

Quartus Compile Info Example Design Description

- 1. This example design demonstrates an automated technique to add custom build information to a Quartus compile. In this example the build information is retrieved using a Tcl script run within System Console (An Intel FPGA tool which provides a debugging infrastructure). The custom build information is added to the bitstream programming file by means of the automatic creation of a Memory Initialization File (.mif) used to initialize values within a specified RAM block in the design. In this example the following build information is captured and stored into the design during compilation:

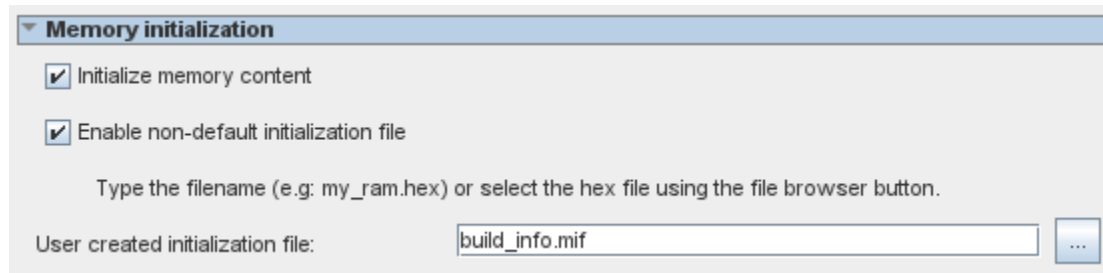
```
Script Version: 1.0
Build Date: 01/26/2023
Build Time: 16:24:01
User: kweldon
Quartus: Version 22.4.0 Build 94 12/07/2022 SC Pro Edition
Project Name: top
Revision Name: top
OS: Windows NT
OS Version: 6.2
```

- 2. The example design is created using Intel’s Platform Designer (system integration tool). Figure 1 below shows the entire design in Platform Designer. The design consists of a Clock Bridge, a Reset Bridge, a JTAG to Avalon Bridge and On-Chip Memory (a RAM block). Both the Clock Bridge and the Reset Bridge have their input port exported (see the Export column in the image below). These two ports (clk and reset) are the only two top level ports that exists when the Platform Designer system is generated (producing a top-level RTL file).

Connections	Name	Description	Export
	clock_in	Clock Bridge Intel FPGA IP	
	in_clk	Clock Input	clk
	out_clk	Clock Output	Double-clk
	reset_in	Reset Bridge Intel FPGA IP	
	clk	Clock Input	Double-clk
	in_reset	Reset Input	reset
	out_reset	Reset Output	Double-clk
	master	JTAG to Avalon Master Bridge Intel FPGA IP	
	clk	Clock Input	Double-clk
	clk_reset	Reset Input	Double-clk
	master_reset	Reset Output	Double-clk
	master	Avalon Memory Mapped Host	Double-clk
	device_info_ram	On-Chip Memory (RAM or ROM) Intel FPGA IP	
	clk1	Clock Input	Double-clk
	s1	Avalon Memory Mapped Agent	Double-clk
	reset1	Reset Input	Double-clk

The JTAG to Avalon Bridge provides access between a host system (in this case our `get_build_info.tcl` Tcl script run using the System Console tool) and the Avalon memory-mapped system within Platform Designer (in this case the entire memory-map consists of only one RAM block).

The On-Chip Memory block is a single port RAM block wrapped in an Avalon interface. Avalon is an open standard providing an easy-to-use protocol for memory-mapped and streaming interfaces. By right-clicking on the On-Chip Memory block in Platform Design and then selecting “Edit Parameters...” you can inspect the configuration parameters for the RAM block. The block is configured to have a data width of 8-bits and a total memory size of 2048 bytes (8-bit address bus). The block is configured to consume one M20K block in the Arria 10 device which was selected for this example. You can also see (you may need to scroll down a bit) that the RAM block is associated with a “`build_info.mif`” file for its memory initialization. This is shown in the image below.



From within Quartus you could create a `.mif` memory initialization file by selecting File->New->Memory Initialization File. We do not need to do that in this example as the `build_info.mif` file will be generated automatically as part of the Quartus compilation.

To generate the RTL design used in Quartus compilation you can press the “Generate HDL...” button in the bottom right corner of the Platform Designer GUI.

3. The `build_info.mif` file is created by the `generate_build_info.tcl` Tcl script during Quartus compilation. Quartus is instructed to run the `generate_build_info.tcl` script after compilation by the addition of the following line to the project’s QSF file (`top.qsf`).

```
set_global_assignment -name POST_FLOW_SCRIPT_FILE "quartus_sh:generate_build_info.tcl"
```

The `generate_build_info.tcl` script creates the `build_info.mif` file, runs the update MIF utility and then run the assembler to build an updated programming file.

The information added to the `build_info.mif` file consists of a single string named “`device_info`” in the `generate_build_info.tcl` script. The `device_info` string variable is appended to for every piece of information that is desired to be captured into RAM. For example, the build data information is added to the `device_info` string with the following two lines found in the `generate_build_info.tcl` script:

```
set data [clock format [clock seconds] -format {Build Date: %m/%d/%Y}]
```

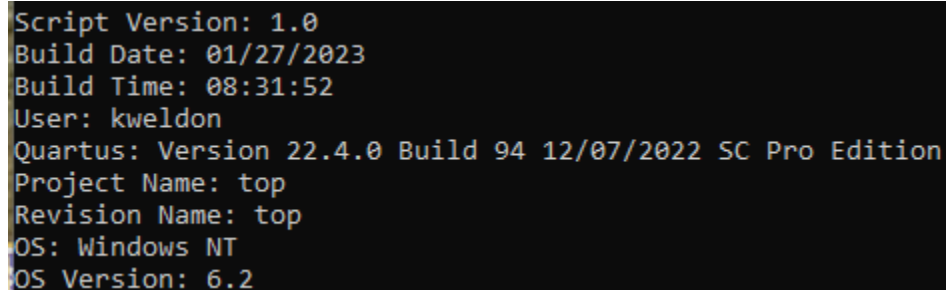
```
set device_info "${device_info}${data}\n"
```

As an example, the string "Hello World" could be added to the build_info.mif file by adding the following two lines to the generate_build_info.tcl script:

```
set data "Hello World"  
set device_info "${device_info}${data}\n"
```

The create_memory_data procedure within the generate_build_info.tcl script takes as input the device_info string and converts each character in the string into an ASCII value used to be written to the build_info.mif file. The build_info.mif file is created within the write_mif_file procedure.

4. After compilation the get_build_info.tcl script is used to read the contents of the device_info_ram RAM block and display its contents to the console.
5. The following steps describe how to run the example design.
 - a. Double click the open_cmd_prompt.bat file
 - b. Run the restore_qar.bat file by typing "restore_qar.bat" and pressing the return key.
 - c. Open the design in Quartus
 - i. You can use the Quartus GUI File->Project open to navigate to open the top.qpf file.
 - ii. If your PATH environment variable includes a path to Quartus you can type "quartus top"
 - d. You will likely need to update the device and clk pin assignment to match your development board.
 - i. From within Quartus select Assignments->Device to change the current device assignment
 - ii. Select Assignments->Assignment Editor to change the clk pin assignment.
 - e. The reset pin exported from the Platform Designer system is connected to an In-System Sources and Probes block which is toggled when the program_fpga.tcl script is executed.
 - f. You can compile the design from within Quartus or from the command line run the compile_design.bat file.
 - g. To program the design run the program_fpga.bat file
 - h. To view the build information stored in the device_info_ram run the get_build_info.tcl script. The output should look similar to the following image.



```
Script Version: 1.0  
Build Date: 01/27/2023  
Build Time: 08:31:52  
User: kweldon  
Quartus: Version 22.4.0 Build 94 12/07/2022 SC Pro Edition  
Project Name: top  
Revision Name: top  
OS: Windows NT  
OS Version: 6.2
```

6. Next.

