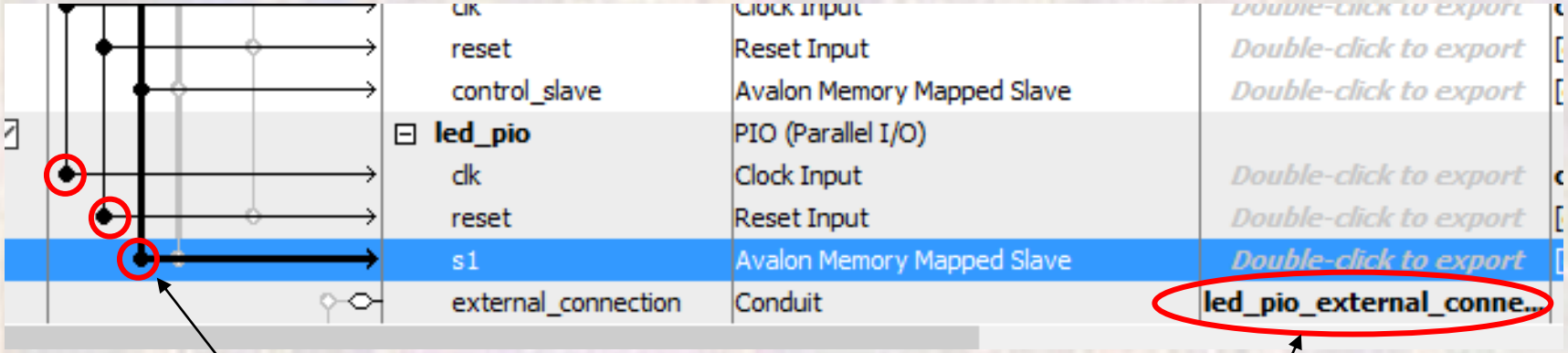


right click the PIO name and change it to led_pio



NIOS I/O

- Customize System
  - Hookup LED output



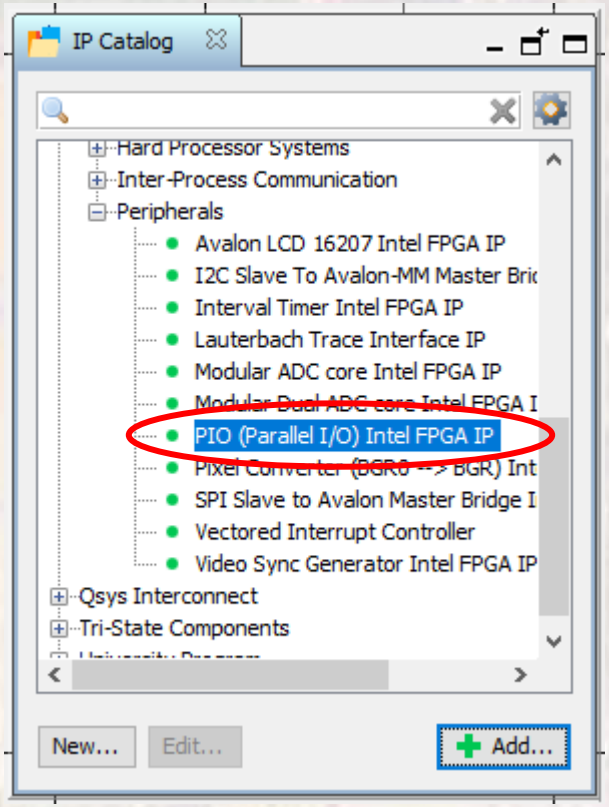
Connect to data master

Double click to export

Makes the connection visible outside the system

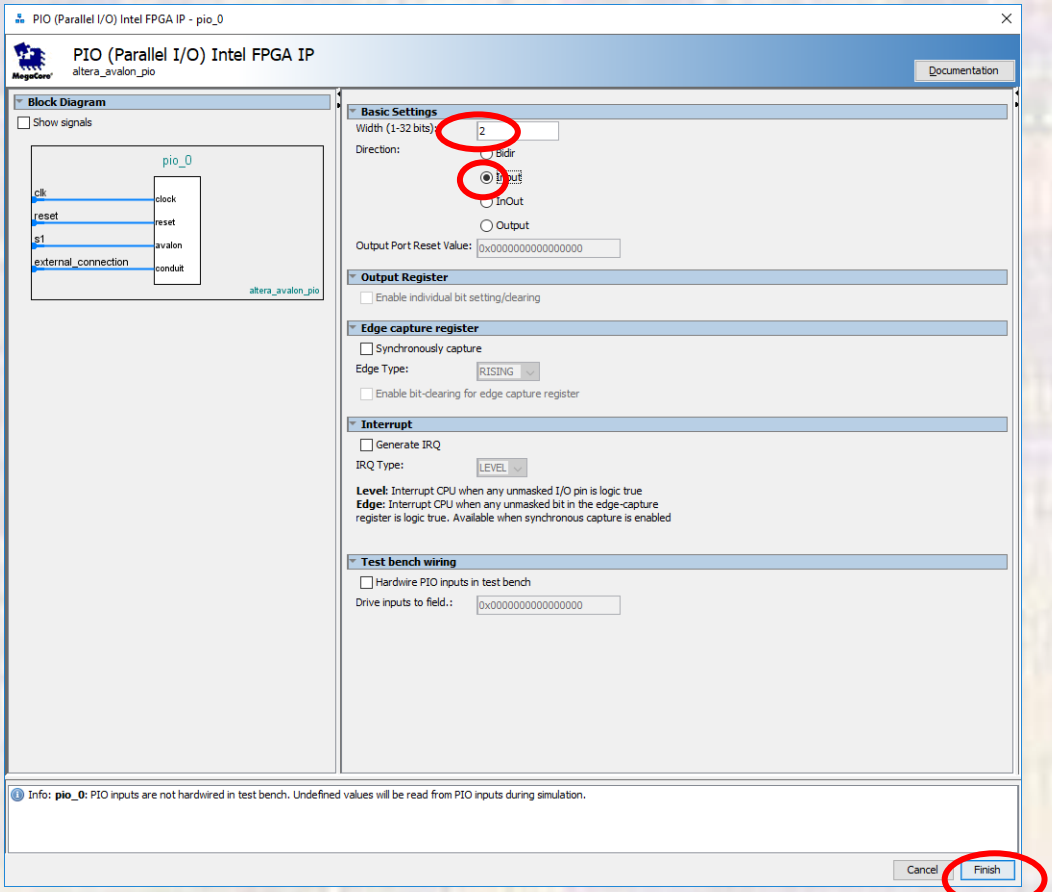
NIOS I/O

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



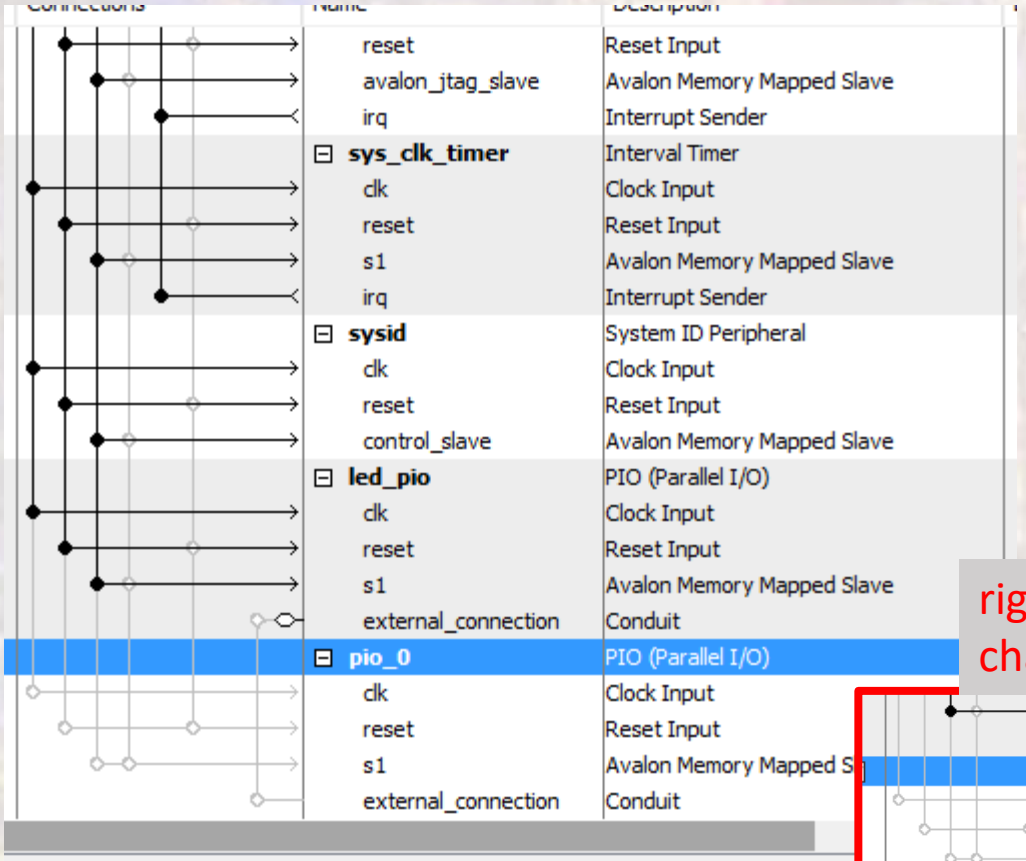
NIOS I/O

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

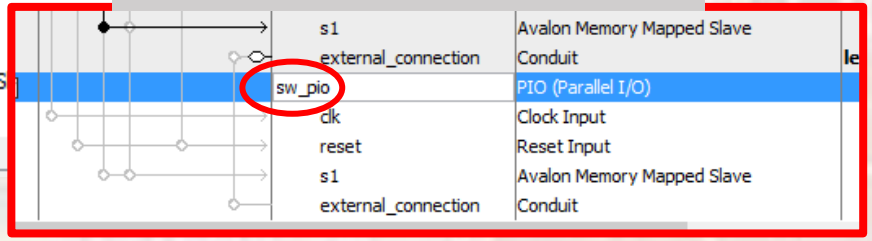


NIOS I/O

- Customize System
  - Hookup switch input

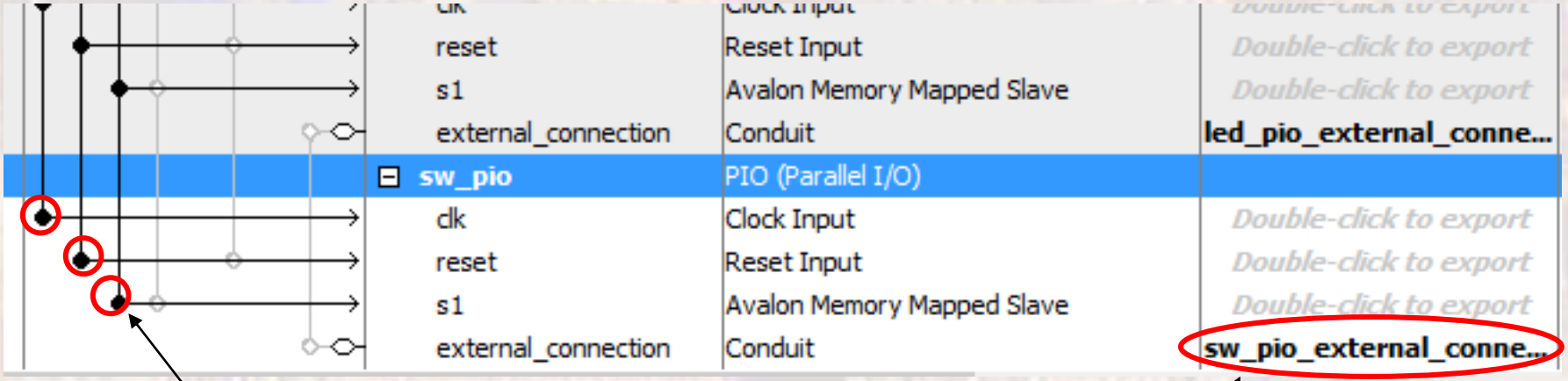


right click the PIO name and change it to sw_pio



NIOS I/O

- Customize System
  - Hookup switch input



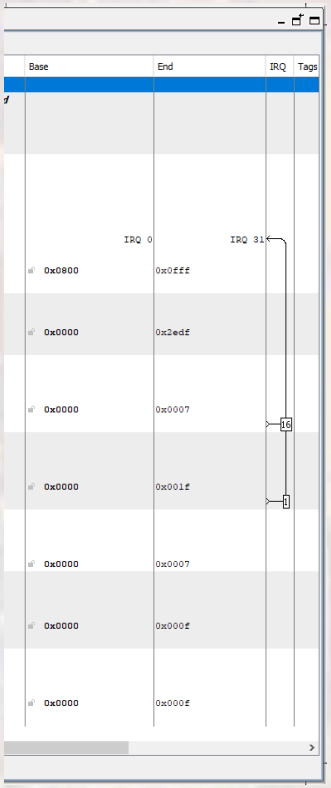
Connect to data master

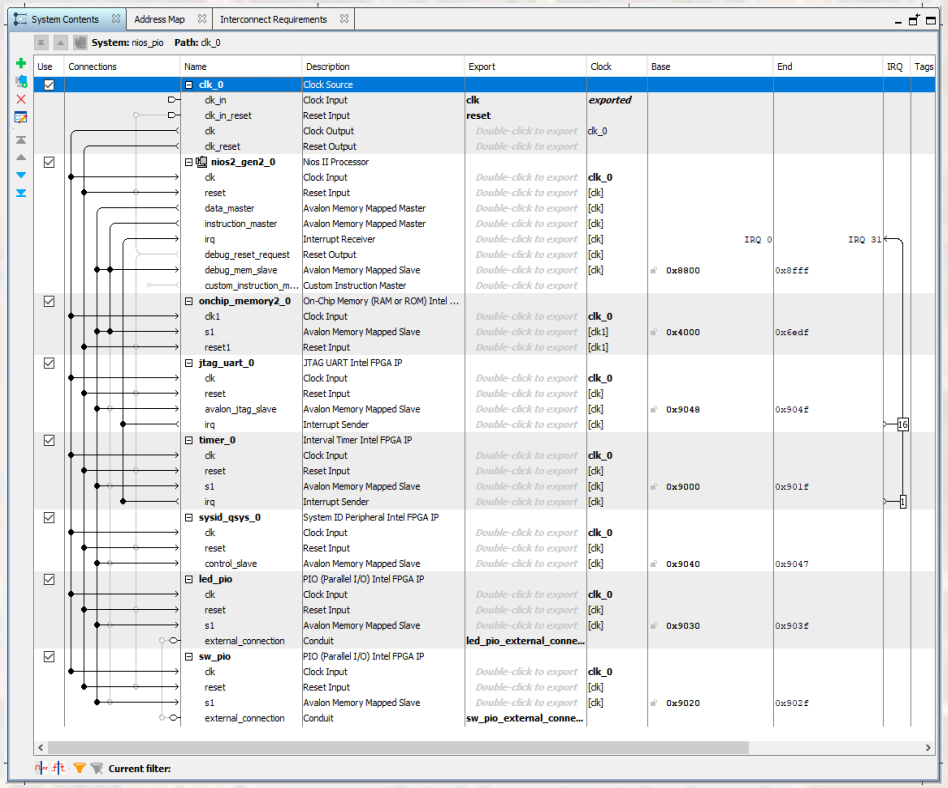
Double click to export

Makes the connection visible outside the system

NIOS I/O

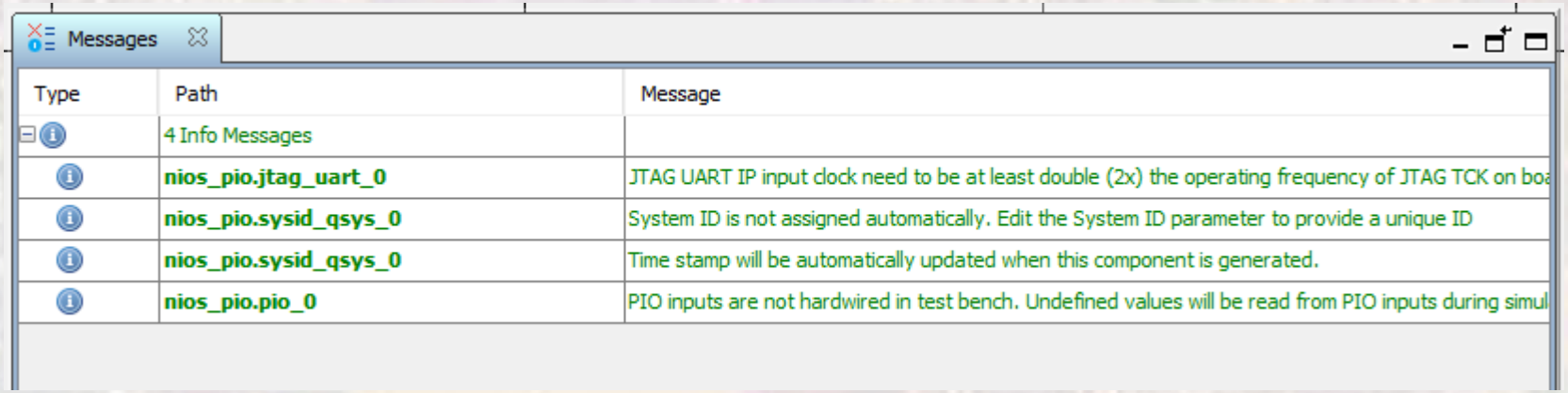
- Complete Custom System
  - Assign base addresses
  - System → Assign Base Addresses





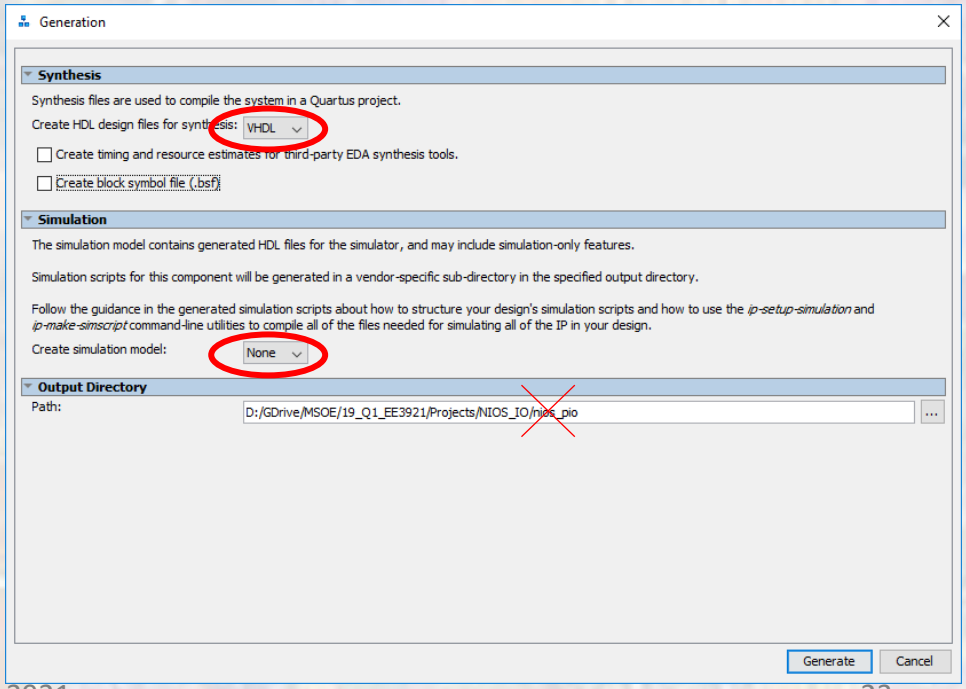
NIOS I/O

- Create Custom System
  - Check for errors

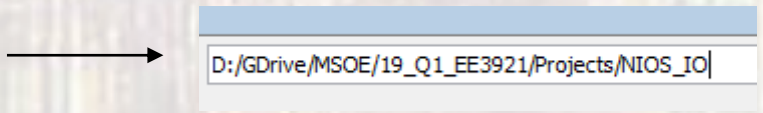


NIOS I/O

- 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 '…'

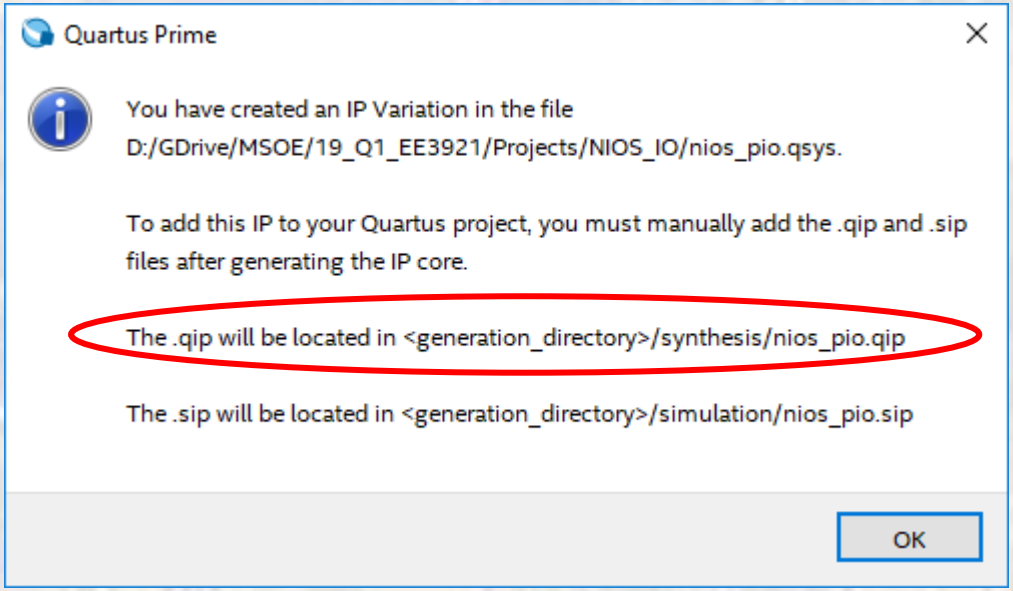


Should point to your project directory



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.vhdl
--
-- 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

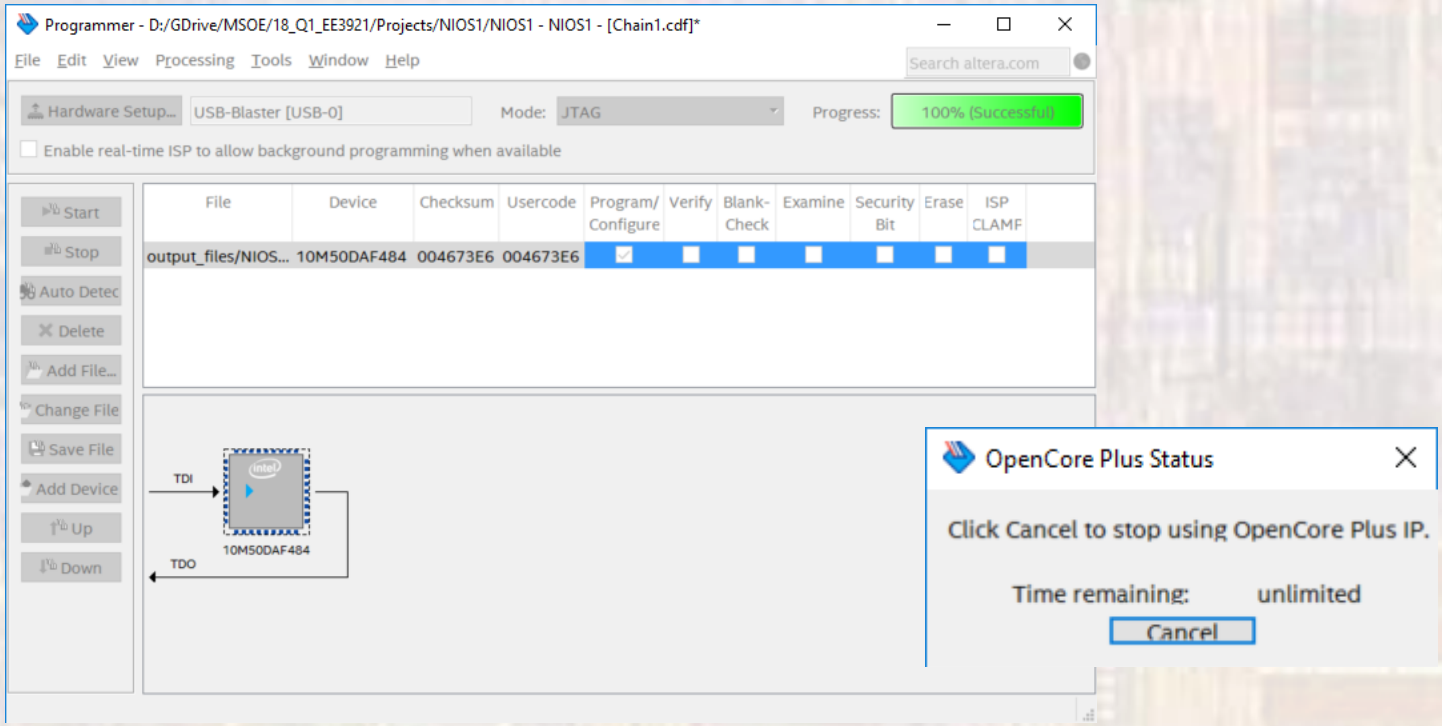
```
architecture behavioral of nios_io_de10 is


    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



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

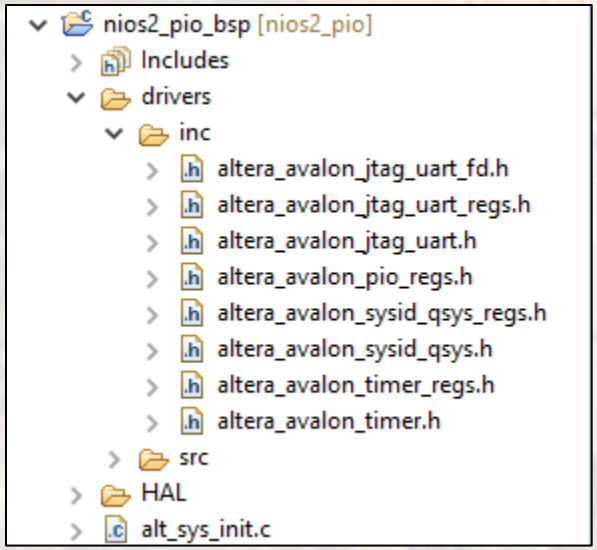
- Tutorial II
  - Open system.h in the bsp
    - scroll down to the PIOs
    - These define software parameters for the PIOs

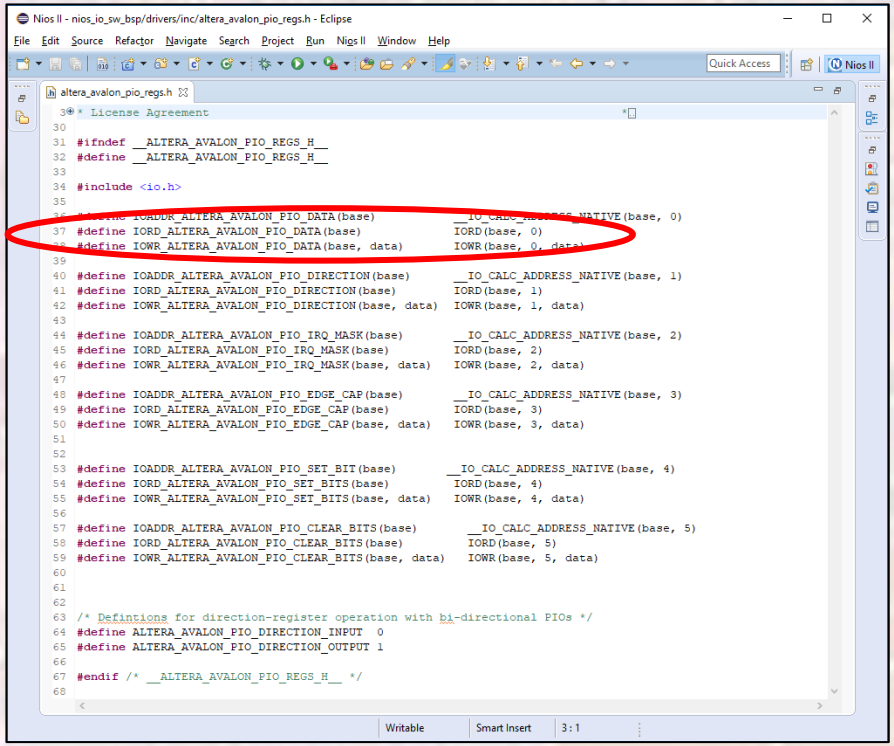
```
221
222 #define ALT_MODULE_CLASS_led_pio altera_avalon_pio
223 #define LED_PIO_BASE 0x9030
224 #define LED_PIO_BIT_CLEARING_EDGE_REGISTER 0
225 #define LED_PIO_BIT_MODIFYING_OUTPUT_REGISTER 0
226 #define LED_PIO_CAPTURE 0
227 #define LED_PIO_DATA_WIDTH 8
228 #define LED_PIO_DO_TEST_BENCH_WIRING 0
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

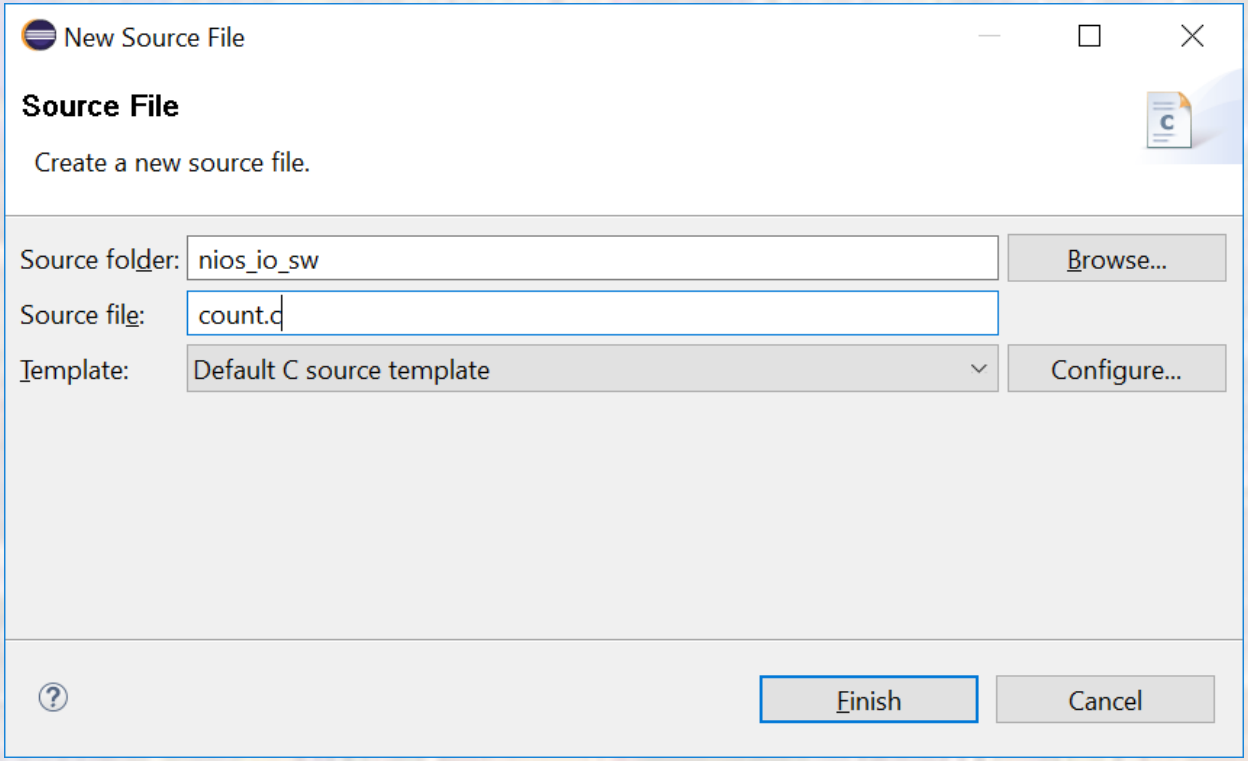
- Tutorial II
  - Expand drivers → inc in the bsp
    - Open altera_Avalon_pio_regs.h
    - PIO command are defined here





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