Custom Instruction Interface

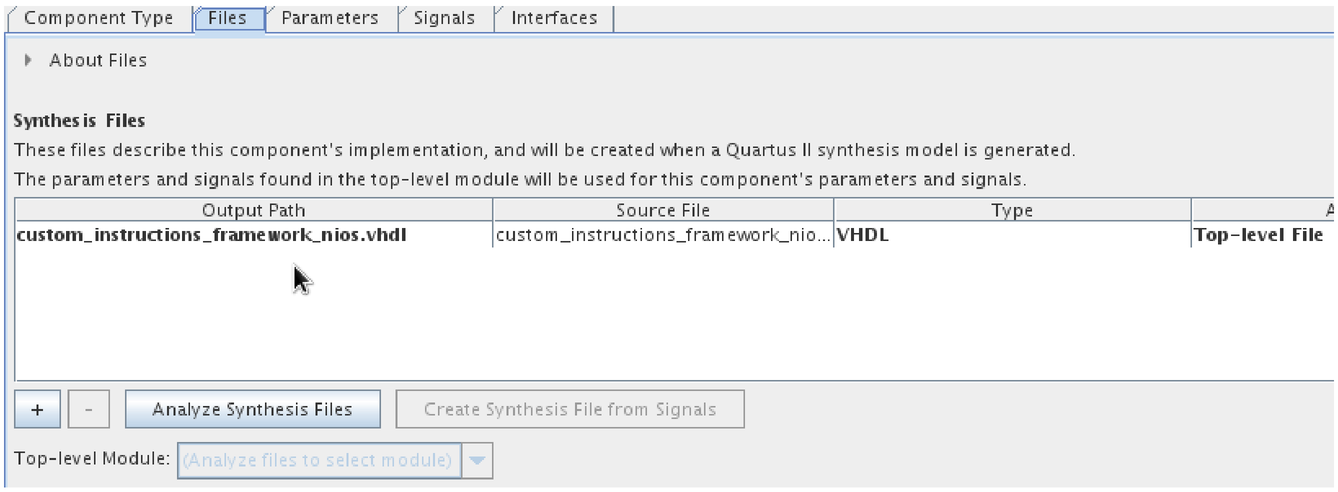
Requirements

- Quartus II 13.0sp1 Installation (as installed in the LiveLinux USB stick)
- FPGA4U development board
- LCD extension board

Generate the Hardware

In this exercise you will make your own custom instruction component for the Nios II. A template is already provided in the previous exercise you made. Check if you Qsys system already has the custom instruction component attached. If yes, you may proceed to "Generate the Software".

1. Create a new component in QSYS
2. Open Qsys (Quartus II → Tools → Qsys) and select "base_system_sdram.qsys".
3. Create a new component (File → New Component …)
4. Change the name in the tab "Component Type) to "custom_instruction"
5. In the tab "Files" add the file **custom_instructions_framework_nios.vhdl** and click on "Analyze Synthesis Files". (See image below)

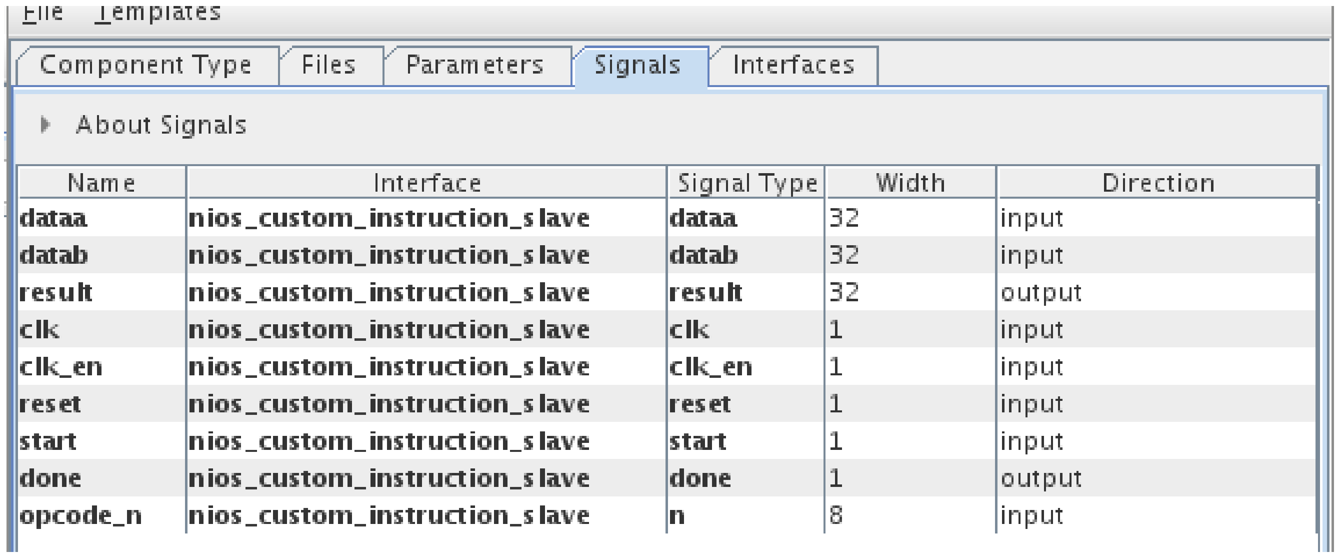


Synthesis Files
These files describe this component's implementation, and will be created when a Quartus II synthesis model is generated.
The parameters and signals found in the top-level module will be used for this component's parameters and signals.

Output Path	Source File	Type	A
custom_instructions_framework_nios.vhdl	custom_instructions_framework_nio...	VHDL	Top-level File

Top-level Module: (Analyze files to select module)

6. Open the "Signals" tab and choose for the first signal: Interface → new custom instruction slave
7. Choose the same interface for the rest of the signals (do not generate a new one)
8. Adapt the signal type according to the following figure:

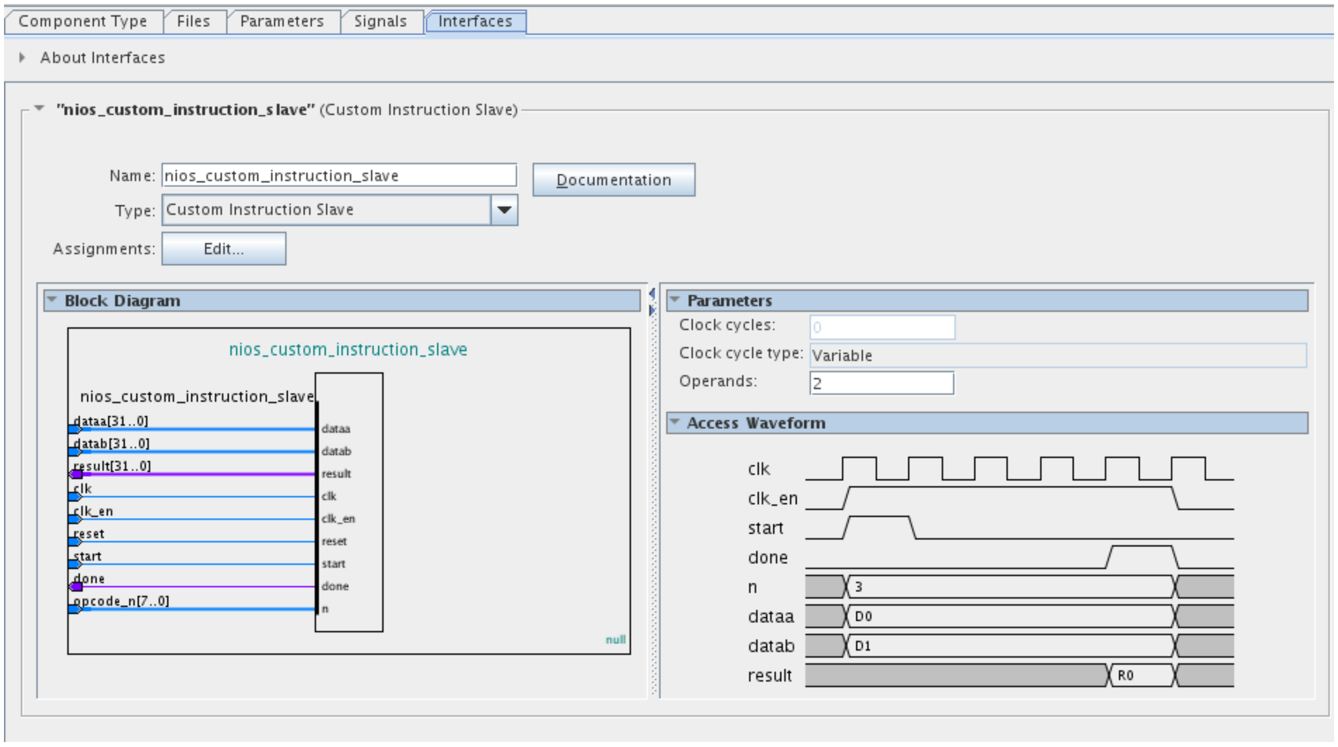

File Templates

Component Type | Files | Parameters | Signals | Interfaces

About Signals

Name	Interface	Signal Type	Width	Direction
dataa	nios_custom_instruction_slave	dataa	32	input
datab	nios_custom_instruction_slave	datab	32	input
result	nios_custom_instruction_slave	result	32	output
clk	nios_custom_instruction_slave	clk	1	input
clk_en	nios_custom_instruction_slave	clk_en	1	input
reset	nios_custom_instruction_slave	reset	1	input
start	nios_custom_instruction_slave	start	1	input
done	nios_custom_instruction_slave	done	1	output
opcode_n	nios_custom_instruction_slave	n	8	input

9. Open the tab “Interface” and click “Remove Interfaces With No Signals”. You new custom instruction interface show look like this:


Component Type | Files | Parameters | Signals | Interfaces

About Interfaces

"nios_custom_instruction_slave" (Custom Instruction Slave)

Name: nios_custom_instruction_slave

Type: Custom Instruction Slave

Assignments: Edit...

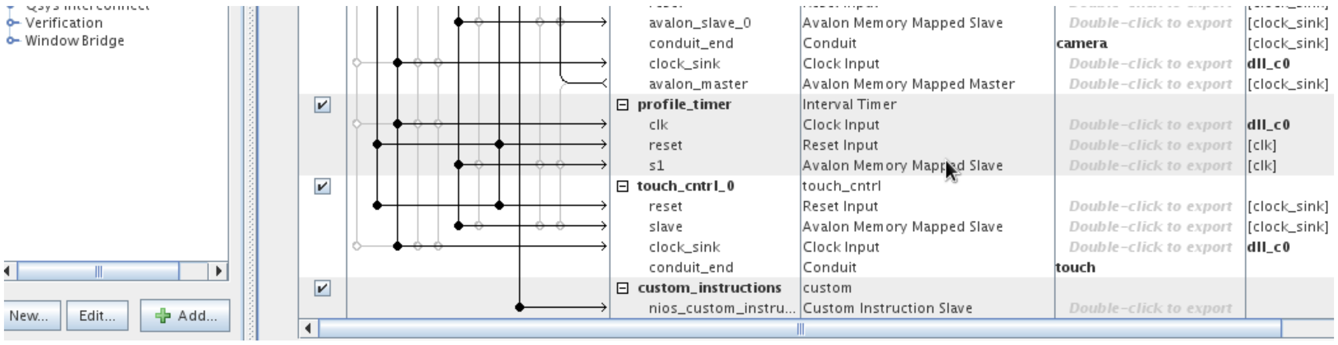
Parameters

Clock cycles: 0

Clock cycle type: Variable

Operands: 2

10. Click “Finish” and add the new component to the Qsys system and connect it:



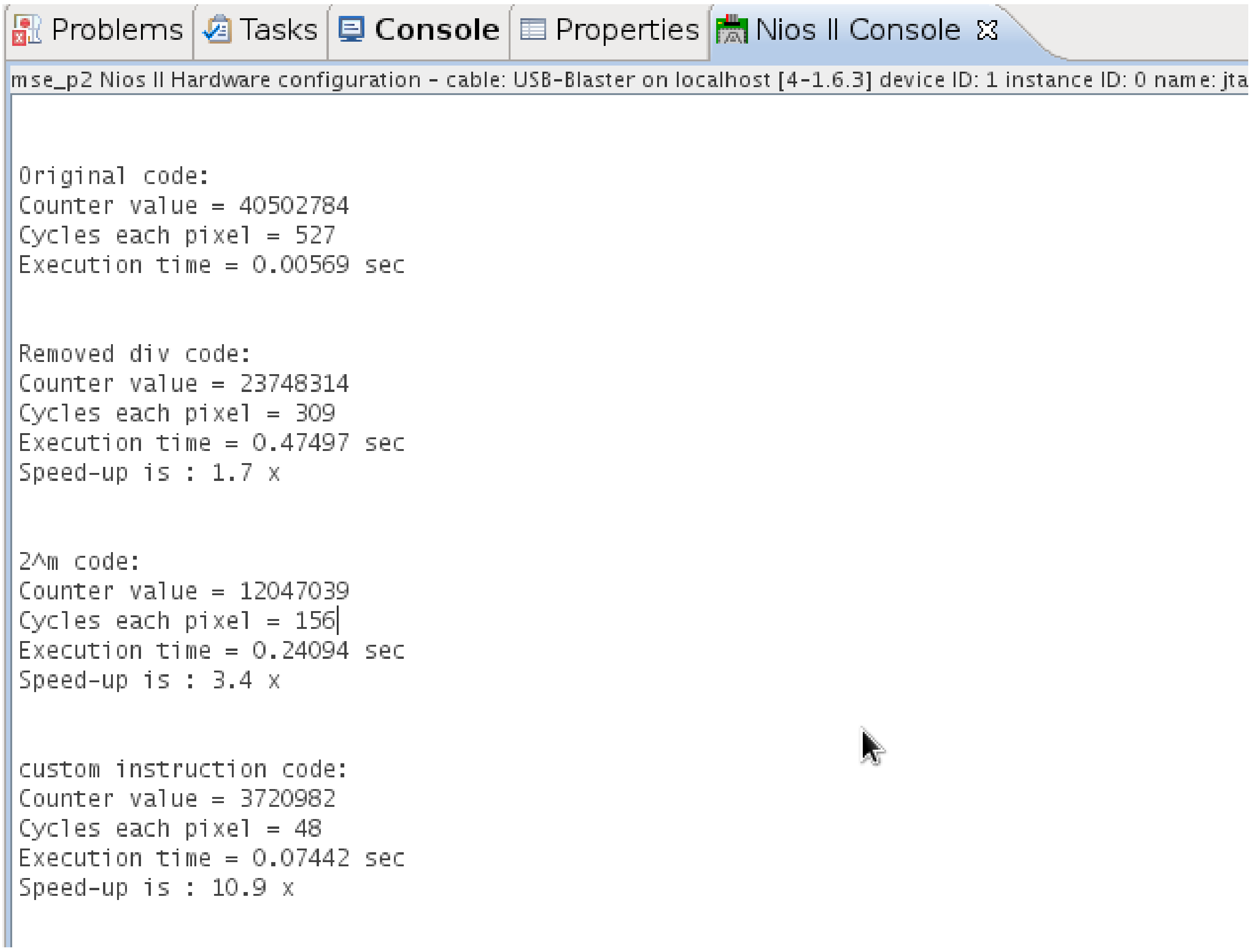
	avalon_slave_0	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
	conduit_end	Conduit	camera	[clock_sink]
	clock_sink	Clock Input	Double-click to export	dll_c0
	avalon_master	Avalon Memory Mapped Master	Double-click to export	[clock_sink]
✓	profile_timer	Interval Timer		
	clk	Clock Input	Double-click to export	dll_c0
	reset	Reset Input	Double-click to export	[clk]
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]
✓	touch_cntrl_0	touch_cntrl		
	reset	Reset Input	Double-click to export	[clock_sink]
	slave	Avalon Memory Mapped Slave	Double-click to export	[clock_sink]
	clock_sink	Clock Input	Double-click to export	dll_c0
	conduit_end	Conduit	touch	
✓	custom_instructions	custom		
	nios_custom_instru...	Custom Instruction Slave	Double-click to export	

11. Generate your System, exit Qsys and synthesize you design.
12. Programm the FPGA

Generate the Software

1. Open the NiosII eclipse (Quartus → Tools → Nios II Software build tools for Eclipse)
2. Select a workspace
3. Create a new Project with BSD from Template (File → New → Nios II Application and BSD from Template)
4. Choose **base_system_sdram.sopcinfo** as “SOPC Information File name”
5. Choose “Hello World” from the Project Templates
6. Set a project name and press on “Finish”.
7. To test the custom instruction: Copy the following files to your project folder, refresh the project and run it on your NiosII Hardware.
   - grayscalehw.c
   - grayscalehw.h
   - KatjaPapa.h
   - hello_world.c

In the console you should see the performance measurements of the custom instruction as show in the following image:



```
mse_p2 Nios II Hardware configuration - cable: USB-Blaster on localhost [4-1.6.3] device ID: 1 instance ID: 0 name: jta


Original code:
Counter value = 40502784
Cycles each pixel = 527
Execution time = 0.00569 sec


Removed div code:
Counter value = 23748314
Cycles each pixel = 309
Execution time = 0.47497 sec
Speed-up is : 1.7 x


2^m code:
Counter value = 12047039
Cycles each pixel = 156
Execution time = 0.24094 sec
Speed-up is : 3.4 x


custom instruction code:
Counter value = 3720982
Cycles each pixel = 48
Execution time = 0.07442 sec
Speed-up is : 10.9 x
```

Now use this example to develop your own custom instruction:

- You can add your own custom instruction hardware to the file **custom_instructions_framework_nios.vhdl**. If you modify your hardware, the Qsys System needs to be regenerated and a new synthesis must be started.
- Test your new custom instruction within the C-Framework (rgb_to_grayscale) provided.