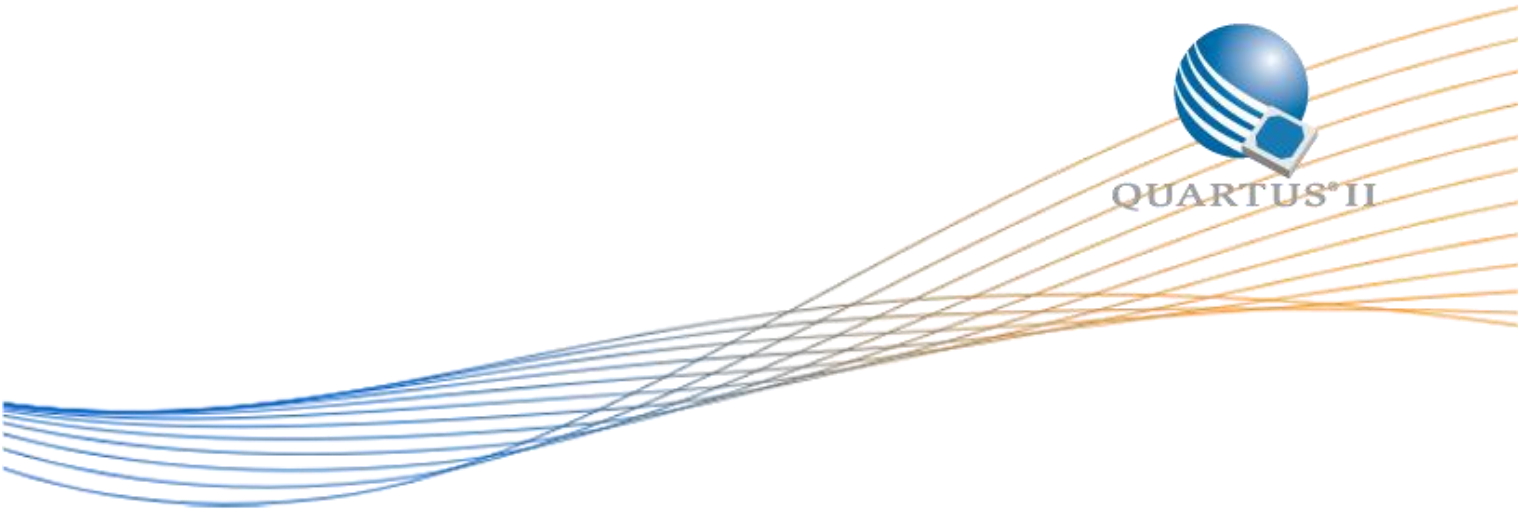


NIOS II Custom Instruction for MAX 10 FPGA Development Kit

This document shows the Custom instruction for NIOS II design example for the MAX 10 FPGA Development Kit.

About Custom Instructions in NIOS II:

When a design includes an Altera Nios II embedded processor, the design can accelerate time-critical software algorithms by adding custom instructions to the Nios II processor instruction set. Custom instructions allow the user to reduce a complex sequence of standard instructions to a single instruction implemented in hardware. This feature can be used for a variety of applications, for example, to optimize Software inner loops for digital signal processing (DSP), packet header processing, and computation-intensive applications. In Qsys, each custom instruction is a separate component in the Qsys system. The user can add as many as 256 custom instructions to your system.
Refer to Nios II Custom Instruction User Guide for further information.

About this Design Example

This design example shows how to implement the cyclic redundancy check (CRC) algorithm as a Nios II custom instruction. The CRC algorithm detects the corruption of data during transmission. The CRC calculation consists of an iterative algorithm involving XOR and shift operations. These operations are carried out concurrently in hardware and iteratively in software. Since the operations are carried out concurrently, the execution is much faster in hardware. This example demonstrates the way to implement an extended multi-cycle Nios II custom instruction.

This design Example performs the following:
1) Computation of CRC using the custom instruction
2) Computation of CRC using a software C code.
3) Computation of CRC using a optimized software C code.

Steps to Run the Program

The steps to run the Custom Instruction for the MAX 10 Dev Kit are:

1. Extract all the files from the custom_inst.par by following the instructions on the design store. The extracted files have the following the file structure:
   i) Crc_hw folder consists of the 2 Verilog files CRC_Component.v and CRC_Custom_Instruction.v for the custom instruction.
   ii) The master_image folder consists of the custom_inst.sof and custom_inst.pof file which can be directly used by the user to program the board. It also contains the .hex file which can be used to directly run the program from the nios terminal.
   iii) The software/src folder contains all the .c files
   crc_main.c : is the main file which calls all the other files.
   crc.c: contains the software implementation and the optimized software implementation of CRC function.
   ci_crc.c: is the .c file for executing the built in function for the custom instruction.
   ci_crc.h: hex file related to custom instruction

crc.h: hex file related to the software implementation.

iv) It also contains platform/nios_setup which contains files related to the qsys setup.

v) Also the top level file custom_inst.v, the sdc file custom_inst.sdc is included

2. Use the command "quartus top" to launch quartus and open the project.
3. After quartus launches:
   i) Go to Tools→Programmer.
   ii) Click on Hardware Setup. A hardware Setup dialog box will open.
   iii) Select USB Blaster under currently selected hardware. Click on close.
   iv) Use the Add file tab to navigate to the master/images folder and select the custom_inst.sof file.
   v) Select the checkbox Program/Configure and Verify.
   vi) Click on start. This starts the downloading of the .pof file on to the Eval Kit.
4. We are using the Nios terminal to display the results. To open the command shell go to: Start→All Programs →Altera→Nios II EDS → Nios II Command Shell in Windows or <Nios II EDS install path>/nios2_command_shell.sh in Unix.
5. Here, make sure that the reset pin on the Dev Kit is in the "off" position. The reset pin is assigned to 1st switch from the SW1 bank of switches. The ON state is written (in very tiny letters on the DIP switch bank) .
6. In the Nios II command shell type the command: nios2-terminal.
   The results of the CRC operation will be displayed.

Steps to Recreate the output files:

If you want to edit the project, follow the below procedure:

Hardware

1) Launch Quartus II and open the top.qpf project
File -> Open Project -> top.qpf
2) Compile the project (Processing -> Start Compilation).
3) Program the ./output_files/top.sof to the board using Quartus II Programmer (Tools -> Programmer - > Add File -> top.sof -> OK -> Start).

Software

1) Open Tools->Nios II Software build tools from Quartus II. This launches the NIOS II Eclipse IDE where you can modify your C Code.

2) Select the workspace for Nios II Eclipse. Then select File->New->Nios II application and BSP from Template

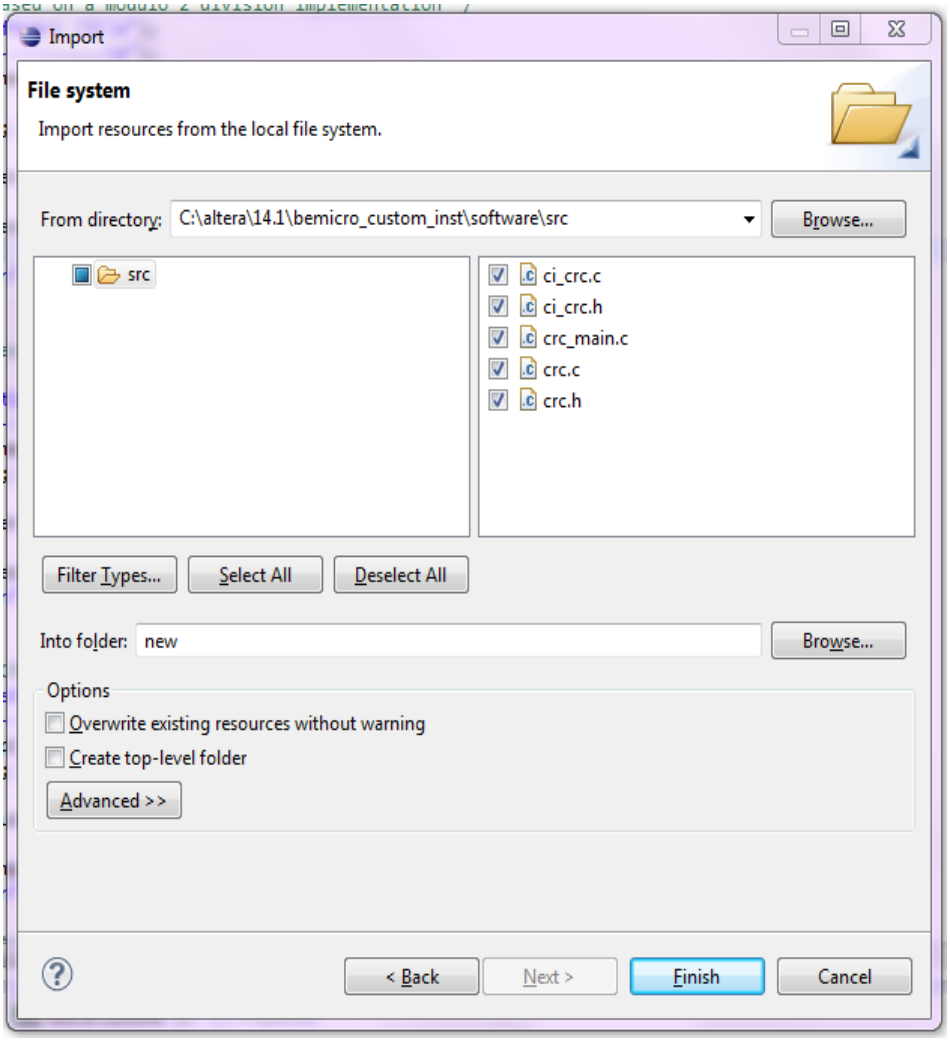
3) In the Window which opens, select the nios_setup.sopcinfo file in your Quartus project folder . The .sopcinfo file contains information about the Qsys system, each module instantiated in the project, and parameter names and values contained in the project.

4) Give the name to your nios project as cust_inst, select hello world small as the project template and click finish. You can see that cust_inst and cust_inst_bsp is created on your workspace.

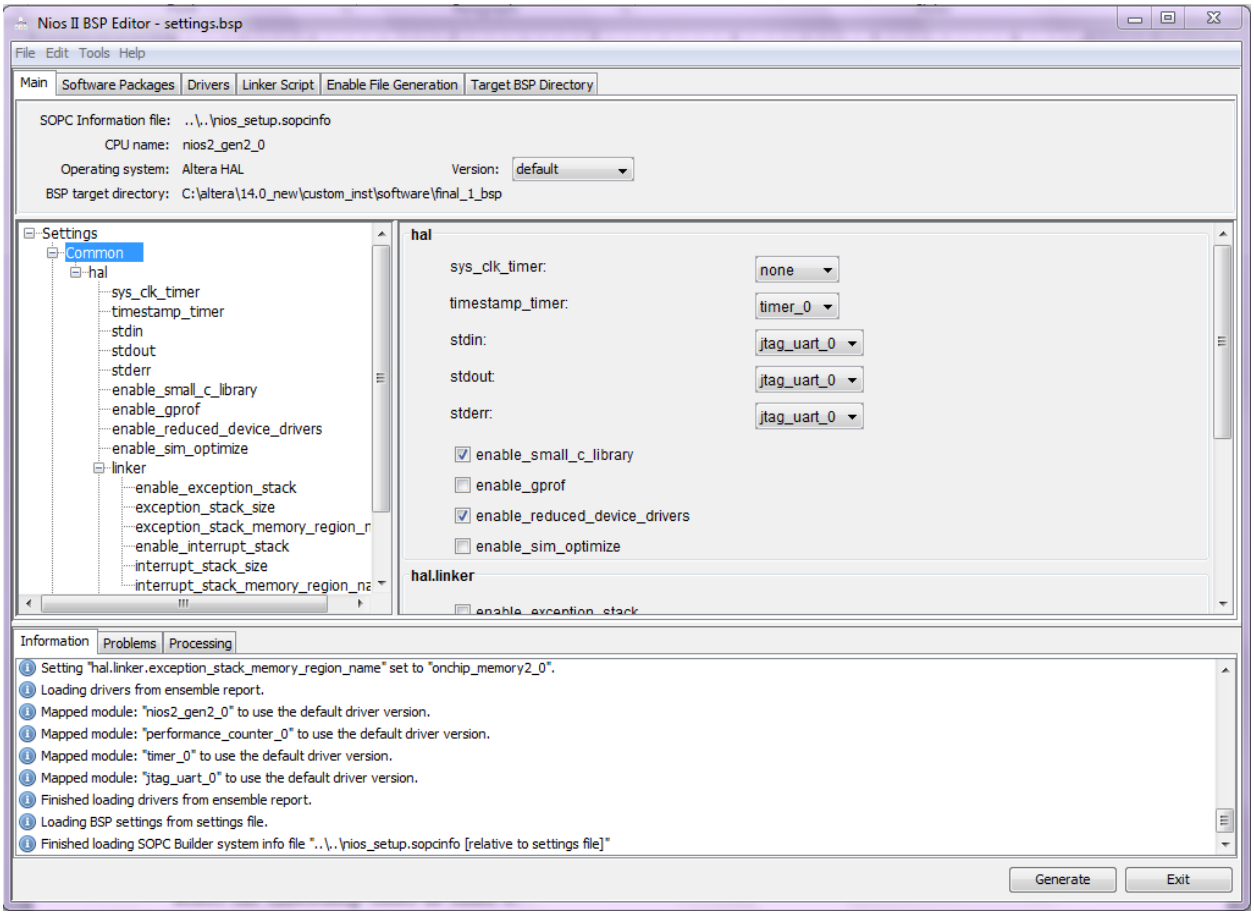
5) Now, we have to replace the contents of hello_world.c source with our source code. First Remove the helloworld.c file from the project. To do this right click on 'hello_world_small.c' and click delete. (screenshot attached below).

Next the 5 files in the software/src folder.
This can be done in 2 ways:

1) Directly Copy and Paste the required Files: Directly copy (Ctrl C) the files from the software/src folder. In the Eclipse tool right click on the custom_inst project and click Paste.
2) Import the required Files: Right Click on the custom_inst project. Click Import. A new Import popup will open.
    a. In this double click on General and the File System.
    b. Select Browse and then browse to the directory software/src and select ok. All the contents of this folder, the 5 .c files are shown in the right. Check the checkboxes for all the 5 files. Click on finish.
    Now all the files are present in the folder.



6)Since the Qsys contains the timer, this needs to be added here as the timestamp timer. For this right click on the custom_inst_bsp and select NIOS II → BSP Editor. A BSP Editor window will open. Here select the timestamp timer as timer 0 and click on generate.



7) Right Click on the custom_inst and Select **Build Project** to build the project. The initial build may take some time.

8) Once the build is finished, to run the project, right click on the project and select Run As -> Run Configurations.

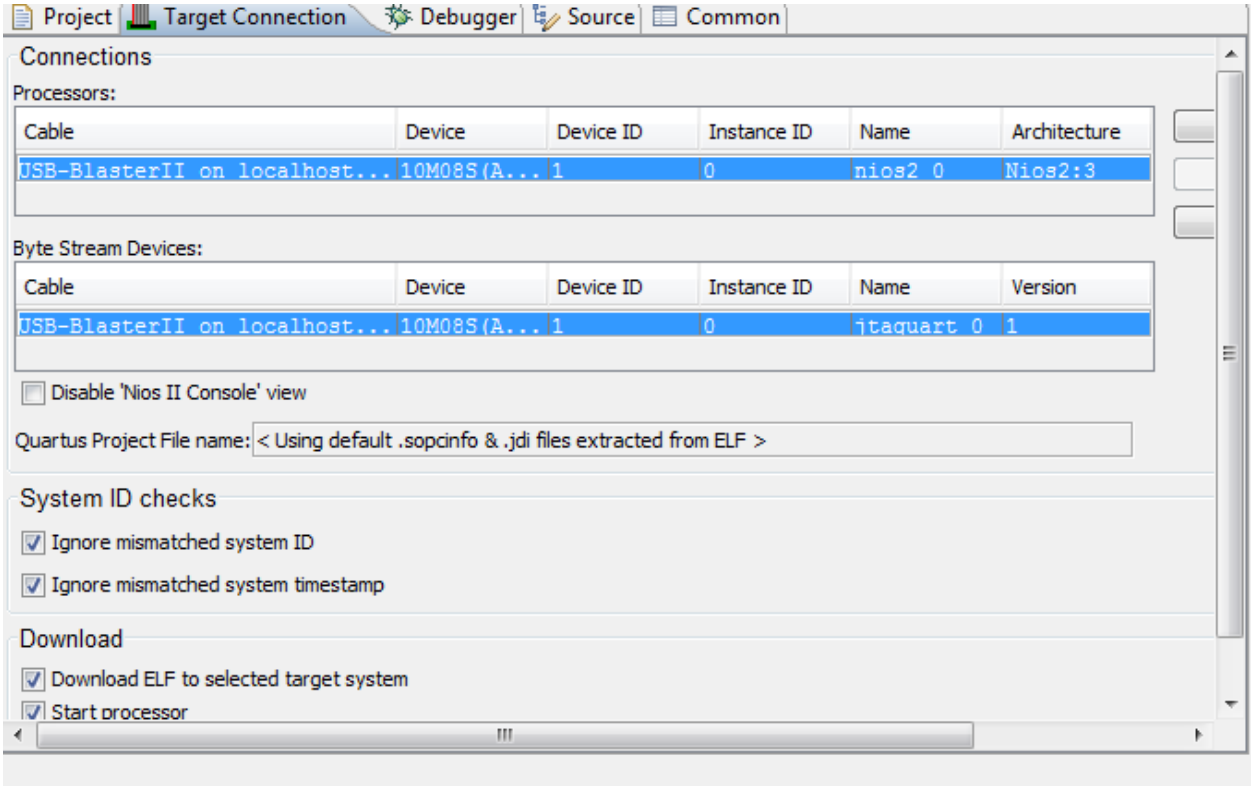
9) Double click on Nios II Hardware, and new configuration opens on the right pane . Make sure you select the project name as custom_inst and .elf file as custom_inst.elf .

10) Select Target Connection Tab . Then Check the following two check boxes

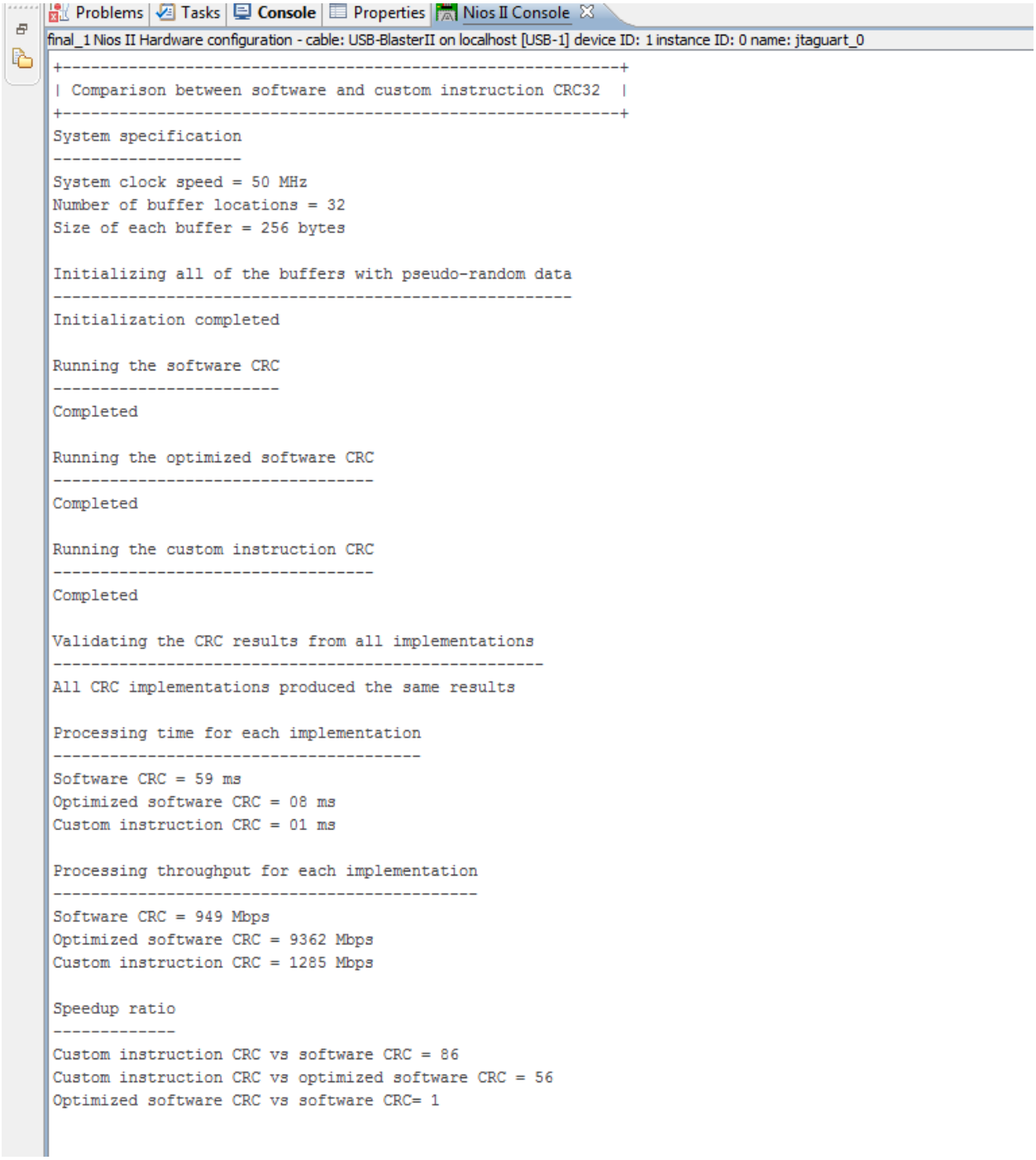
Ignore mismatched system ID

Ignore mismatched system timestamp

Next Click Apply and Run.



11) The following output is observed:



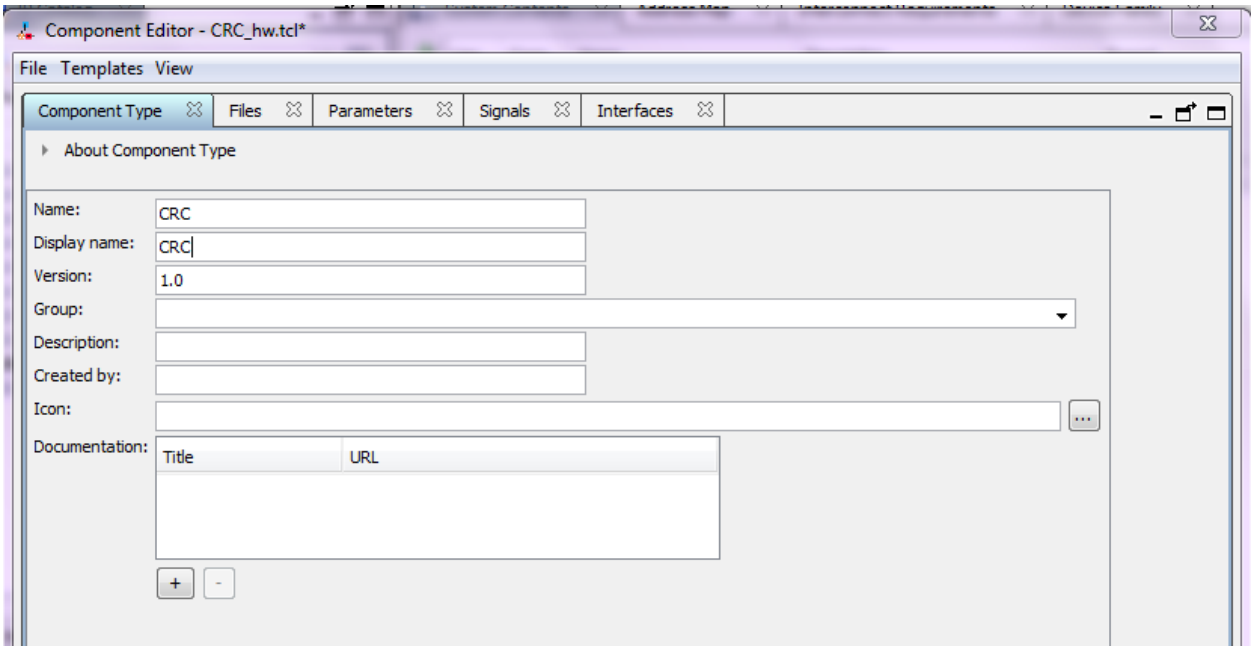
The user can modify the number of buffer locations and size of the buffers and run the program to observe the results.

NOTE:

The current project extracted from the .qar already contains the custom component added and working.

However, the steps to add this custom component and generate the Qsys system are shown below for user's reference.

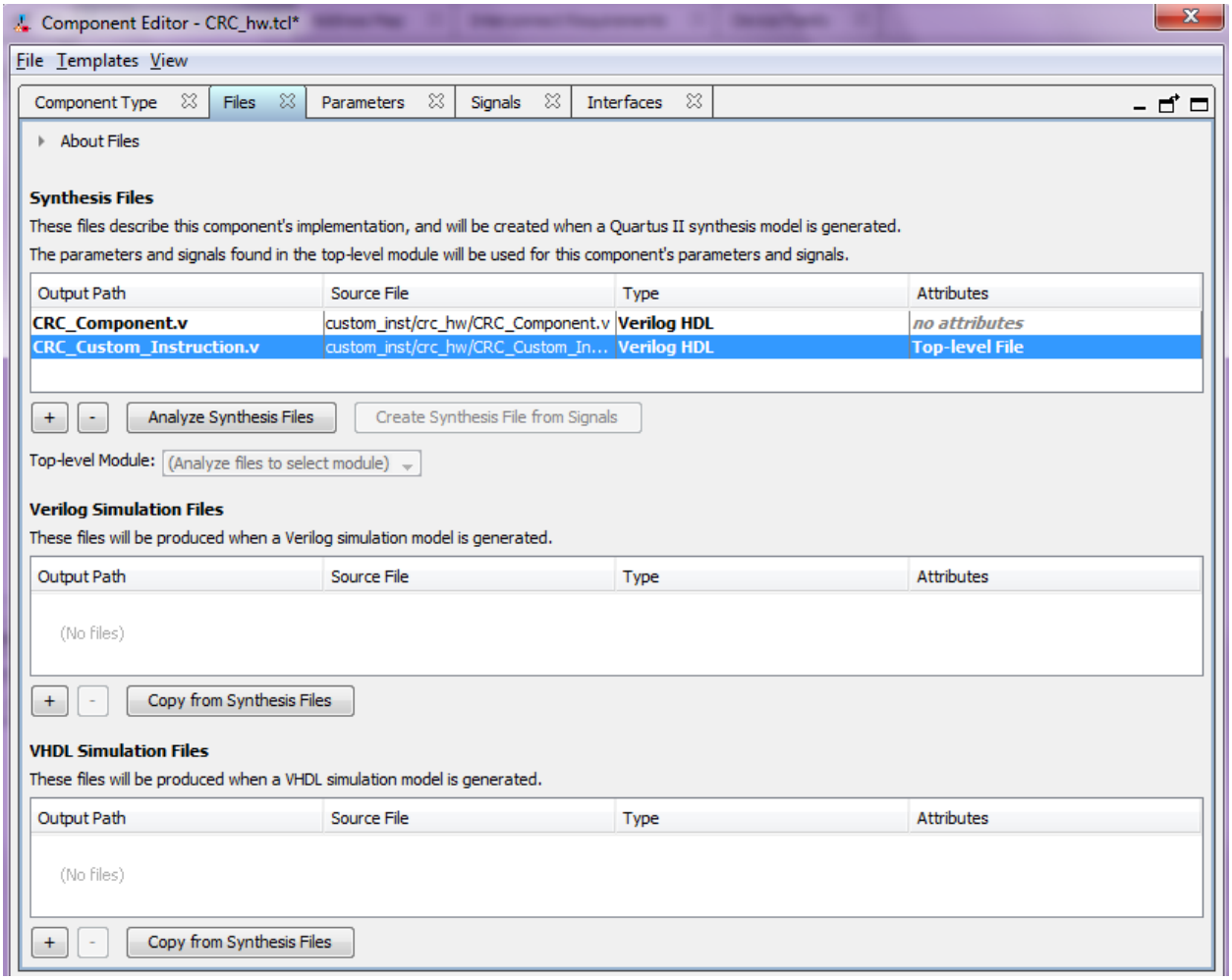
1) Open Qsys from Quartus. From the IP catalogue New component is selected. A new component window will open. The new component's name and display name is entered.



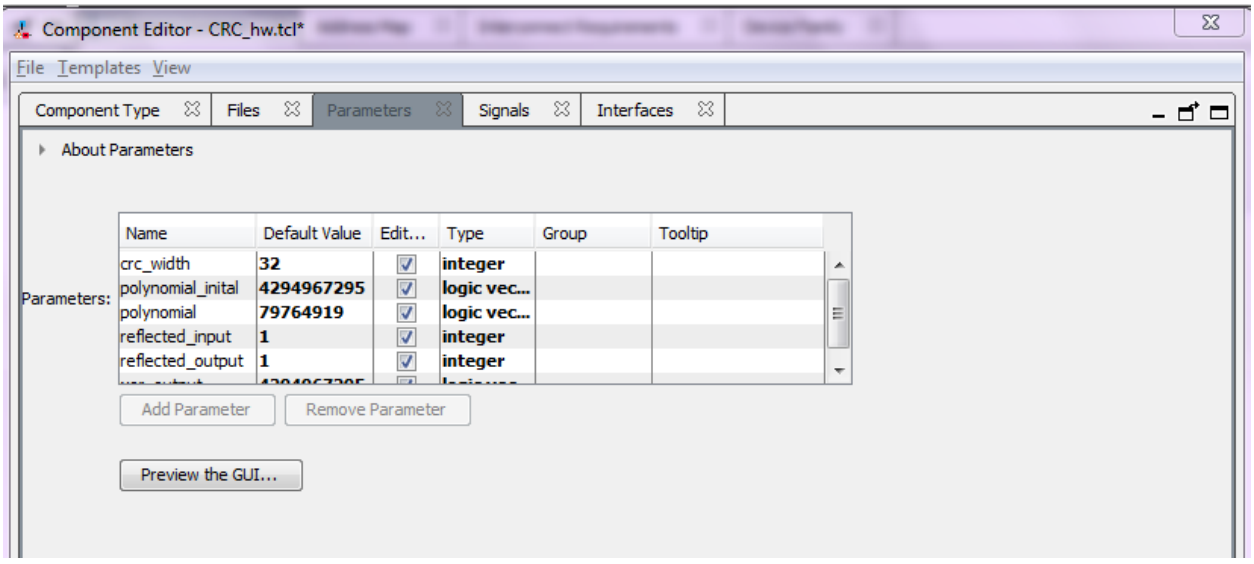
2) Click on Next. The Files Tab is displayed. The folder crc_hw contains the Verilog files, CRC_Component.v and CRC_Custom_Instruction.v for the custom instruction. These files need to be added here.

Next, we have to set the top level entity. For this, double click on the attributes section of the CRC_Custom_Instruction.v file and check the top level file option. This sets the CRC_Custom_Instruction.v file as the top level entity.

Next click on the analyze and synthesize files. This will report compilation errors if any in the .v files.



3) Click on the Next tab. Here the parameters present in the .v files will be displayed.

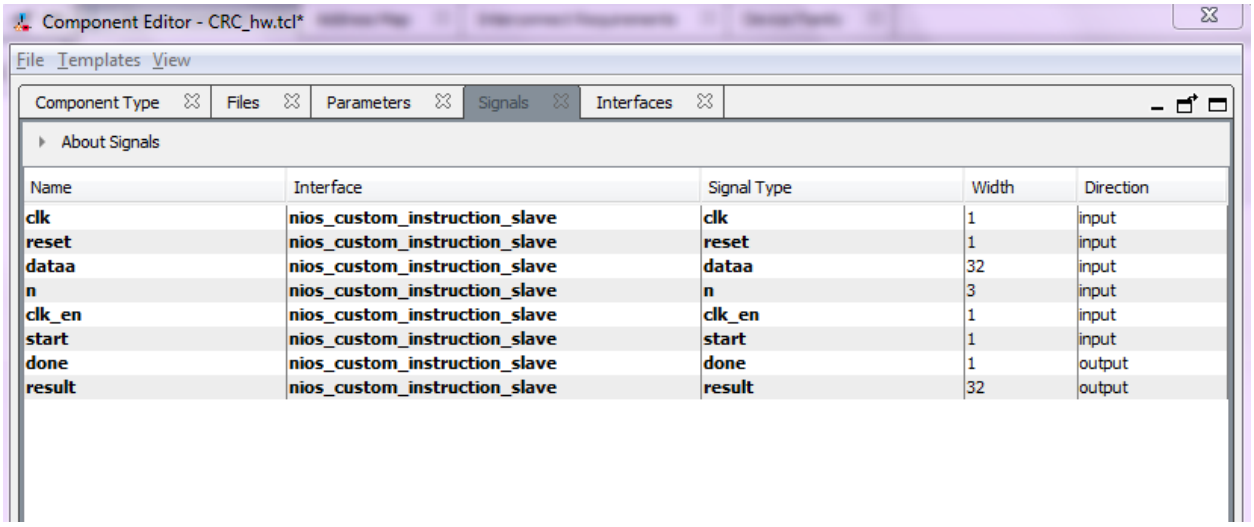


4) Click on next. The signals tab is displayed. Here, for the 1st signal (the clock) in the interface section select the "New Custom Instruction Slave". After choosing this, the interface section will display the new custom instruction slave as the nios_custom_instruction_slave.

In the interface section select nios_custom_instruction_slavefor all the signals.

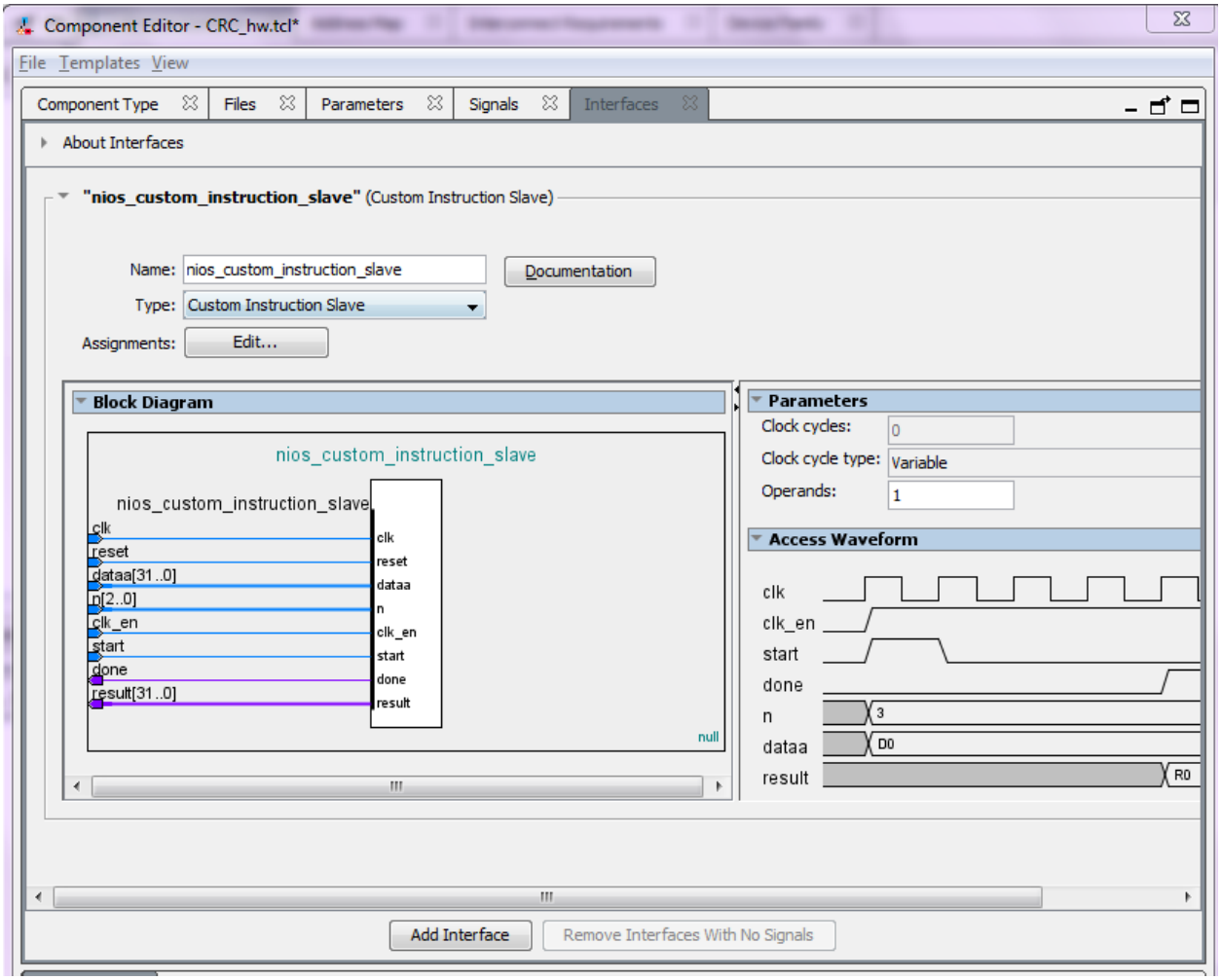
In the Signal type select the same option as the signal name.

After selecting all the above the signals tab will have values as shown below.



Name	Interface	Signal Type	Width	Direction
clk	**nios_custom_instruction_slave**	**clk**	1	input
reset	**nios_custom_instruction_slave**	**reset**	1	input
dataa	**nios_custom_instruction_slave**	**dataa**	32	input
n	**nios_custom_instruction_slave**	**n**	3	input
clk_en	**nios_custom_instruction_slave**	**clk_en**	1	input
start	**nios_custom_instruction_slave**	**start**	1	input
done	**nios_custom_instruction_slave**	**done**	1	output
result	**nios_custom_instruction_slave**	**result**	32	output

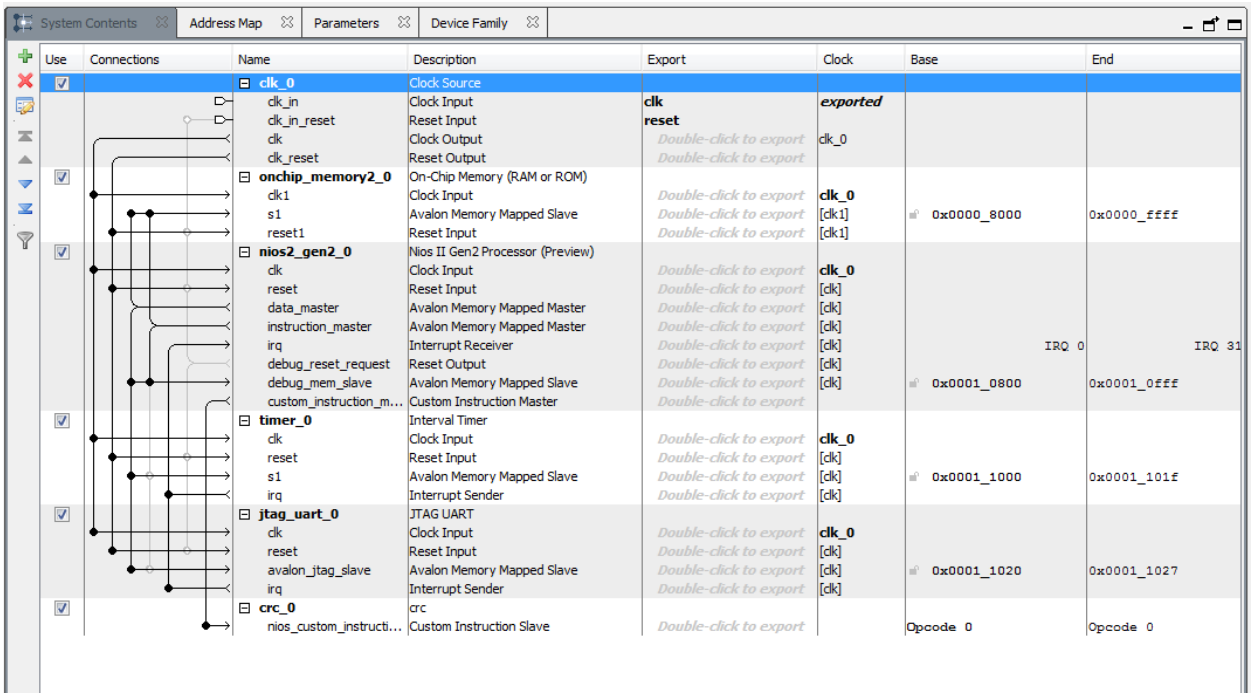
5) Next Click on Interfaces. Click on Remove Interfaces with no signals. In the Parameters options select the number of operands as 1.



6)Next, Click on finish. Save the changes to CRC_hw.tcl on being prompted.

Now a new component CRC will appear under the new component category in the IP catalogue.

7) Add the other components, such as the Nios, On Chip Memory, Timer, Jtag Uart to the Qsys system and connect the ports. The overall Qsys system looks as below:


System Contents
Address Map
Parameters
Device Family
Use
Connections
Name
Description
Export
Clock
Base
End
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
onchip_memory2_0
On-Chip Memory (RAM or ROM)
clk1
Clock Input
Double-click to export
clk_0
s1
Avalon Memory Mapped Slave
Double-click to export
[clk1]
0x0000_8000
0x0000_ffff
reset1
Reset Input
Double-click to export
[clk1]
nios2_gen2_0
Nios II Gen2 Processor (Preview)
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
timer_0
Interval Timer
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]
jtag_uart_0
JTAG UART
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_1020
0x0001_1027
irq
Interrupt Sender
Double-click to export
[clk]
crc_0
crc
nios_custom_instructi...
Custom Instruction Slave
Double-click to export
Opcode 0
Opcode 0