GUIDE SOPC CUSTOM MM COMPONENT

Goal

Create a custom component in Qsys with an MM Interface.

Prerequisites

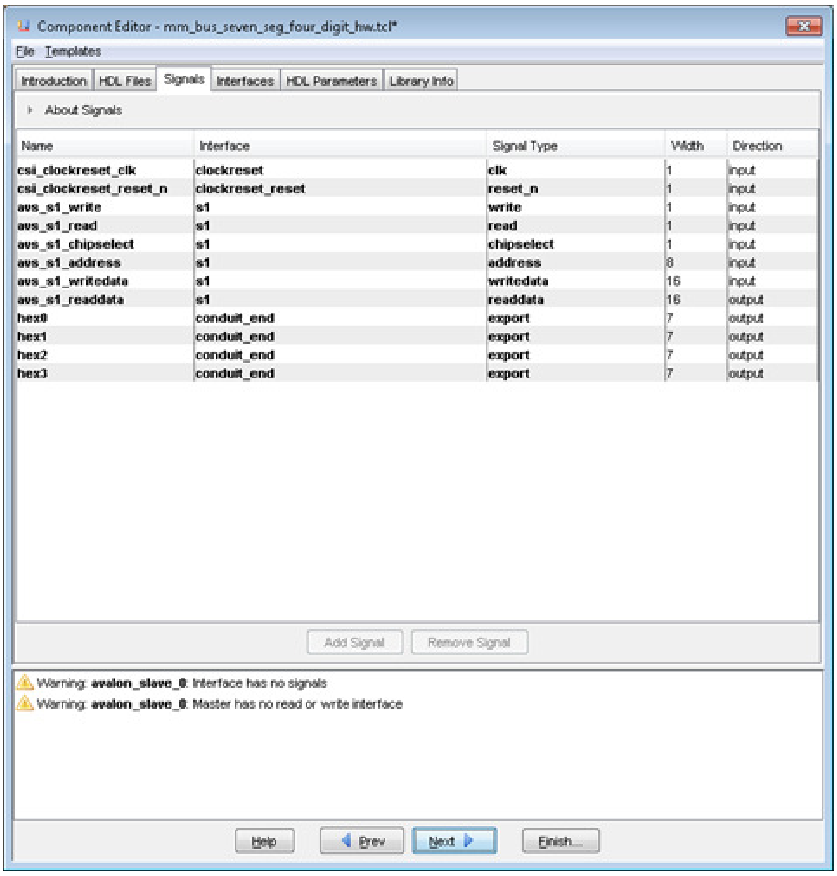
- Quartus
- You must be familiar with Qsys
- Use the FIR Compiler II MegaCore Function User Guide as reference for this guide

How To Do It

A) With Qsys open, press "new" in the "Component Library" panel to the left. This opens the following dialog:

Add your hdl files and let it analysed. If analysis fails, you should go back to ModelSim and debug your code. Ensure that your interfaces are correct, that you have no unresolved generics and that the code can be syntehsized (no test bench constructions).

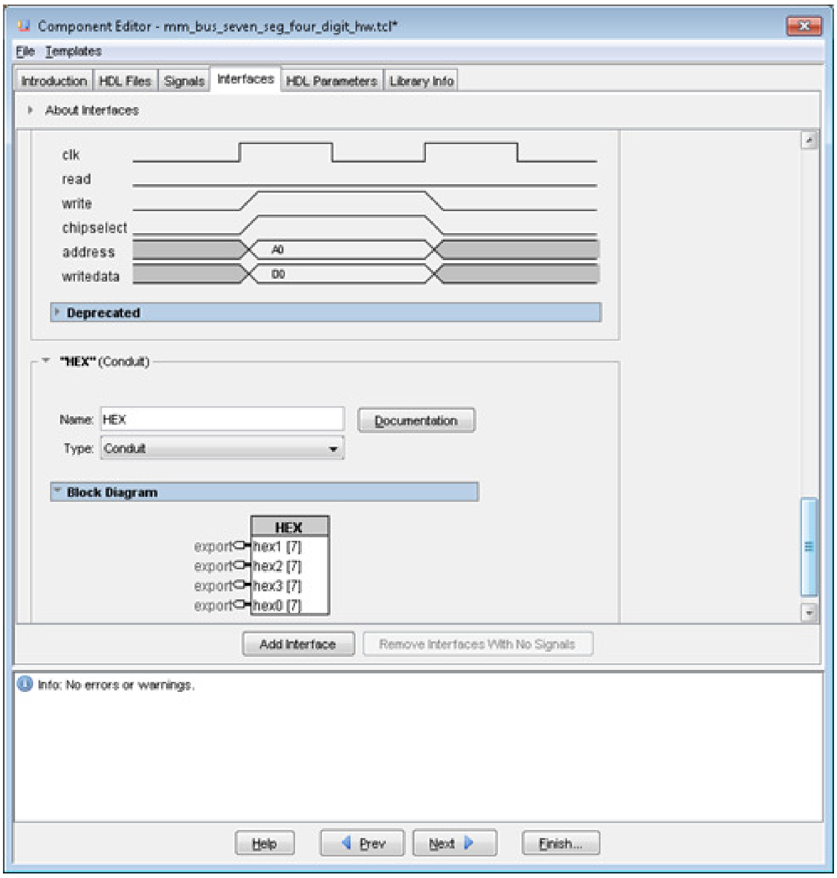
B) Identify the interfaces. The next page lets you assign the design entity ports to well-known interfaces (Avalon-MM / Avalon-ST / clock / conduit) See mnl_avalon_spec.pdf Right-click on the interface and select ex "new Avalon Memory Mapped Interface" if needed. Using the avs_s1... names should let the builder map the signals automatically. Re-use this interface for the remaining MM signals. Create a new "Conduit interface" for the hex outputs and set the for "export". Note that the "clk" and "reset_n" both must use the same clock interface. This way we indicate that they are related.


Component Editor - mm_bus_seven_seg_four_digit_hw.tcl*
File Templates
Introduction HDL Files Signals Interfaces HDL Parameters Library Info
About Signals
Add Signal Remove Signal
Help Prev Next Finish...

Name	Interface	Signal Type	Width	Direction
csi_clockreset_clk	clockreset	clk	1	input
csi_clockreset_reset_n	clockreset_reset	reset_n	1	input
avs_s1_write	s1	write	1	input
avs_s1_read	s1	read	1	input
avs_s1_chipselect	s1	chipselect	1	input
avs_s1_address	s1	address	8	input
avs_s1_writedata	s1	writedata	16	input
avs_s1_readdata	s1	readdata	16	output
hex0	conduit_end	export	7	output
hex1	conduit_end	export	7	output
hex2	conduit_end	export	7	output
hex3	conduit_end	export	7	output

Warning: **avalon_slave_0**: Interface has no signals
Warning: **avalon_slave_0**: Master has no read or write interface

C) Going to the "Interfaces" page, you should start out by pressing "Remove Interfaces with no Signals" if it is not faded out. Select the clock signal. A reset signal may not be required. Note the waveform at the bottom, this should reflect what you want.

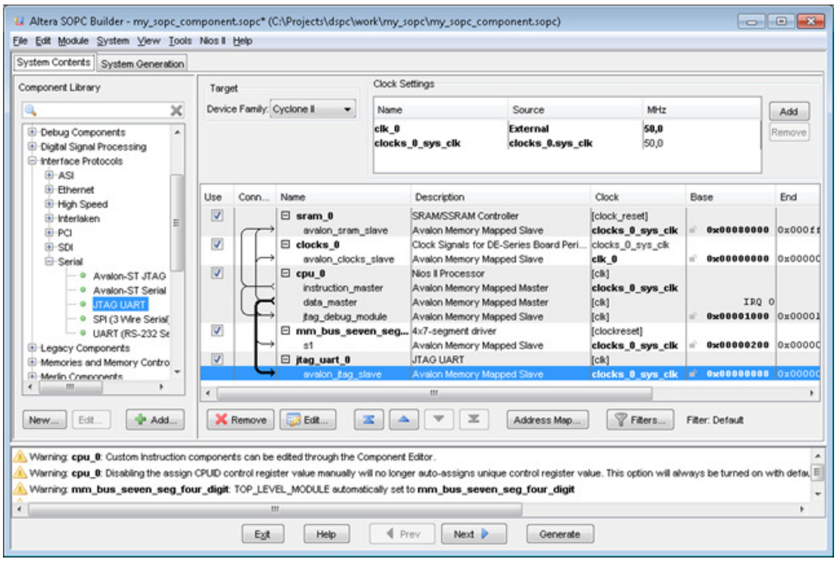


Name: HEX
Type: Conduit

Info: No errors or warnings.

D) In the next page you may specify author, group and a lot more. Adding a group, say "ASE" will put your custom component in an ASE folder in the SOPC component library. Press Finish

E) You are now able to add your custom component to your project. You can right-click and filter the interfaces shown in the overview.



Name	Source	MHz
clk_0	External	50,0
clocks_0_sys_clk	clocks_0.sys_clk	50,0

Use	Conn...	Name	Description	Clock	Base	End
☑		sram_0	SRAM/SSRAM Controller	[clock_reset]		
		avalon_sram_slave	Avalon Memory Mapped Slave	clocks_0_sys_clk	0x00080000	0x000ff
☑		clocks_0	Clock Signals for DE-Series Board Peri...	clocks_0_sys_clk		
		avalon_clocks_slave	Avalon Memory Mapped Slave	clk_0	0x00000000	0x0000C
☑		cpu_0	Nios II Processor	[clk]		
		instruction_master	Avalon Memory Mapped Master	clocks_0_sys_clk		
		data_master	Avalon Memory Mapped Master	[clk]	IRQ 0	
		jtag_debug_module	Avalon Memory Mapped Slave	[clk]	0x00001000	0x00001
☑		mm_bus_seven_seg...	4x7-segment driver	[clockreset]		
		s1	Avalon Memory Mapped Slave	clocks_0_sys_clk	0x00000200	0x0000C
☑		jtag_uart_0	JTAG UART	[clk]		
		avalon_jtag_slave	Avalon Memory Mapped Slave	clocks_0_sys_clk	0x00000008	0x0000C

Warning: **cpu_0**: Custom Instruction components can be edited through the Component Editor.
Warning: **cpu_0**: Disabling the assign CPUID control register value manually will no longer auto-assigns unique control register value. This option will always be turned on with defau
Warning: **mm_bus_seven_seg_four_digit**: TOP_LEVEL_MODULE automatically set to **mm_bus_seven_seg_four_digit**

Finally, your Qsys component should now contain the hex interfaces for the seven-segment displays, you will have to pin-map these like the remaining signals. In Qsys see tools->show instantiation, to see how the new interfaces are named, copy these and insert them into the top-level .vhd file in Quartus:

```
    component soc_system is
        port (
            clk_clk                                : in    std_logic
            ...
            hex_id1                                : out   std_logic_vector(6 downto
            hex_id2                                : out   std_logic_vector(6 downto
            hex_id3                                : out   std_logic_vector(6 downto
            hex_id4                                : out   std_logic_vector(6 downto
                );
     end component soc_system;

begin

     u0 : component soc_system
         port map (
             clk_clk                               => CLOCK_50,
             .....
             hex_id1                               => HEX0,
             hex_id2                               => HEX1,
             hex_id3                               => HEX2,
             hex_id4                               => HEX3
         );
```