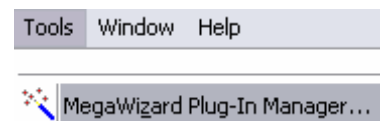


Implementing the MC8051 IP Core On A Cyclone Nios Board

First of all it is necessary to exchange the simulation models of all the memory blocks with real memory that can be found inside the target FPGA. It is also recommended to implement a PLL to get a clock signal with a lower frequency than that of the on-board oscillator. The VHDL code for these entities is generated by the backend tool, i.e. *Quartus II 4.0* for Altera FPGAs.

Step 1: Choose the function that should be generated.

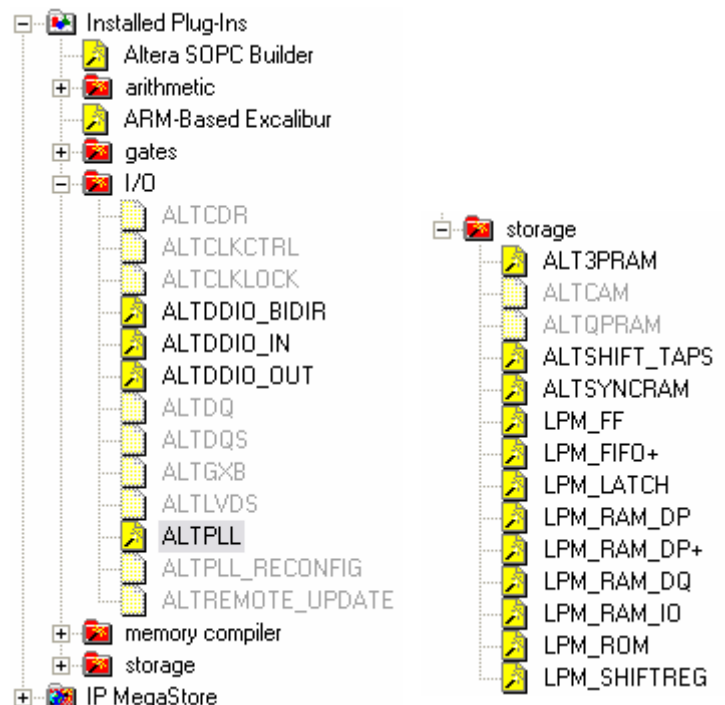
1. After starting Quartus II, launch the *MegaWizard Plug-In Manager* that is located in the *Tools*-Menu.



2. Select the function block that should be generated.

The PLL will be designed first. This function can be found in the *I/O* folder and is named *ALTPLL*.

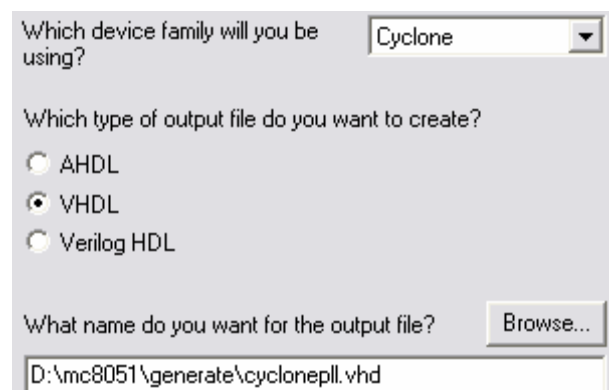
In a second turn the memories will be drawn. The functions for those blocks can be found in the folder named *storage*. It contains the functions *LPM_ROM* for the *mc8051_rom* entity and *LPM_RAM_DQ* for the entity *mc8051_ram* and *mc8051_ramx* respectively.



3. Set the device family to *Cyclone*.

Then choose the language of the HDL output file.

Finally the name of the output file and its destination directory must be set. It is recommended to store all code files belonging to FPGA functions in a directory named *generate*.



When the function has been selected and the output path has been chosen, the behaviour and the parameters of this function must be defined. This must be done for the PLL, the ROM and the internal and external RAM.

Step 2: Define the behaviour and the parameters of the PLL.

1. Enter the input frequency for the PLL. In the case of the NIOS Cyclone Board this frequency is 50 MHz.

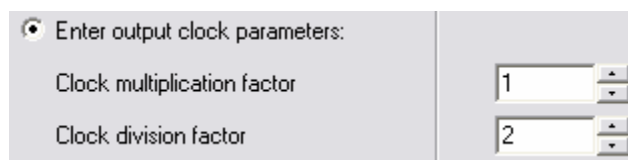
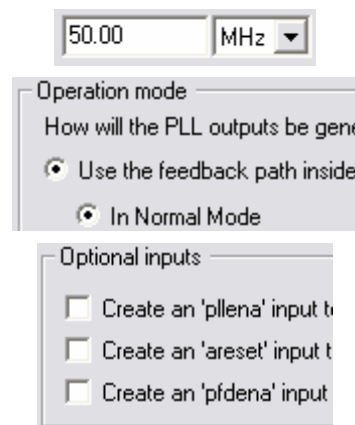
The PLL operates with an *internal feedback* in *normal mode*.

2. It is also possible to add optional inputs to the PLL. To keep the design simple, none of the inputs are selected.

3. If only one clock output is used, this output is automatically set to *c0*. Entering the value of the frequency or selecting the factor for the multiplication and the division respectively can define the output frequency of the PLL.

Be sure that the chosen settings can be implemented which is shown at the top of the window.

4. Since no other PLL outputs are selected for operation, the remaining dialogs can be skipped. Finally all files that will be generated are shown.



c0 - Core Output Clock
Able to implement the requested PLL

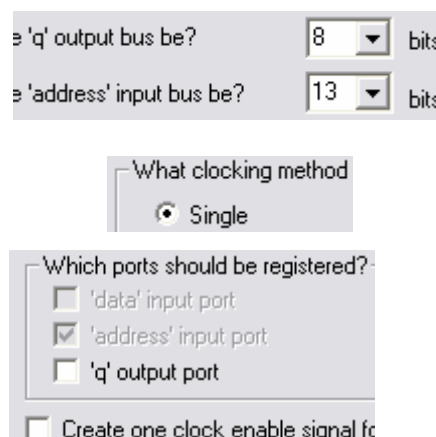
Step 3: Define the behaviour and the parameters of the ROM.

1. As the ROM is a memory function, the size of the memory and the width of the data bus have to be defined. In the case shown at the right, the ROM is organized 8k x 8 bit.

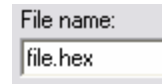
To meet the signals of the MC8051 design, single clocking must be used.

2. The data output *q* of the ROM must not be registered for the MC8051 design since it is not supposed to be.

As the ROM and the RAM blocks have designated address and data busses, there is no need for a clock enable signal.



3. The program file must be loaded into the memory.
This can be achieved by pointing to the HEX file that contains the user program.
4. Confirm the generation of the files.

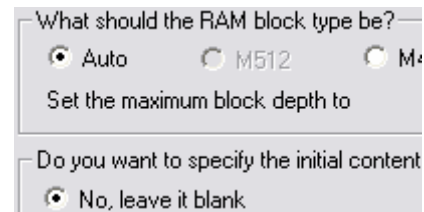


File name:
file.hex

Step 4: Define the behaviour and the parameters of the internal and external RAM.

1. Enter the width of the address and data bus as shown for the ROM.
2. Select the behaviour of the input and output as shown for the ROM.

3. Let the software decide how to implement the RAM block. If it is necessary, the RAM can be preloaded with an additional HEX file.



What should the RAM block type be? —
 Auto M512 M:
Set the maximum block depth to
Do you want to specify the initial content
 No, leave it blank

4. Confirm the generation of the files.

Since the package file and the top level design refers to the simulation models for the memory blocks, the declaration of the ports and the wiring must be updated for exchange with the implementation models. If required the PLL must also be added to the design.

Step 5: Update the component declaration in the package file.

After all files for the implementation models have been generated, the component declaration for the memories in the package file *mc8051_p.vhd* must be updated to the entity declaration that can be found in the generated code file. For the case of the additional PLL it is also necessary to add a new component declaration for that entity. The package file should contain the following component declarations.

```
component mc8051_ram
  port (address : in  std_logic_vector(6 downto 0); -- address input
        clock   : in  std_logic;                -- clock input
        data    : in  std_logic_vector(7 downto 0); -- data input
        wren    : in  std_logic;                -- write enable
        q       : out std_logic_vector(7 downto 0)); -- data output
end component;

component mc8051_ramx
  port (address : in  std_logic_vector(12 downto 0); -- address input
        clock   : in  std_logic;                -- clock input
        data    : in  std_logic_vector(7 downto 0); -- data input
        wren    : in  std_logic;                -- write enable
        q       : out std_logic_vector(7 downto 0)); -- data output
end component;

component mc8051_rom
  port (address : in  std_logic_vector(12 downto 0); -- address input
        clock   : in  std_logic;                -- clock input
        q       : out std_logic_vector(7 downto 0)); -- data output
end component;

component cyclonepll
  port (inclk0 : in  std_logic; -- PLL input
        c0     : out std_logic); -- PLL output
end component;
```

Step 6: Update the wiring of the top level design.

The wiring between the entities found in the top level design *mc8051_top_struct.vhd* also has to be updated. If the width of the address busses of the memories differs from the 16 bit wide busses provided by the MC8051 core, the smaller bus signals *s_rom_adr_sml* and *s_ramx_adr_sml* have to be inserted. If the PLL should be implemented, a new clock signal *clk_pll* has to be added between the PLL output and the components of the top level design.

Since the switches located on the NIOS Cyclone Board are active low, the active high reset signal *s_reset* for the MC8051 core can be gained from the reset switch by an additional inverter.

```
architecture struc of mc8051_top is

  signal s_clk_pll      : std_logic; -- core clock, PLL output
  signal s_reset       : std_logic; -- reset signal, active high

  signal s_rom_adr_sml : std_logic_vector(12 downto 0); -- *** new
  signal s_ramx_adr_sml : std_logic_vector(12 downto 0); -- *** new

begin
  -- architecture structural

  s_rom_adr_sml <= std_logic_vector(s_rom_adr(12 downto 0));
  s_ramx_adr_sml <= std_logic_vector(s_ramx_adr(12 downto 0));
  s_reset <= not reset;

  i_mc8051_core : mc8051_core
    port map (clk      => s_clk_pll, -- mc8051 core
              reset    => s_reset,
              ...
    );
```

```
i_mc8051_ram : mc8051_ram
  port map (address => s_ram_adr,      -- internal RAM
            clock  => s_clk_pll,
            clken  => s_ram_en,
            data   => s_ram_data_in,
            wren   => s_ram_wr,
            q      => s_ram_data_out);

i_mc8051_rom : mc8051_rom
  port map (address => s_rom_adr_sml,  -- ROM
            clock  => s_clk_pll,
            clken  => s_rom_en,
            q      => s_pre_rom_data);

i_mc8051_ramx : mc8051_ramx
  port map (address => s_ramx_adr_sml, -- external RAM
            clock  => s_clk_pll,
            clken  => s_ramx_en,
            data   => s_ramx_data_out,
            wren   => s_ramx_wr,
            q      => s_ramx_data_in);

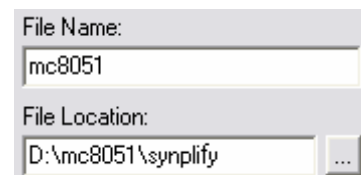
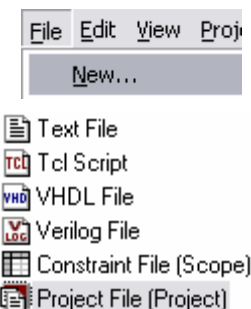
i_cyclonepll : cyclonepll
  port map (inclk0 => clk,            -- PLL
            c0     => s_clk_pll);
```

When the VHDL files for the MC8051 have been changed for implementation, it is time for synthesis. *Synplify Pro 7.5* is used for that steps and generates a single netlist file that can be then used for implementation.

Step 7: Generate a new project file for synthesis.

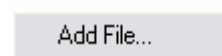
1. After starting Synplify Pro, open the *File* menu and select *New*.
2. A window appears where the type of the files that should be generated can be selected. At that point, choose *Project File*.
3. Set the file name for the project file and point to the working directory for Synplify Pro where the files should be stored.

After confirming a new project will be opened.



Step 8: Add the source files to the project.

1. To define which source code files should be used, hit the *Add File* button.

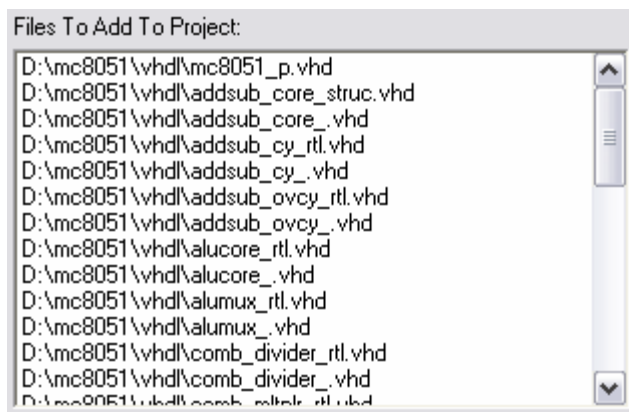


2. Add the VHDL files in the following way:

First add the package file *mc8051_p.vhd*.

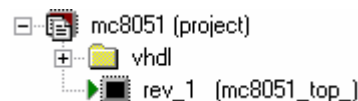
Then select all files that contain the entities of the top level design. The VHDL configuration files need not to be added.

Finally add the top level design files to the project: *mc8051_top.vhd* and *mc8051_top_struct.vhd*.



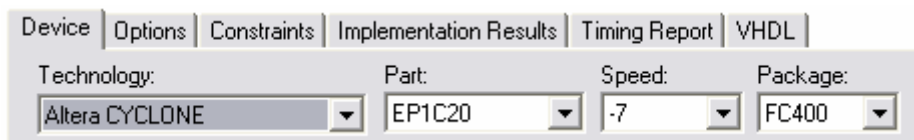
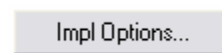
Code files of components that depend on the target technology such as PLLs or memory blocks must not be added to the project for synthesis.

3. After confirming a new folder named *vhdl* appears in the project window.

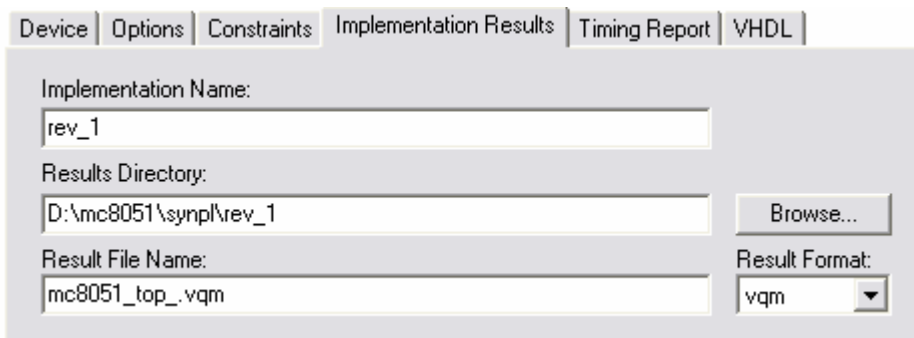


Step 9: Set up the implementation options.

1. To choose the target technology and the file format for the output of the synthesis, hit the *Impl Options* button that is located on the left of the project window.



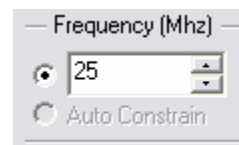
2. As the window appears, the *Device* folder is selected where the target FPGA can be declared. To implement the design for the NIOS Cyclone Board, the FPGA that is located at this board must be chosen. For that case use the options as shown above.



3. It is also possible to change the output directory and the file format for the output file. Swap to the *Implementation Results* folder where those options are specified. If Quartus II

is used to implement the design, the file format should be set to *vqm* (Verilog Quartus Mapping).

4. Before running the synthesis, the operating frequency should be entered at the project window. If a PLL is used, the frequency at the output of the PLL has to be entered.



Step 10: Run the synthesis and check the warnings and the notes.

1. The synthesis can be started by hitting the *Run* button which first runs the compiler. If there are no errors in the VHDL code, the mapper generates the output file that can take a few minutes.
2. All messages that are produced during synthesis can be found in the SRR file located in the output directory. These messages are divided into errors, warnings and notes. The first thing to check is which entity was selected as the top level entity. This information can be found in the notes section as shown below.



```
@N: "D:\mc8051\vhd1\mc8051_top_.vhd":74:7:74:16|Top entity is set to mc8051_top.
```

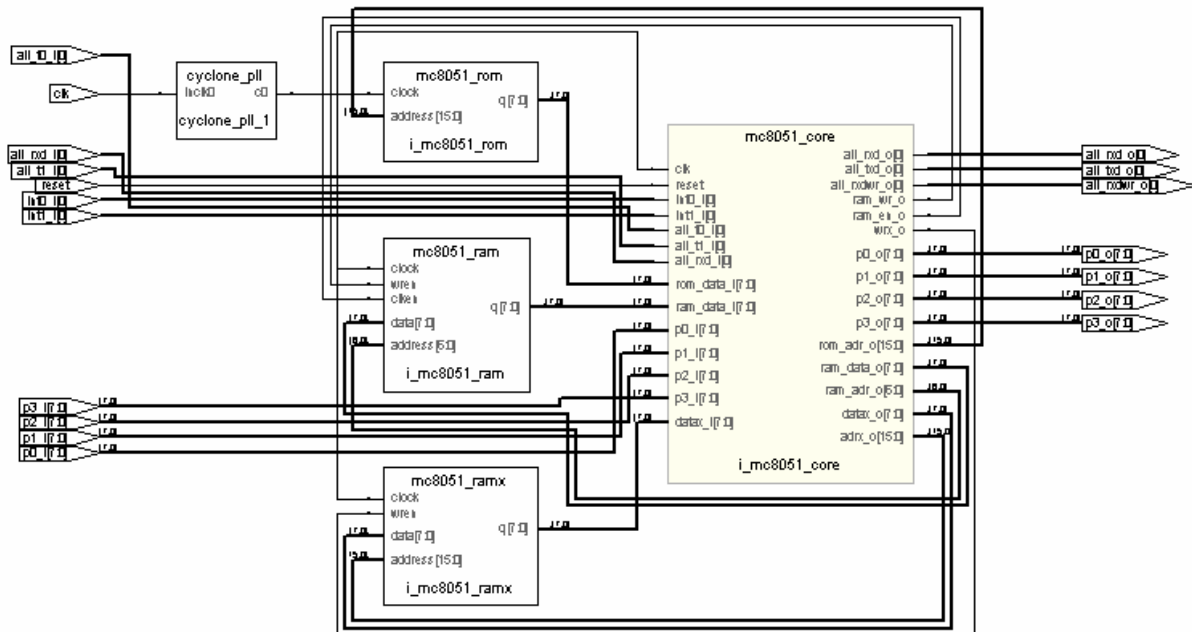
Since the source file for the implementation models of the memories and the PLL depends on the target technology, they were not added for synthesis. Each entity of such a model is treated as a black box and produces a message in the warnings section.

```
@W: CD280 : "D:\mc8051\vhd1\mc8051_p.vhd":808:12:808:21|Unbound component mc8051_ram mapped to black box  
@W: CD280 : "D:\mc8051\vhd1\mc8051_p.vhd":827:12:827:21|Unbound component mc8051_rom mapped to black box  
@W: CD280 : "D:\mc8051\vhd1\mc8051_p.vhd":818:12:818:22|Unbound component mc8051_ramx mapped to black box  
@W: CD280 : "D:\mc8051\vhd1\mc8051_p.vhd":834:12:834:21|Unbound component cyclonepll mapped to black box
```

Step 11: Check the generated RTL schematic.

1. The synthesis generates an RTL schematic in addition to the output file that can be shown via the *RTL view* button that is located in the tool bar.
2. Navigate through the RTL schematic to verify if the design is built up in a correct way. The zoom functions and functions to descend and ascend the hierarchy can be found in the tool bar.





When entering the RTL view, the top level design is shown as above. The core of the MC8051 design and all the input and output ports are displayed whereby the entities for the PLL, the ROM and the RAMs are treated as black boxes.

To descend the hierarchy, select the appropriate function from the tool bar, move the mouse cursor to the desired entity and click to swap to the next level. To ascend the hierarchy to the previous level, move the cursor next to a entity and click when an arrow that points above appears.

If the design is built in a correct way, the implementation can be done. To generate the files that are needed to download the design to the NIOS Cyclone Board, *Quartus II 4.0* is used.

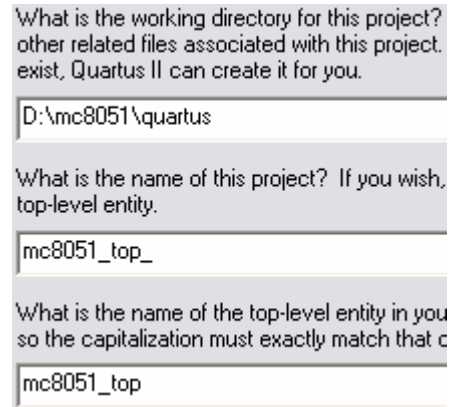
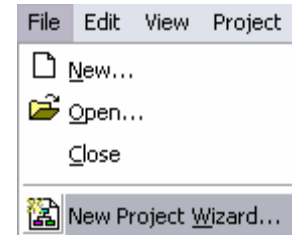
Step 12: Open a new project file for implementation.

1. After starting Quartus II, open the *File* menu and start the *New Project Wizard*.

An introduction screen is shown before the settings for the implementation have to be declared.

2. Choose the working directory where all the files generated by Quartus II will be written to. It is recommended to copy the VQM net list file generated by Synplify Pro to that directory.

The name for the project and the name of the top level entity must also be entered. This can be easily done by pointing to the VQM net list. Take care of the name belonging to the top level entity: there must be no underscore character at the end of the name.



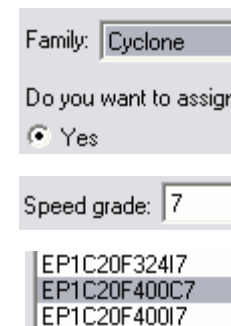
File name	Type
../generate/mc8051_ramx.vhd	VHDL File
../generate/mc8051_rom.vhd	VHDL File
../generate/cyclonepll.vhd	VHDL File
../generate/mc8051_ram.vhd	VHDL File
mc8051_top_.vqm	Verilog Quartus Mapping File

3. Choose all source files for the project. As shown above, it is necessary to add the VQM net list and all the VHDL files for the black boxes which can be found in the *generate* directory created before.

4. Select the target technology that is used for implementation. The NIOS Cyclone Board contains an FPGA from the *Cyclone* family of Altera.

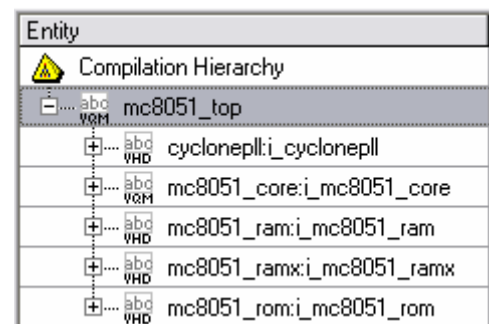
To get a smaller list of the FPGAs, choose speed grade 7.

The device *EP1C20F400C7* can be found in the list shown on the left.



5. After confirming the data given for creating the project file, the hierarchy of the project is displayed on the left.

The top level entity *mc8051_top* should contain the entity *mc8051_core* and all the black boxes added before.



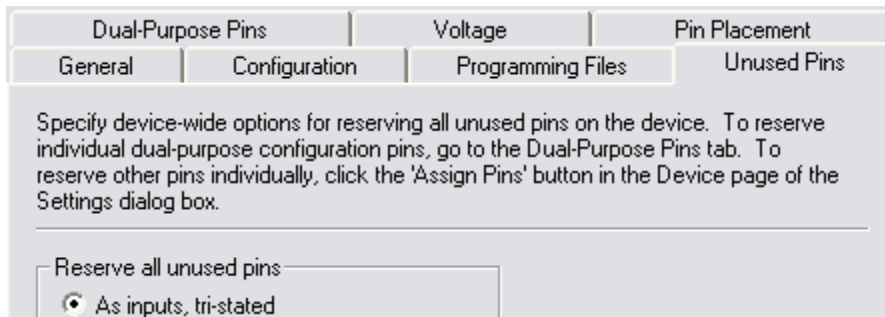
Step 13: Assign the ports of the top level entity.

Before starting the compiler, it is necessary to link the ports of the design with the pins of the FPGA.

1. Choose *Device* form the *Assignments* menu.



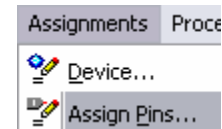
In the next window, select *Device & Pin Options* to define the behaviour of the pins.



Since there are many components located on the NIOS Cyclone Board, it could be a problem if a unused pin drives another device. To prevent that, select the *Unused Pins* folder and choose that all unused pins are treated as tri-stated inputs.

2. Now all ports can be linked to the pins. This can be done manually or by loading the assignments from a TCL script.

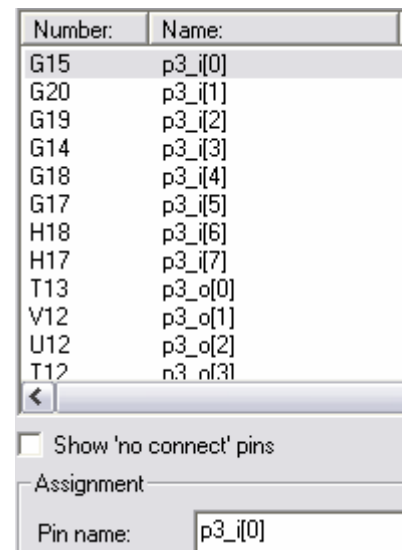
To place the pins manually, select *Assign Pins* in the *Assignments* menu.



A window appears where all pins are listed. To assign a specific pin, scroll to the pin with the desired number, enter the name of the port in the *Pin name* box and click on the *Add* button. Bus signals must be de-referenced with the brackets [].

For example, to assign the signal *p3_i[0]* which stands for the LSB of the input port 3, select the line with pin number G15, enter the signal name as shown on the right and confirm the assignment.

It is recommended to assign all ports since they are linked randomly to a pin if the user does not assign those ports.



To load the pin assignments from a TCL script, activate the *TCL Console* by selecting the appropriate command in the *View – Utility Windows* menu.



As the TCL console appears, type

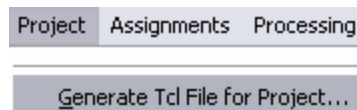
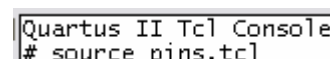
source pins.tcl

To load the assignments from the *pins.tcl* file that is delivered with the MC8051 design.

The file which contains the constraints which are loaded via the TCL console should be located in the Quartus II working directory.

To use the constraints of the current project for other designs, it is possible to export them to a TCL script.

Select *Generate TCL File for Project* in the *Project* menu to generate such a file.



Step 14: Run the compiler and check the errors and the warnings.

To start the compiler, click on the arrow symbol that can be found in the tool bar.



A common mistake is a present underscore character at the end of the name belonging to the top level entity. In that case the following error message appears.



Since the HEX file is smaller than the size of the memory used for the ROM, the remaining bytes are filled with zeros. A warning message as shown below is also forced for that case.



Step 15: Configure the hardware programmer and download the bit stream to the FPGA.

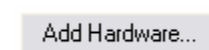
1. The programmer can be configured and used for download by selecting *Programmer* in the *Tools* menu.



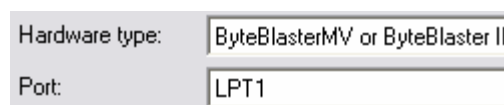
2. As the programmer appears, it is first necessary to do the *Hardware Setup*.



A window appears where the hardware for downloading the bit stream is listed. To set up a new device, hit the *Add Hardware* button.



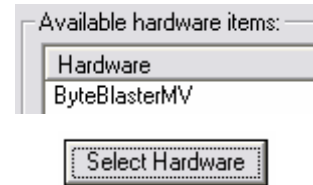
If the parallel port should be used to download a bit stream via the JTAG interface, select *ByteBlaster* as hardware type and *LPT1* as associated port. Confirm this configuration to



get back to the hardware listing.

The defined hardware should now be present in the list.

For activation of that hardware, select the entry and hit the *Select Hardware* button.



- The hardware should now also be present at the top of the programmer window. Set up the NIOS Cyclone Board for operation by connecting the parallel port to the JTAG interface and applying the supply voltage to the board. After that, hit the *Auto Detect* button.



File	Device	Checksum	Usercode	Program/ Configure
1. <none>	EP1C20	00000000	<none>	<input type="checkbox"/>

The programmer should now list the connected device. To add a bit stream file for download, double click on the <none> entry in the *File* section. A window appears where the bit stream can be selected. The SOF file *mc8051_top_.sof* located in the working directory for Quartus II contains the bit stream for the project.

File	Device	Checksum	Usercode	Program/ Configure
1. ...rtus/mc8051_top_.sof	EP1C20F400	004A3543	FFFFFFFF	<input checked="" type="checkbox"/>

The *File* section should now contain the bit stream file. Check the *Program/Configure* flag as shown above.

To start the download, hit the *Start* button. If there is an error during the download, an appropriate error message is shown.



Design tree

mc8051	
c	generation of a ROM-file for simulation
dc	synthesis with Design Compiler
docu	documentation
generate	VHDL code generated by Quartus
msim	simulation with Modelsim
quartus	implementation with Alteras Quartus
synpl	synthesis with Synplicity's Synplify Pro
tb	testbench and memory models
vhdl	synthesizeable VHDL source code
vss	simulation with Synopsys VSS

Literature

Cyclone Device Handbook

http://www.altera.com/literature/hb/cyc/cyclone_device_handbook.pdf

Cyclone Nios Board, Getting Started User Guide

http://www.altera.com/literature/ug/ug_nios_gsg_cyclone_1c20.pdf

Cyclone Nios Board, Reference Manual

http://www.altera.com/literature/manual/mnl_nios_board_cyclone_1c20.pdf