NIOS Intro

Last updated 10/12/20

NIOS Basic

These slides describe the development of a simple NIOS Processor

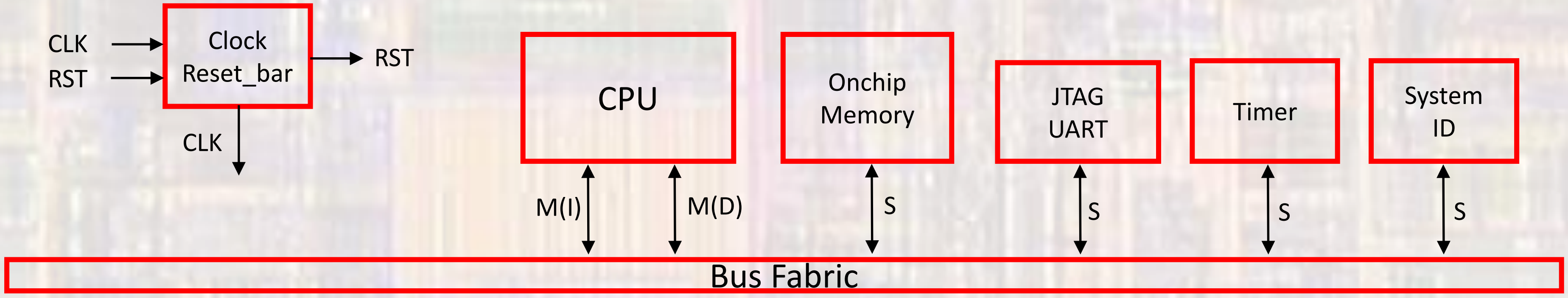
Upon completion: You should be able implement your own NIOS processor and write code for it

NIOS

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

Basic NIOS System

- Basic NIOS System
  - Create a processor system to allow printing to the console



Basic NIOS System

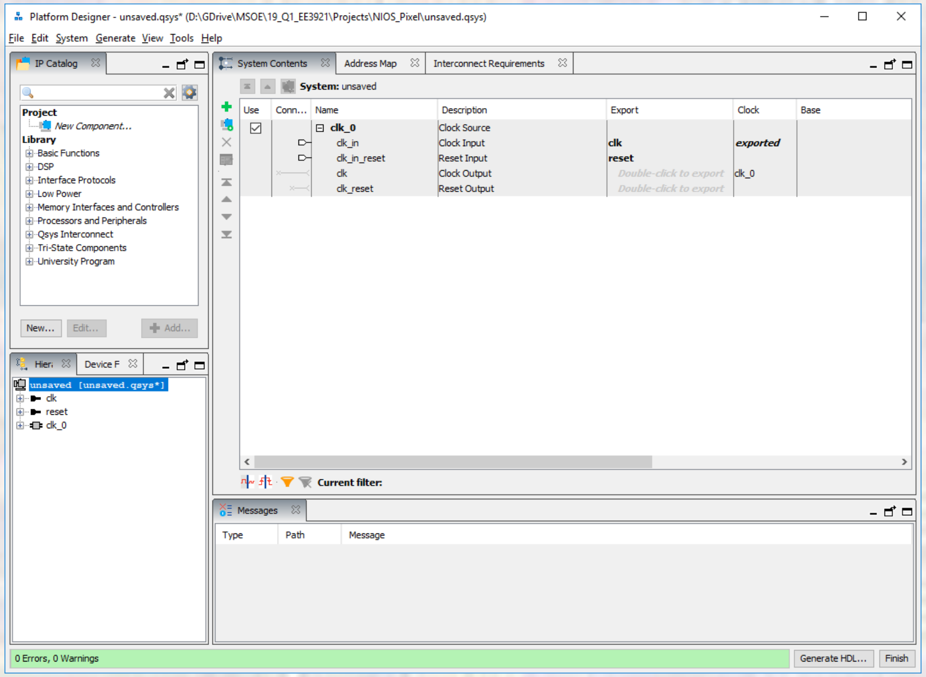
- Basic NIOS System

HARDWARE

Basic NIOS System

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

- Open Tools → Platform Designer



Platform Designer - unsaved.qsys* (D:\GDrive\MSOE\19_Q1_EE3921\Projects\NIOS_Pixel\unsaved.qsys)

System: unsaved

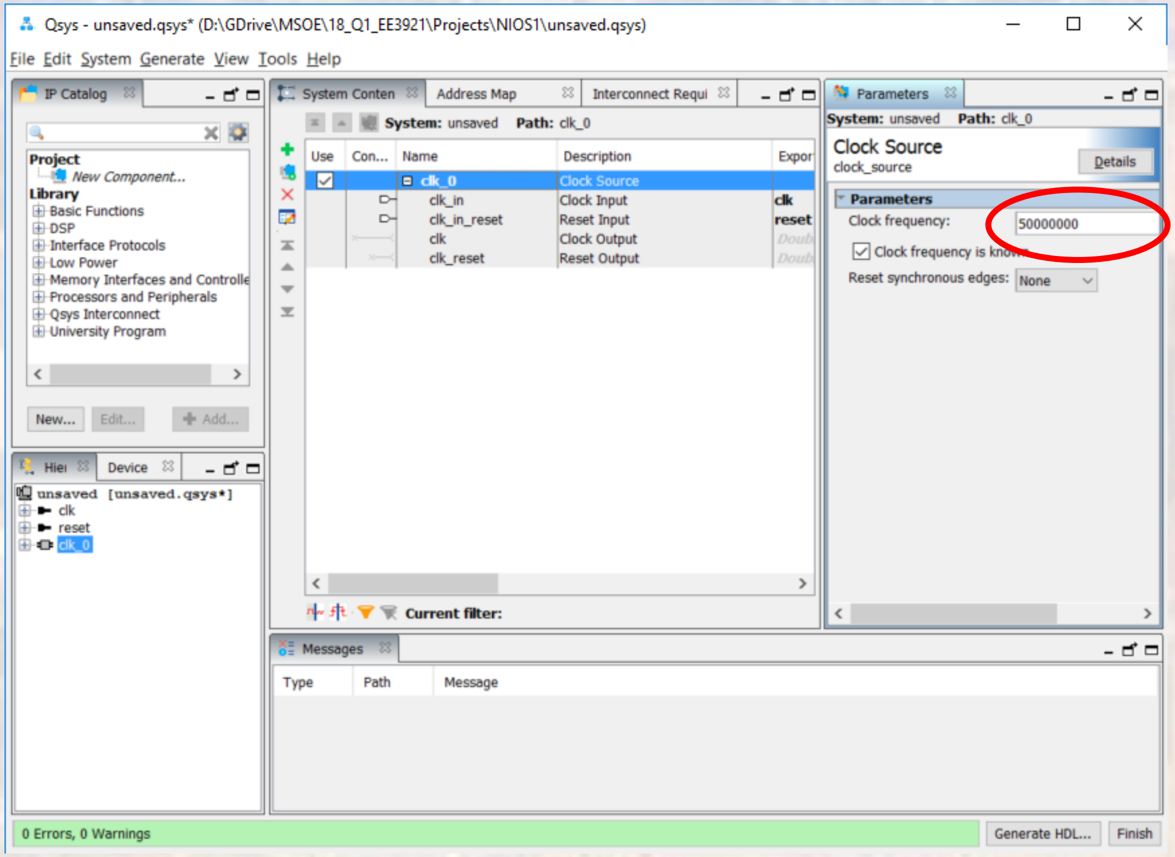
Use	Conn…	Name	Description	Export	Clock	Base
✓		clk_0	Clock Source			
		clk_in	Clock Input	clk	exported	
		clk_in_reset	Reset Input	reset		
		clk	Clock Output	Double-click to export	clk_0	
		clk_reset	Reset Output	Double-click to export		

unsaved [unsaved.qsys*]
clk
reset
clk_0

0 Errors, 0 Warnings

Basic NIOS System

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



Qsys - unsaved.qsys* (D:\GDrive\MSOE\18_Q1_EE3921\Projects\NIOS1\unsaved.qsys)

System: unsaved Path: clk_0

Use	Con...	Name	Description	Expor
✓		clk_0	Clock Source	
		clk_in	Clock Input	clk
		clk_in_reset	Reset Input	reset
		clk	Clock Output	Doub
		clk_reset	Reset Output	Doub

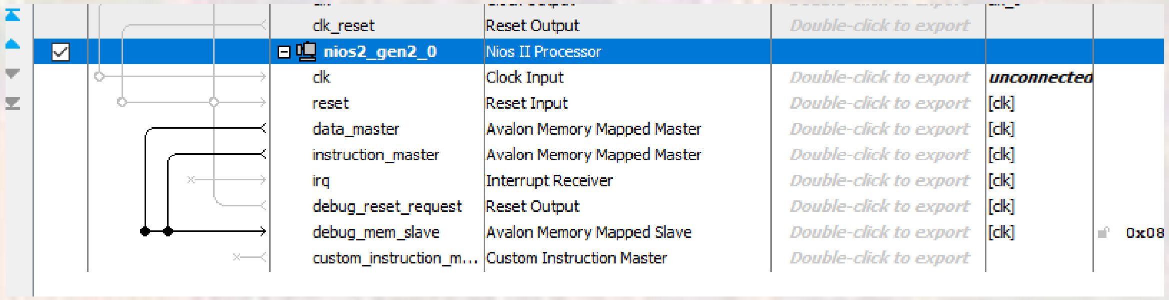
Clock Source
clock_source

Parameters
Clock frequency: 50000000
✓ Clock frequency is known
Reset synchronous edges: None

0 Errors, 0 Warnings

Basic NIOS System

- Add NIOS
  - Processors and Peripherals → Embedded Processors → NIOS II Processor
  - NIOS II/f
  - No other changes for now

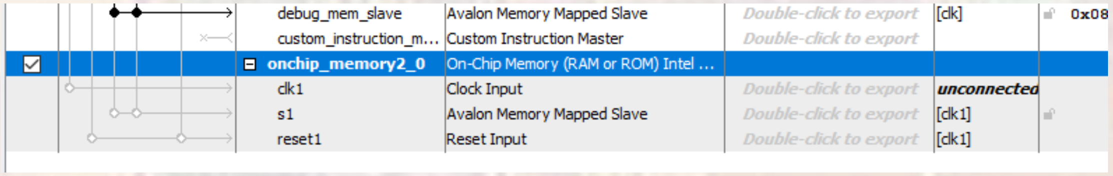


clk_reset	Reset Output	Double-click to export		
nios2_gen2_0	Nios II Processor			
clk	Clock Input	Double-click to export	*unconnected*	
reset	Reset Input	Double-click to export	[clk]	
data_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]	
irq	Interrupt Receiver	Double-click to export	[clk]	
debug_reset_request	Reset Output	Double-click to export	[clk]	
debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x08
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

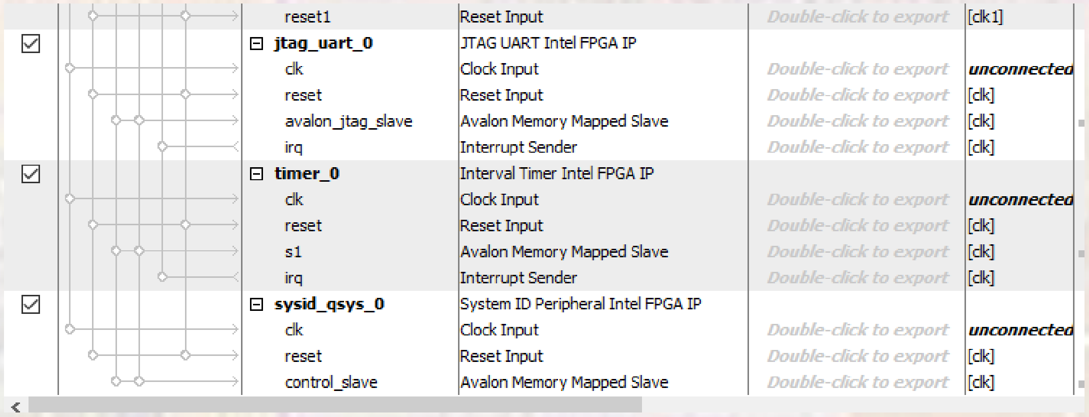
    Size = 12,000 bytes



debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x08
custom_instruction_m...	Custom Instruction Master	Double-click to export		
onchip_memory2_0	On-Chip Memory (RAM or ROM) Intel ...			
clk1	Clock Input	Double-click to export	*unconnected*	
s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	
reset1	Reset Input	Double-click to export	[clk1]	

Basic NIOS System

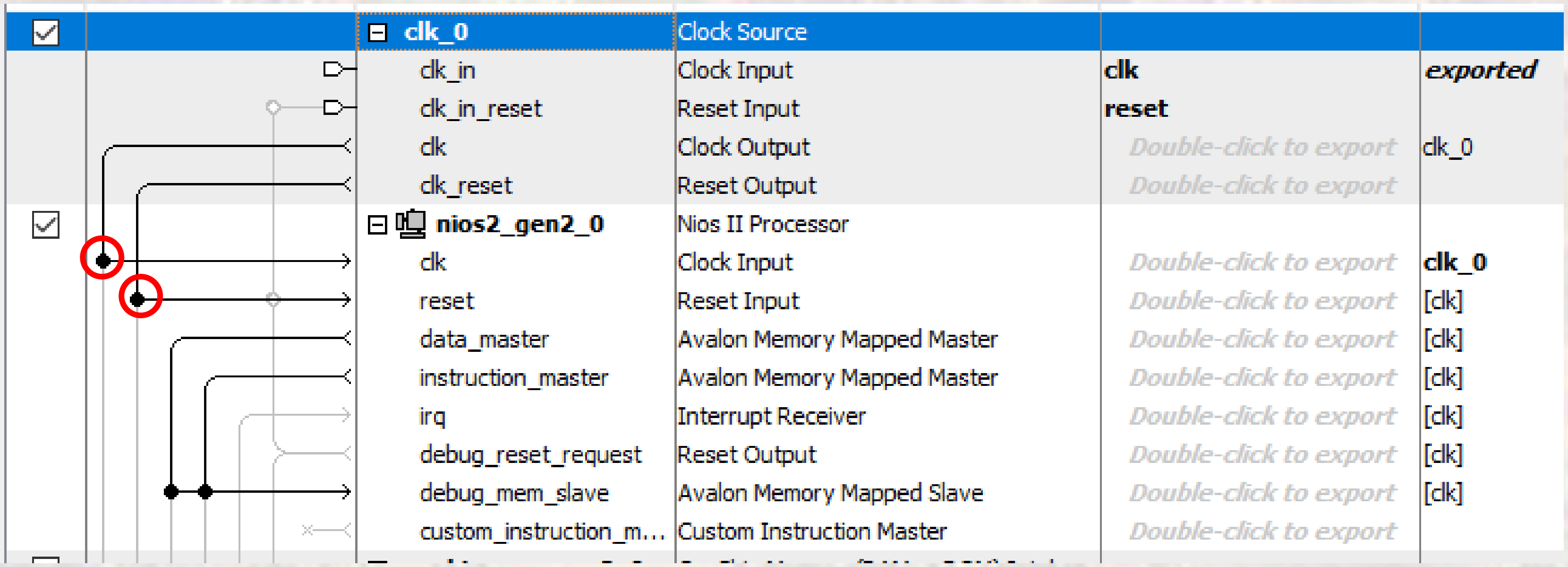
- Add JTAG
  - Interface Protocols → Serial→ JTAG Uart Intel FPGA IP
- Add Timer
  - Processors and Peripherals → Peripherals → Interval Timer Intel FPGA IP
- Add System ID
  - Basic Functions → Simulation; Debug and Verification → Debug and Performance → System ID Peripheral Intel FPGA IP



	reset1	Reset Input	Double-click to export	[clk1]
☑	⊟ jtag_uart_0	JTAG UART Intel FPGA IP		
	clk	Clock Input	Double-click to export	unconnected
	reset	Reset Input	Double-click to export	[clk]
	avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]
	irq	Interrupt Sender	Double-click to export	[clk]
☑	⊟ timer_0	Interval Timer Intel FPGA IP		
	clk	Clock Input	Double-click to export	unconnected
	reset	Reset Input	Double-click to export	[clk]
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]
	irq	Interrupt Sender	Double-click to export	[clk]
☑	⊟ sysid_qsys_0	System ID Peripheral Intel FPGA IP		
	clk	Clock Input	Double-click to export	unconnected
	reset	Reset Input	Double-click to export	[clk]
	control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]

Basic NIOS System

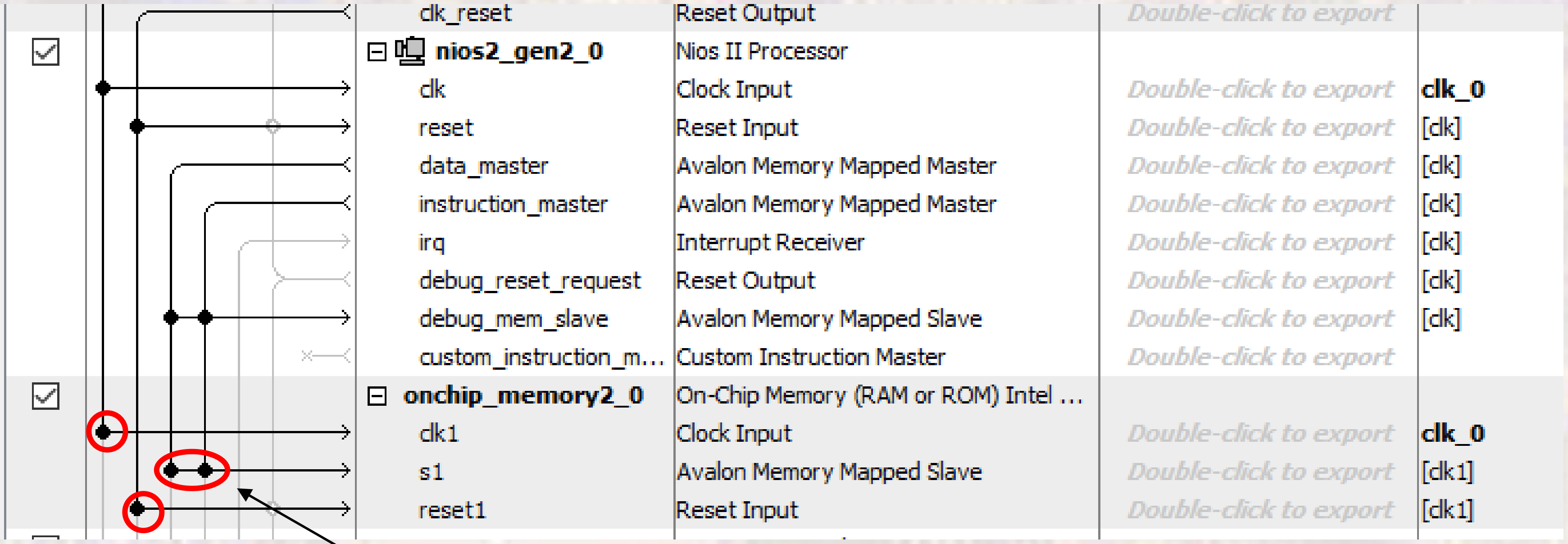
- Connect up basic NIOS system
  - NIOS Inputs



⊟ clk_0	Clock Source		
clk_in	Clock Input	**clk**	*exported*
clk_in_reset	Reset Input	**reset**	
clk	Clock Output	*Double-click to export*	clk_0
clk_reset	Reset Output	*Double-click to export*	
⊟ **nios2_gen2_0**	Nios II Processor		
clk	Clock Input	*Double-click to export*	**clk_0**
reset	Reset Input	*Double-click to export*	[clk]
data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]
instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]
irq	Interrupt Receiver	*Double-click to export*	[clk]
debug_reset_request	Reset Output	*Double-click to export*	[clk]
debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]
custom_instruction_m...	Custom Instruction Master	*Double-click to export*	

Basic NIOS System

- Connect up basic NIOS system
  - On-chip Memory

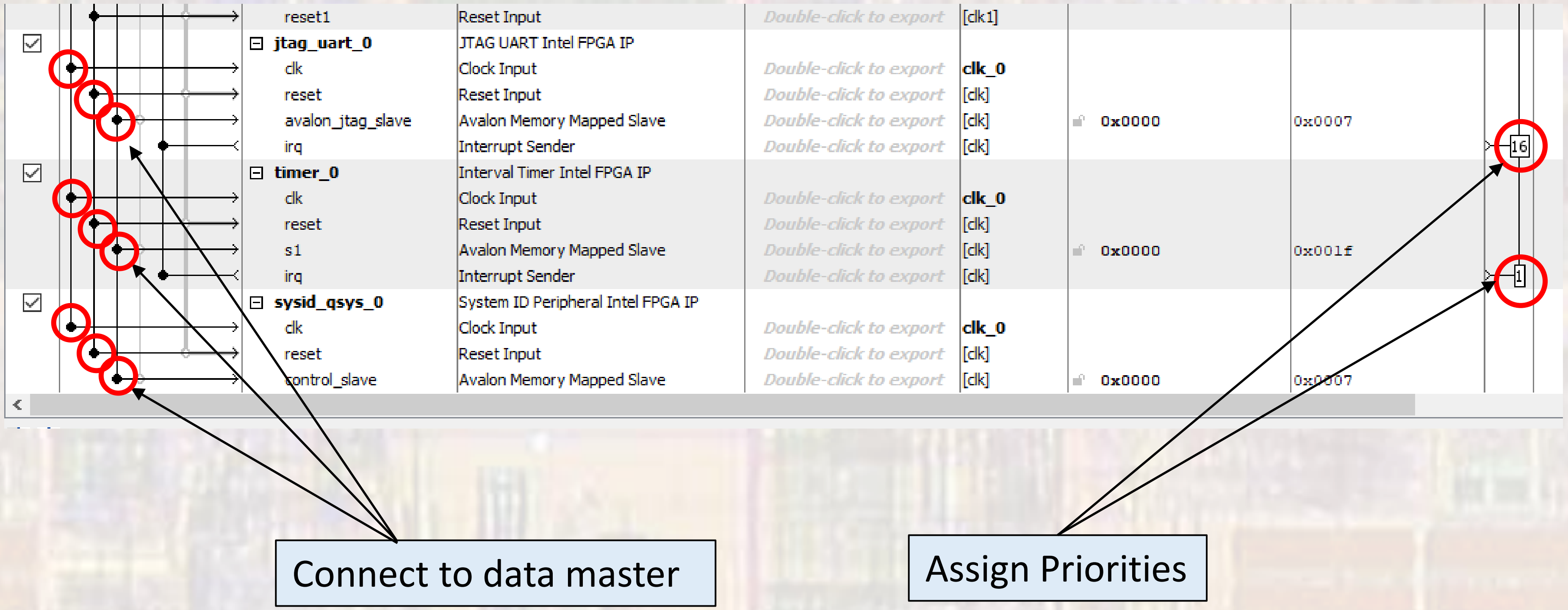


	Name	Description	Export	Clock
	clk_reset	Reset Output	Double-click to export	
✓	nios2_gen2_0	Nios II Processor		
	clk	Clock Input	Double-click to export	clk_0
	reset	Reset Input	Double-click to export	[clk]
	data_master	Avalon Memory Mapped Master	Double-click to export	[clk]
	instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]
	irq	Interrupt Receiver	Double-click to export	[clk]
	debug_reset_request	Reset Output	Double-click to export	[clk]
	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]
	custom_instruction_m...	Custom Instruction Master	Double-click to export	
✓	onchip_memory2_0	On-Chip Memory (RAM or ROM) Intel ...		
	clk1	Clock Input	Double-click to export	clk_0
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]
	reset1	Reset Input	Double-click to export	[clk1]

Connect to data and instruction masters

Basic NIOS System

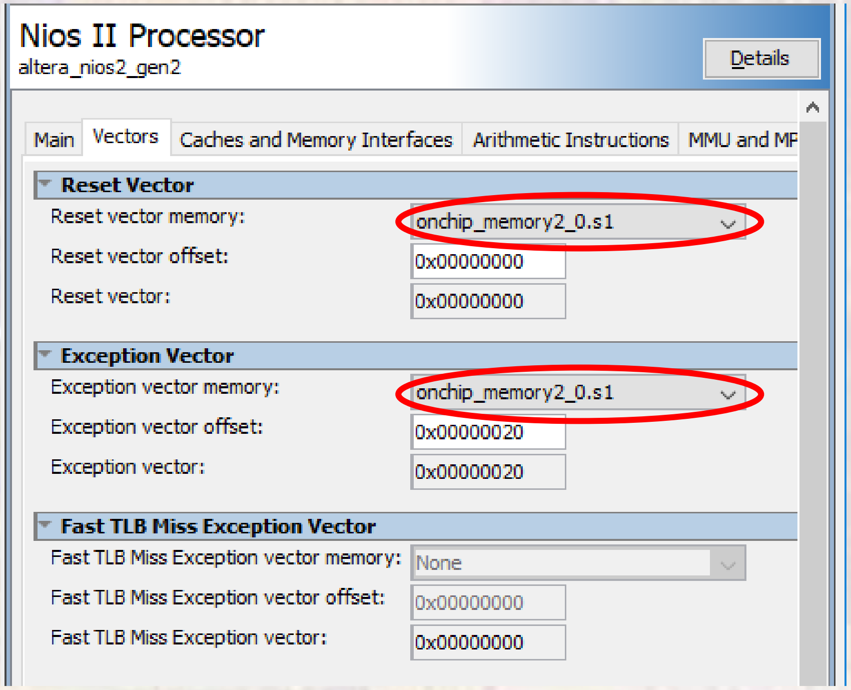
- Connect up basic NIOS system
  - JTAG, Timer, SysID



	reset1	Reset Input	*Double-click to export*	[clk1]			
✓	jtag_uart_0	JTAG UART Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	clk_0			
	reset	Reset Input	*Double-click to export*	[clk]			
	avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000	0x0007	
	irq	Interrupt Sender	*Double-click to export*	[clk]			16
✓	timer_0	Interval Timer Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	clk_0			
	reset	Reset Input	*Double-click to export*	[clk]			
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000	0x001f	
	irq	Interrupt Sender	*Double-click to export*	[clk]			1
✓	sysid_qsys_0	System ID Peripheral Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	clk_0			
	reset	Reset Input	*Double-click to export*	[clk]			
	control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000	0x0007	

Basic NIOS System

- Connect up basic NIOS system
  - Assign the NIOS II Reset and Exception vectors
    - Open the NIOS Processor
    - Select Vectors
    - Select on-chip memory for Reset and Exception



Reset Vector

Reset vector memory:	onchip_memory2_0.s1
Reset vector offset:	0x00000000
Reset vector:	0x00000000

Exception Vector

Exception vector memory:	onchip_memory2_0.s1
Exception vector offset:	0x00000020
Exception vector:	0x00000020

Fast TLB Miss Exception Vector

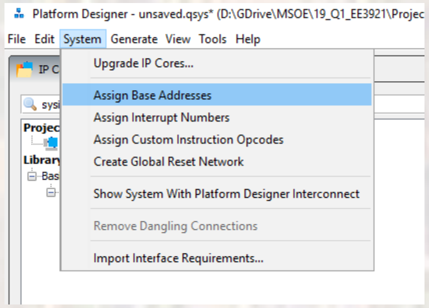
Fast TLB Miss Exception vector memory:	None
Fast TLB Miss Exception vector offset:	0x00000000
Fast TLB Miss Exception vector:	0x00000000

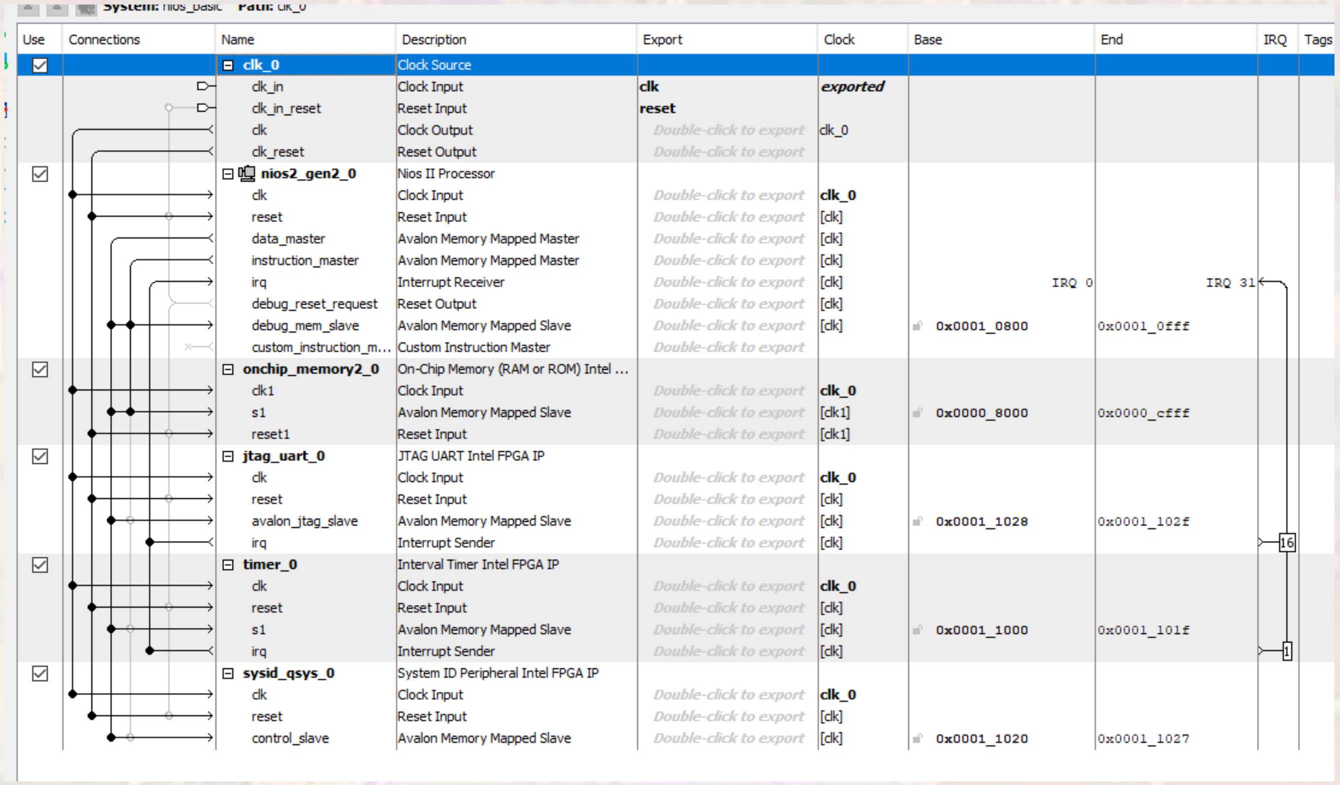
Basic NIOS System

- Complete Basic System
  - Assign base addresses

  - System → Assign Base Addresses

Assign Base Addresses

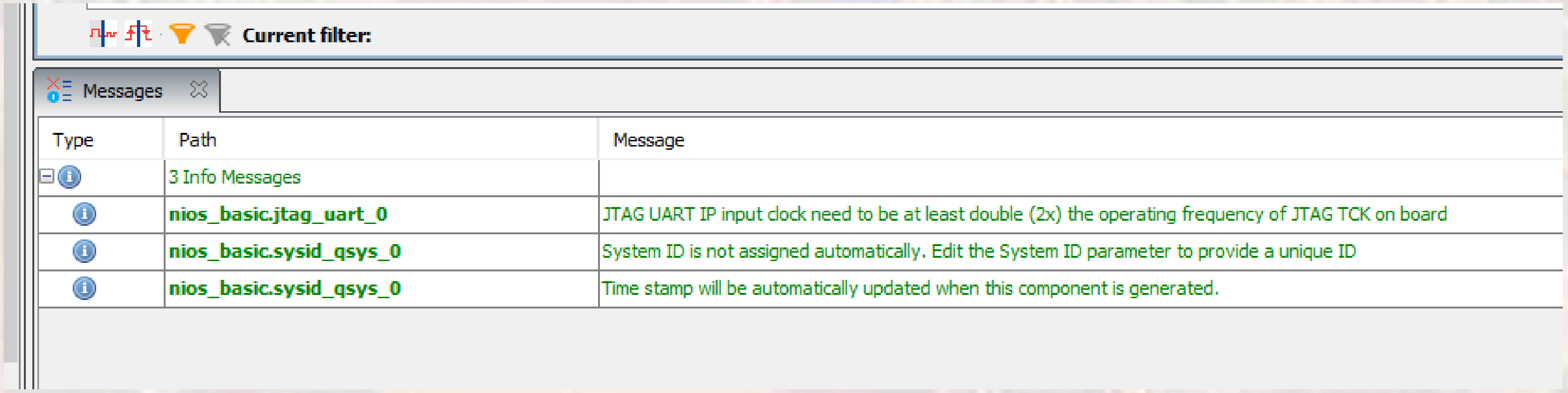

Platform Designer - unsaved.qsys* (D:\GDrive\MSOE\19_Q1_EE3921\Projec
File Edit System Generate View Tools Help
Upgrade IP Cores...
Assign Base Addresses
Assign Interrupt Numbers
Assign Custom Instruction Opcodes
Create Global Reset Network
Show System With Platform Designer Interconnect
Remove Dangling Connections
Import Interface Requirements...


System: nios_basic Path: clk_0

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags
☑		⊟ clk_0	Clock Source						
		clk_in	Clock Input	clk	exported				
		clk_in_reset	Reset Input	reset					
		clk	Clock Output	Double-click to export	clk_0				
		clk_reset	Reset Output	Double-click to export					
☑		⊟ nios2_gen2_0	Nios II Processor						
		clk	Clock Input	Double-click to export	clk_0				
		reset	Reset Input	Double-click to export	[clk]				
		data_master	Avalon Memory Mapped Master	Double-click to export	[clk]				
		instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]				
		irq	Interrupt Receiver	Double-click to export	[clk]	IRQ 0	IRQ 31		
		debug_reset_request	Reset Output	Double-click to export	[clk]				
		debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_0800	0x0001_0fff		
		custom_instruction_m...	Custom Instruction Master	Double-click to export					
☑		⊟ onchip_memory2_0	On-Chip Memory (RAM or ROM) Intel ...						
		clk1	Clock Input	Double-click to export	clk_0				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]	0x0000_8000	0x0000_cfff		
		reset1	Reset Input	Double-click to export	[clk1]				
☑		⊟ jtag_uart_0	JTAG UART Intel FPGA IP						
		clk	Clock Input	Double-click to export	clk_0				
		reset	Reset Input	Double-click to export	[clk]				
		avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1028	0x0001_102f		
		irq	Interrupt Sender	Double-click to export	[clk]			16	
☑		⊟ timer_0	Interval Timer Intel FPGA IP						
		clk	Clock Input	Double-click to export	clk_0				
		reset	Reset Input	Double-click to export	[clk]				
		s1	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1000	0x0001_101f		
		irq	Interrupt Sender	Double-click to export	[clk]			1	
☑		⊟ sysid_qsys_0	System ID Peripheral Intel FPGA IP						
		clk	Clock Input	Double-click to export	clk_0				
		reset	Reset Input	Double-click to export	[clk]				
		control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0001_1020	0x0001_1027		

Basic NIOS System

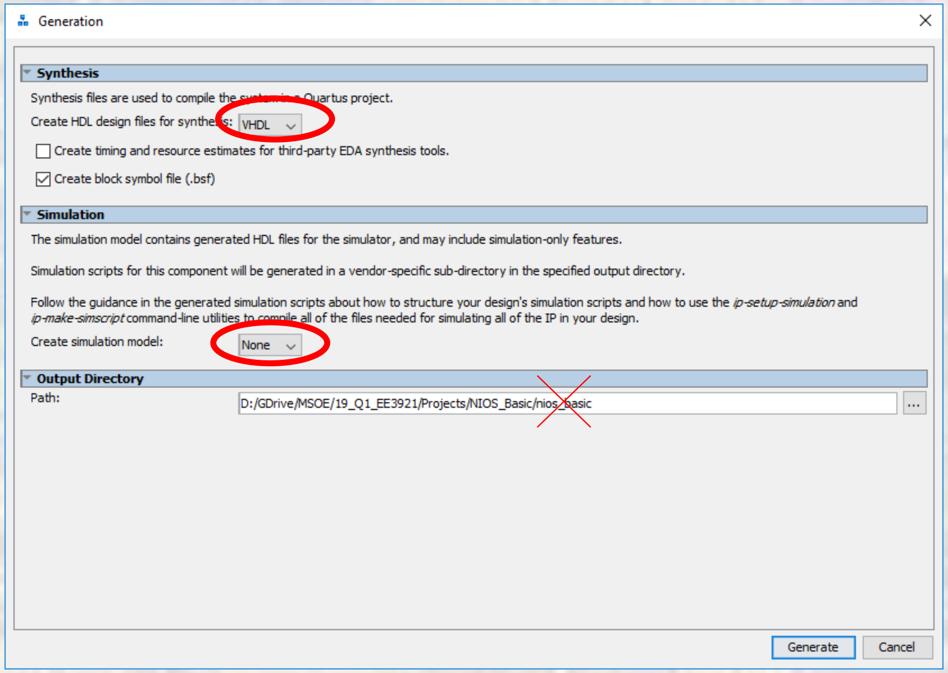
- Create Basic System
  - Check for errors



Type	Path	Message
	3 Info Messages	
	nios_basic.jtag_uart_0	JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board
	nios_basic.sysid_qsys_0	System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID
	nios_basic.sysid_qsys_0	Time stamp will be automatically updated when this component is generated.

Basic NIOS System

- 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 '…'



Generation

Synthesis

Synthesis files are used to compile the system in a Quartus project.

Create HDL design files for synthesis: VHDL

☐ Create timing and resource estimates for third-party EDA synthesis tools.

☑ Create block symbol file (.bsf)

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

Output Directory

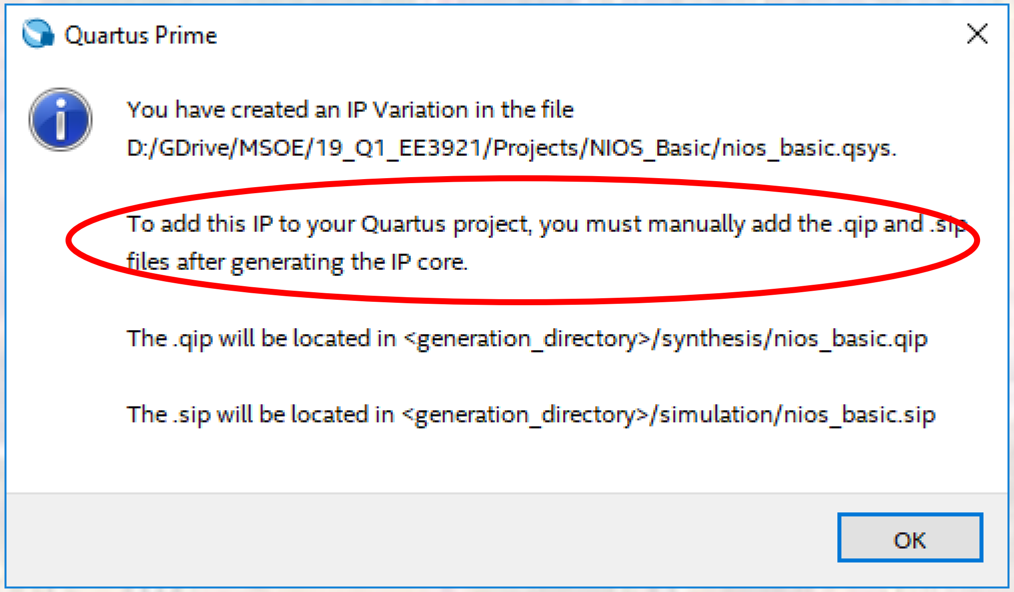
Path: D:/GDrive/MSOE/19_Q1_EE3921/Projects/NIOS_Basic/~~nios_basic~~

Should point to your project directory

→ D:/GDrive/MSOE/19_Q1_EE3921/Projects/NIOS_Basic

Basic NIOS System

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



Quartus Prime

You have created an IP Variation in the file D:/GDrive/MSOE/19_Q1_EE3921/Projects/NIOS_Basic/nios_basic.qsys.

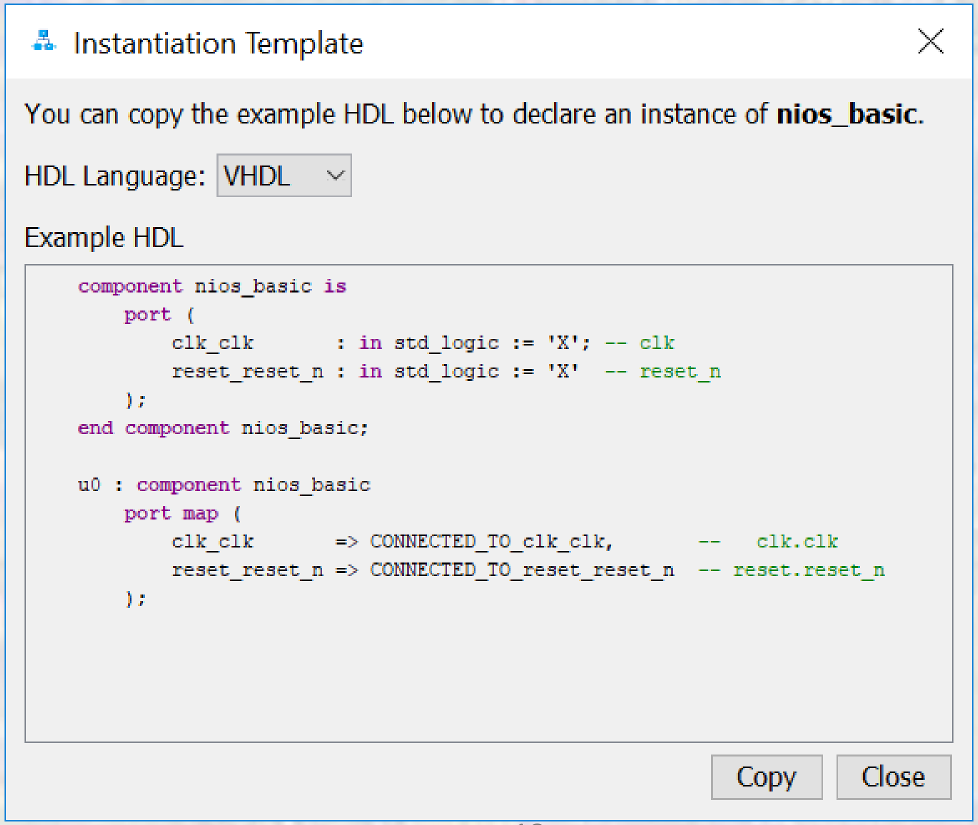
To add this IP to your Quartus project, you must manually add the .qip and .sip files after generating the IP core.

The .qip will be located in <generation_directory>/synthesis/nios_basic.qip

The .sip will be located in <generation_directory>/simulation/nios_basic.sip

Basic NIOS System

- Create DE10 Design
  - Instantiate into a VHDL file
    - Open a new VHDL design (nios_basic_de10.vhdl)
    - In Platform Designer: Generate → Show Instantiation Template



You can copy the example HDL below to declare an instance of **nios_basic**.

HDL Language: VHDL

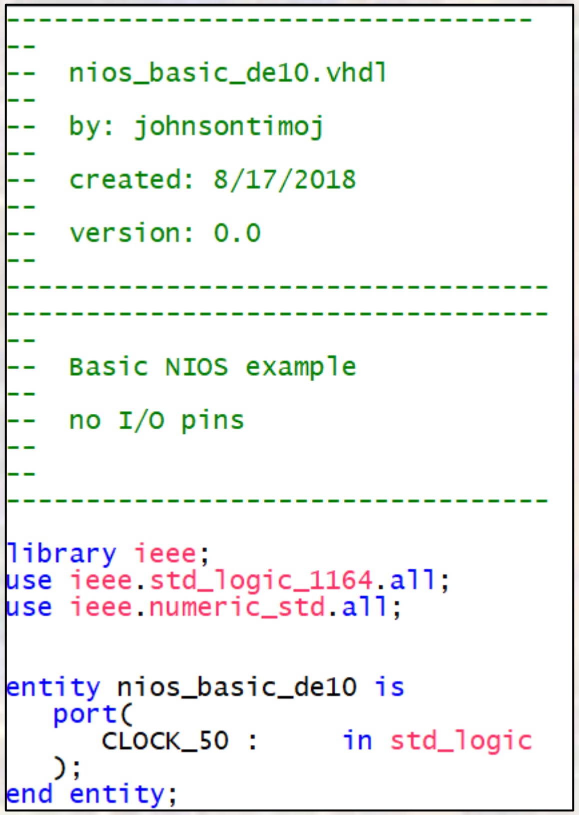
Example HDL

```
    component nios_basic is
        port (
            clk_clk       : in std_logic := 'X'; -- clk
            reset_reset_n : in std_logic := 'X'  -- reset_n
        );
    end component nios_basic;

    u0 : component nios_basic
        port map (
            clk_clk       => CONNECTED_TO_clk_clk,       --   clk.clk
            reset_reset_n => CONNECTED_TO_reset_reset_n  -- reset.reset_n
        );
```

Basic NIOS System

- Create DE10 Design
  - Instantiate into a VHDL file

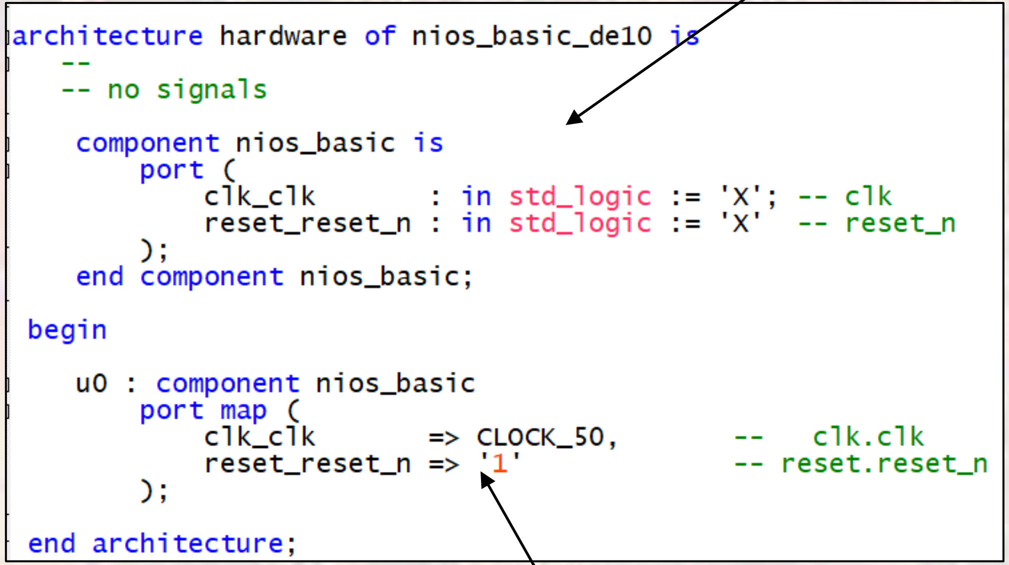


```
-----------------------------------
--
--   nios_basic_de10.vhdl
--
--   by: johnsontimoj
--
--   created: 8/17/2018
--
--   version: 0.0
--
-----------------------------------
-----------------------------------
--
--   Basic NIOS example
--
--   no I/O pins
--
--
-----------------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


entity nios_basic_de10 is
   port(
      CLOCK_50 :      in std_logic
   );
end entity;
```

Instantiation template component



```
architecture hardware of nios_basic_de10 is
   --
   -- no signals

    component nios_basic is
        port (
            clk_clk         : in std_logic := 'X'; -- clk
            reset_reset_n : in std_logic := 'X'   -- reset_n
        );
    end component nios_basic;

 begin

    u0 : component nios_basic
        port map (
            clk_clk         => CLOCK_50,        --   clk.clk
            reset_reset_n => '1'                -- reset.reset_n
        );

 end architecture;
```

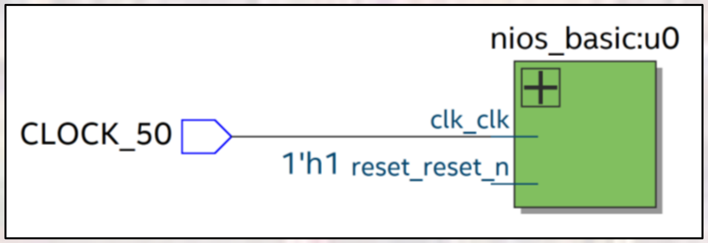
Instantiation template instance mapped to DE10 qsf pin aliases

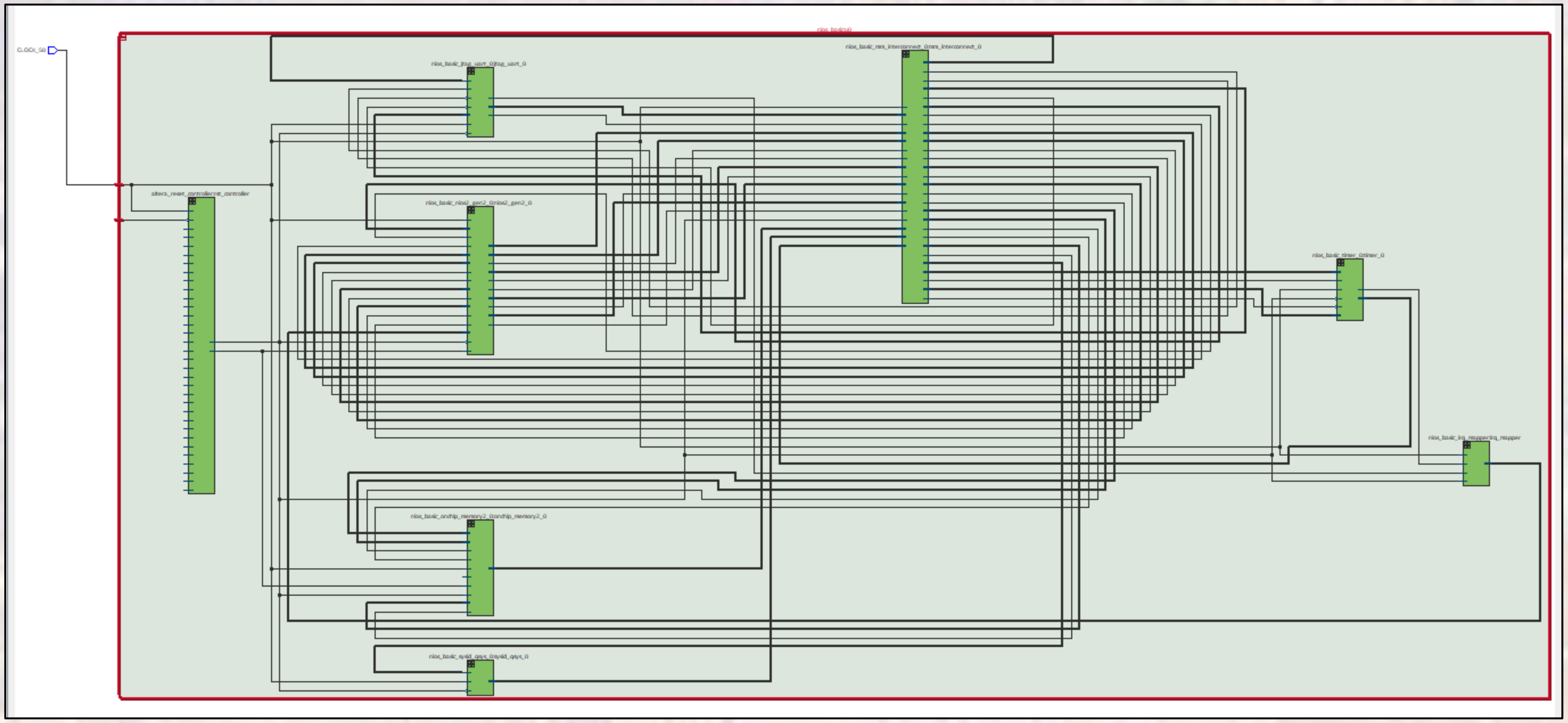
Basic NIOS System

- 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 SDC file for timing analysis
  - Be sure to set your top level entity
  - Start Compilation

Basic NIOS System

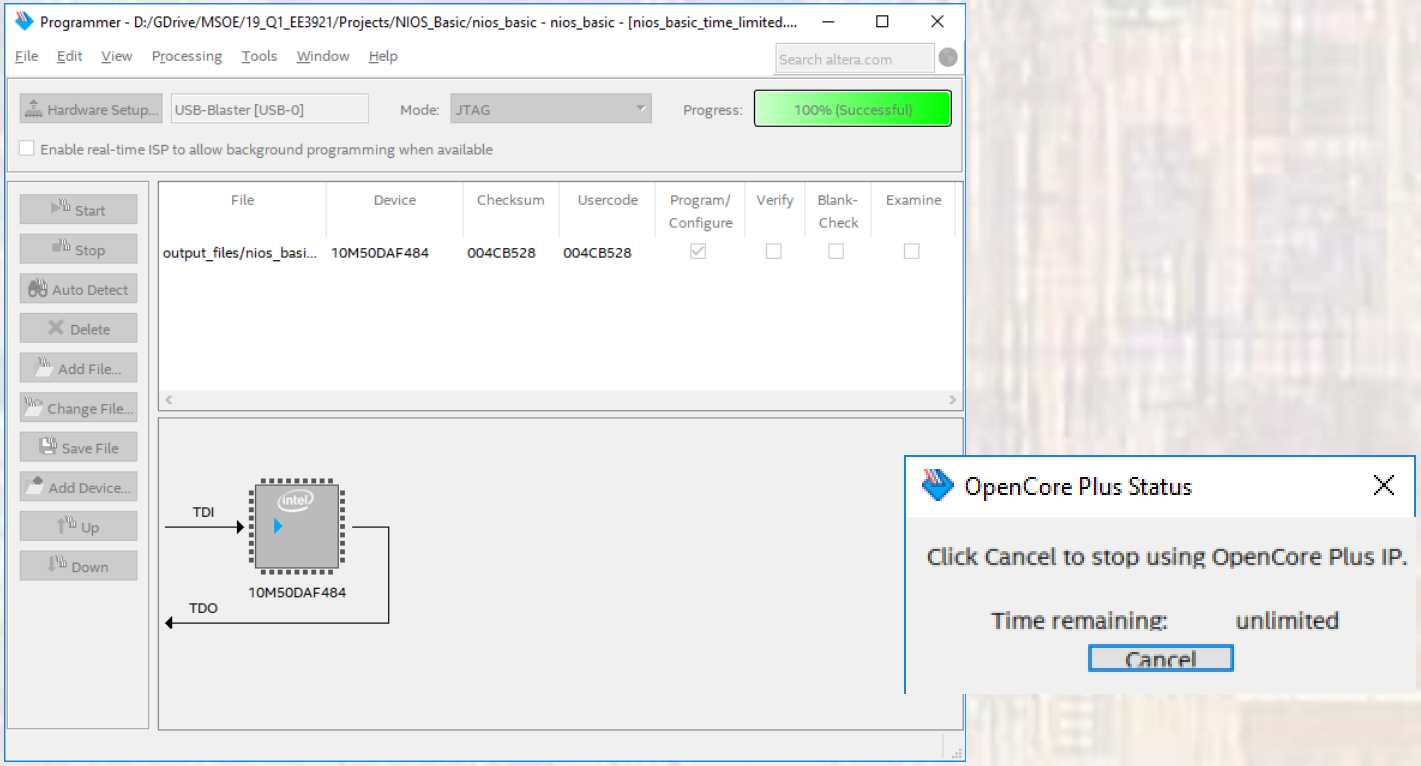
- Create DE10 Design





Basic NIOS System

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


Programmer - D:/GDrive/MSOE/19_Q1_EE3921/Projects/NIOS_Basic/nios_basic - nios_basic - [nios_basic_time_limited....

Hardware Setup... USB-Blaster [USB-0] Mode: JTAG Progress: 100% (Successful)

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine
output_files/nios_basi...	10M50DAF484	004CB528	004CB528	✓			

OpenCore Plus Status

Click Cancel to stop using OpenCore Plus IP.

Time remaining: unlimited

Basic NIOS System

- Basic NIOS System

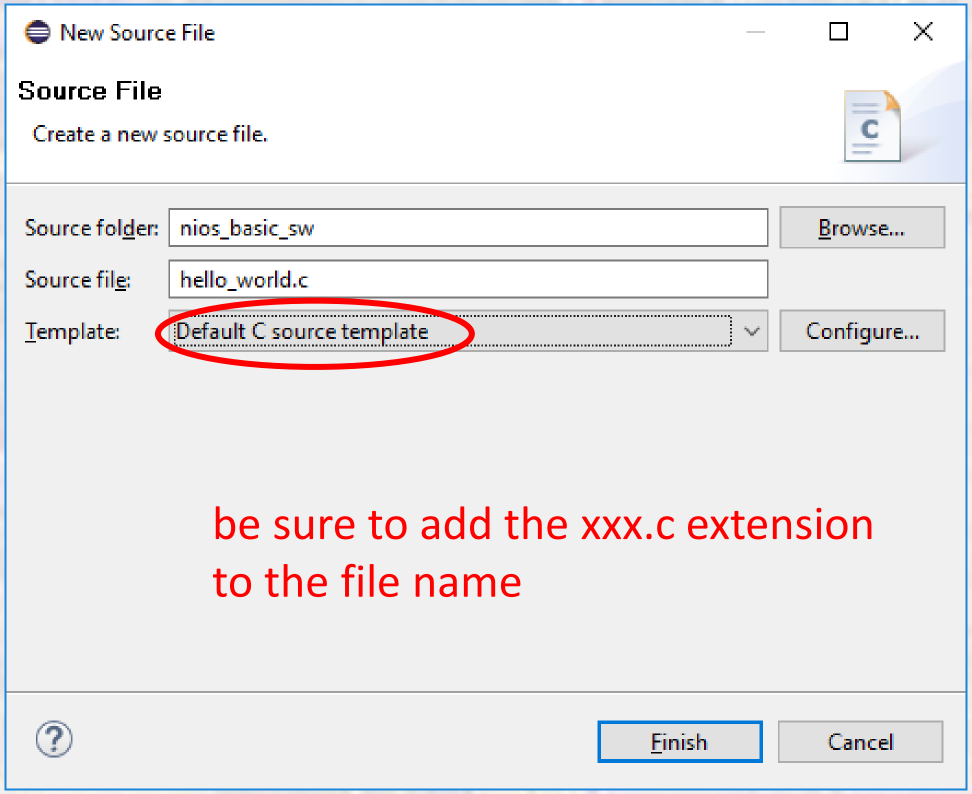
SOFTWARE

Basic NIOS System

- 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)

Basic NIOS System

- Create Eclipse System
  - Create program
    - Right click on the project directory and choose New → c source file



New Source File

Source File

Create a new source file.

Source folder: nios_basic_sw

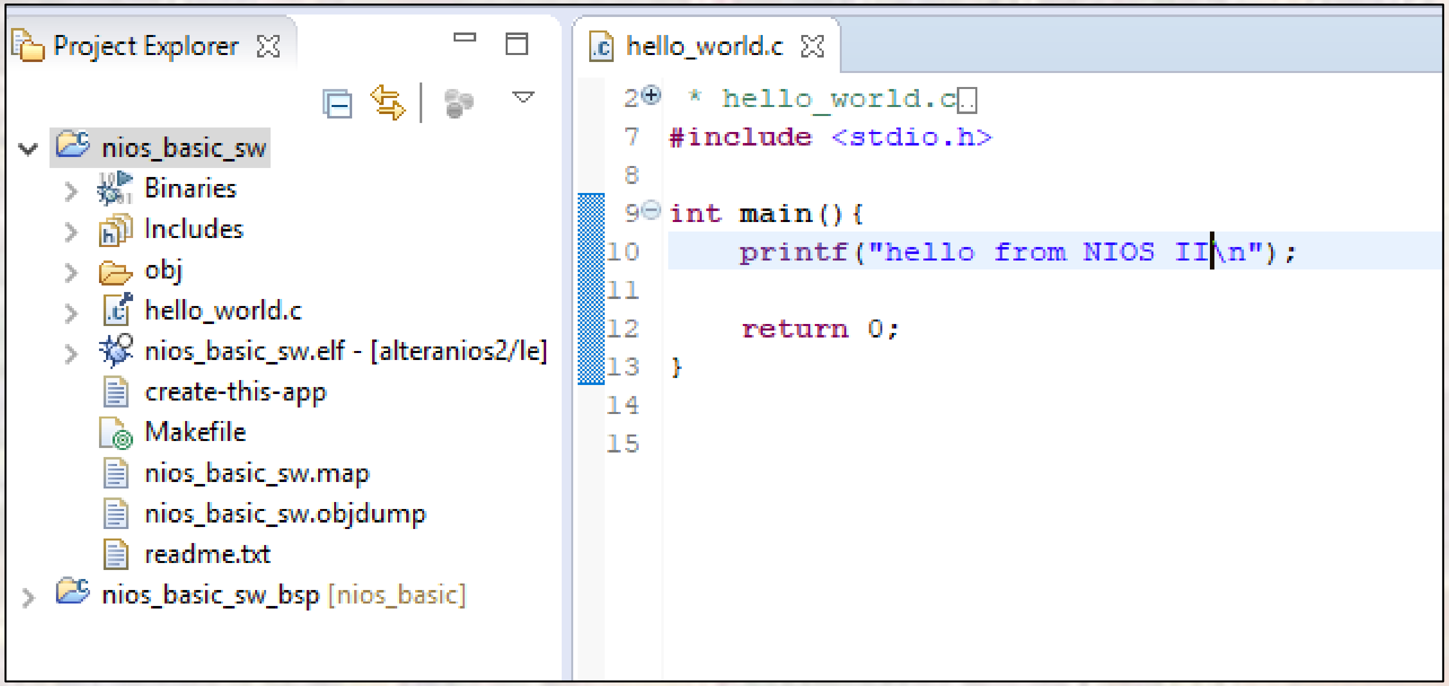
Source file: hello_world.c

Template: Default C source template

be sure to add the xxx.c extension to the file name

Basic NIOS System

- Create Eclipse System
  - Create program
    - Type in the program



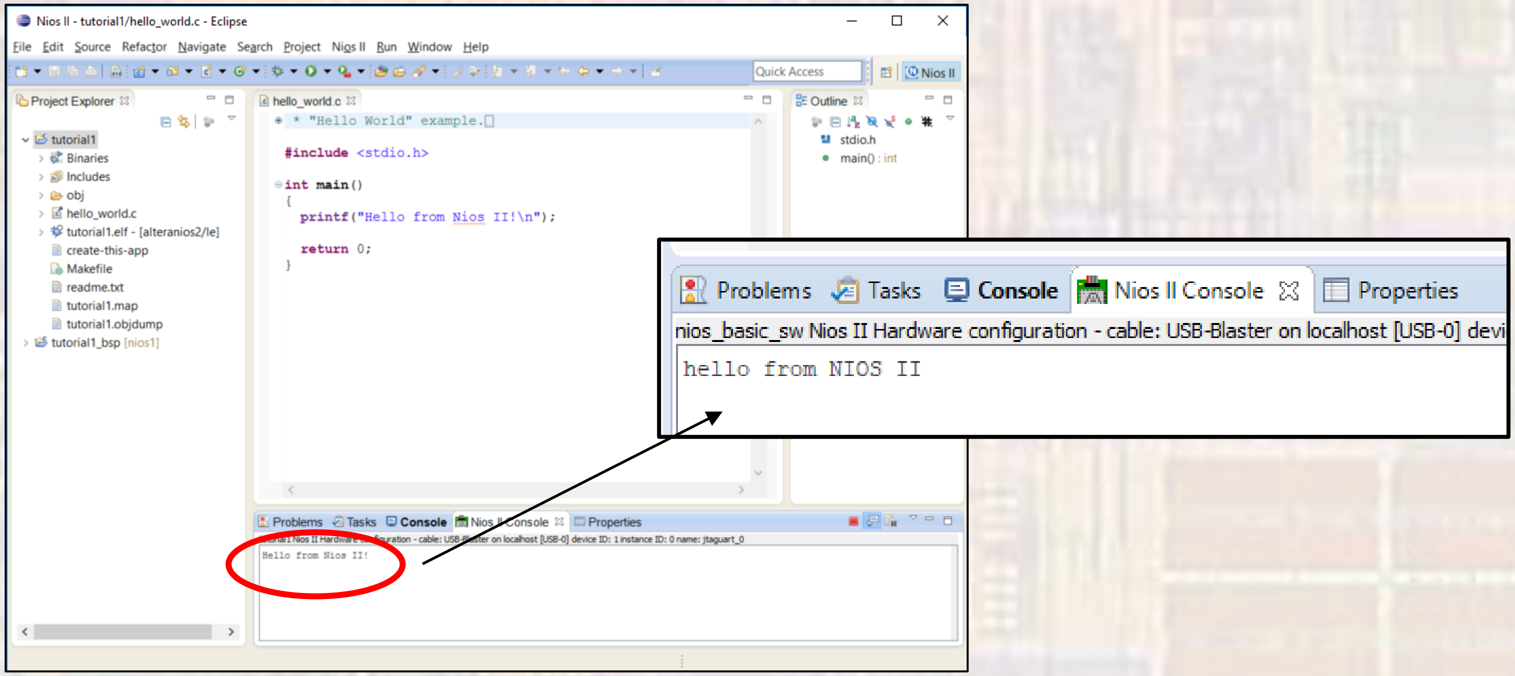
```
 * hello_world.c
#include <stdio.h>

int main(){
    printf("hello from NIOS II\n");

    return 0;
}
```

Basic NIOS System

- Create Eclipse System
  - Compile and run the software
    - Select the code file (hello_world.c)
    - Project → Build Project
    - Right Click on the project → run as → Nios II Hardware



```
 * "Hello World" example.

#include <stdio.h>

int main()
{
  printf("Hello from Nios II!\n");

  return 0;
}
```

```
tutorial1 Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
Hello from Nios II!
```

```
nios_basic_sw Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] devi
hello from NIOS II
```