FH University of Applied Sciences TECHNIKUM WIEN

Chip Design 2

mc8051 Getting Started

Version: 1.0
Author: R. Höller, P. Rössler

Copyright Notice

Introduction

This distance learning letter shows how to use the mc8051 IP (Intellectual Property) core which is a VHDL implementation of the popular mc8051 8-bit microcontroller. Here, the mc8051 is used in a simple design (blinking LED, realized by a GPIO pin of the mc8051) which is simulated as well as implemented on a Xilinx Artix-7 FPGA, contained on a Digilent Basys3 board. It is assumed that the reader is already familiar with the languages VHDL and C, the FPGA implementation tool Xilinx Vivado and the ModelSim/QuestaSim simulator which have already been used in previous lectures. This tutorial refers to Xilinx Vivado 2016.1 HL WebPACK and ModelSim-Intel FPGA Starter Edition 10.5b. However, most things will probably apply to other releases of these tools.

Furthermore, the SDCC (Small Device C Compiler) is used for the example to compile a C application which runs on the mc8051 CPU core. You can download release 3.6.0 of SDCC as well as a user guide from the CIS (Campus Information System) website of this lecture[1]. Please install SDCC with the default settings (full installation) to your PC. Don't forget to confirm that the PATH environment variable will be modified at the end of the installation dialogue!

Directory Structure of the mc8051 IP Core Demo Design

Download the file **"mc8051_basys3.zip"** from the CIS webpage of this lecture and unzip it to

```
d:\work\
```

Figure 1 depicts the directory structure of the mc8051 IP core demo design.

[1] ports of SDCC to platforms other than MS Windows/64 Bit can be downloaded from http://sdcc.sourceforge.net/

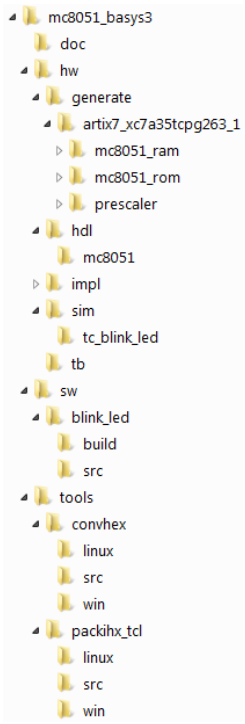

Figure 1: Directory structure of the mc8051 IP core demo design

The meaning of the various directories of the demo design is as follows:

- **doc** … documentation (user guides, datasheets etc.)
  - **hw** … all design data related to hardware
  - **generate** … generated hardware blocks (RAM, ROM, PLL …)
    - **artix7_xc7a35tcpg236_1** … generated hardware blocks for xc7a35tcpg236-1 FPGA
  - **hdl** … HDL code (VHDL)
    - **mc8051** … HDL code of the mc8051 IP core
  - **impl** … implementation directory (Xilinx Vivado project directory)
  - **sim** … working directory of ModelSim simulator
    - **tc_blink_led** … test case “Blink LED”
  - **tb** … testbenches
- **sw** … all design data related to software
  - **blink_led** … "Blink LED" software application, running on the mc8051
    - **build** ... SDCC compiler working directory
    - **src** ... C source code of the mc8051 "Blink LED" example
- **tools** … misc. tools
  - **convhex** ... convhex tool (referenced by the ModelSim do-files)
    - **src** ... C source code of convhex tool
    - **win** ... Windows executable of convhex tool
    - **linux** ... Linux executable of convhex tool
  - **packihx_tcl** ... packihx tool, TCL version (referenced by the ModelSim do-files)
    - **src** ... C source code of packihx tool
    - **win** ... Windows executable of packihx tool

- **linux** ... Linux executable of packihx tool

A similar directory structure is recommended for your own projects!

Simulation of the mc8051 IP core demo design

Typically, whenever FPGA-internal building blocks and resources (PLLs, on-chip memories) are used in a design, dedicated libraries are needed to simulate these technology-dependent blocks. Since the mc8051 demo design incorporates such FPGA-internal resources, you need to install the Xilinx technology libraries prior to any simulation.

Start the ModelSim simulator. You can see the Xilinx technology libraries in the “Library” window of ModelSim, as shown in Figure 2. If you don’t see them you have to install the libraries as described in the distance learning letter “Installing Xilinx Simulation Libraries” which can be found on the CIS website of this lecture.

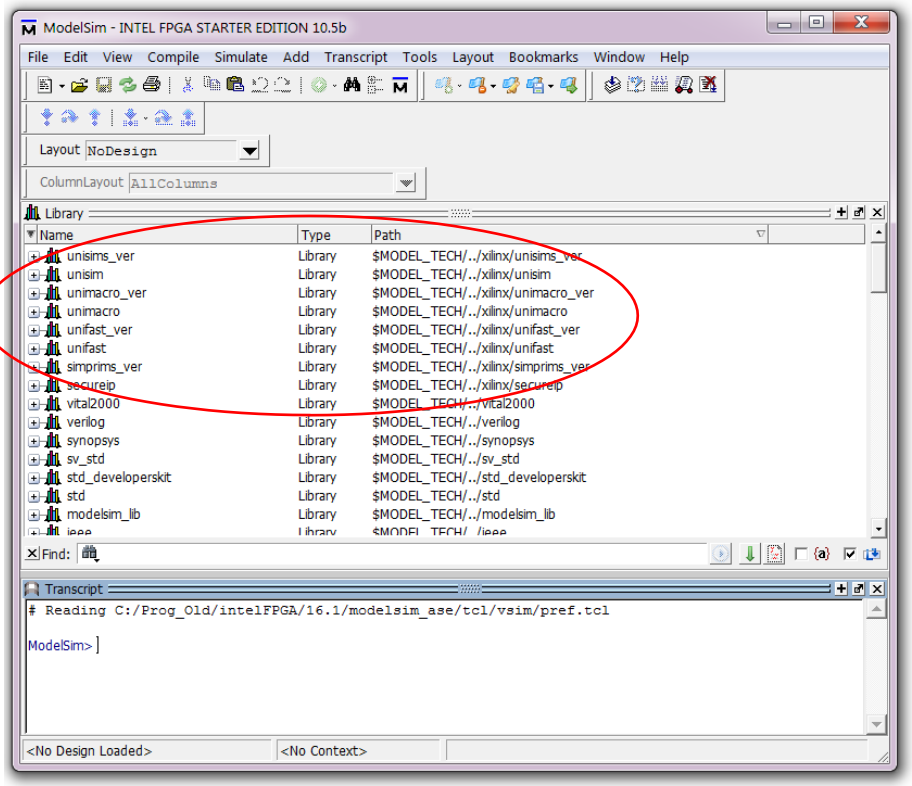

Figure 2: ModelSim with Xilinx Simulation Libraries installed

In the ModelSim console enter

```
cd d:/work/mc8051_basys3/hw/sim/tc_blink_led
```

to change to the simulation directory of the demo design. You can verify the current path by entering

```
pwd
```

Next, enter

```
dir
```

to view the contents of the directory. As you can see, there are some do-scripts for automation of the hardware/software design flow. We will use them in the following.

If you simulate the design for the first time, generate a work library by entering

```
vlib work
```

By running the “compile” script

```
do compile.do
```

all VHDL/Verilog files that are needed to simulate the mc8051 demo design will be compiled.

The next script must be executed whenever you change the software application running on the mc8051. The script

```
do build_mc8051_sw.do
```[2]

compiles the C source file

```
d:/work/mc8051_basys3/sw/blink_led/src/main.c
```

using the “Small Device C Compiler” (SDCC) and converts the executable into a MIF (Memory Initialization File) which defines the content of the mc8051 program memory. However, as you might have noticed, a default MIF file is already prepared in simulation directory. Therefore, you you don’t have to run the “build_mc8051_sw.do” script as long as you do not change the mc8051 software application.

[2] on Linux-based systems use the script „build_mc8051_sw_linux.do“

The next script

```
do sim.do
```

loads the compiled design from the “work” library and starts the simulation for 150 us (which takes about 10 seconds, depending on the performance of your PC). The simulation results can be observed in the ModelSim “Wave” window. You can see the toggling signal “led_o(0)”, see Figure 3).

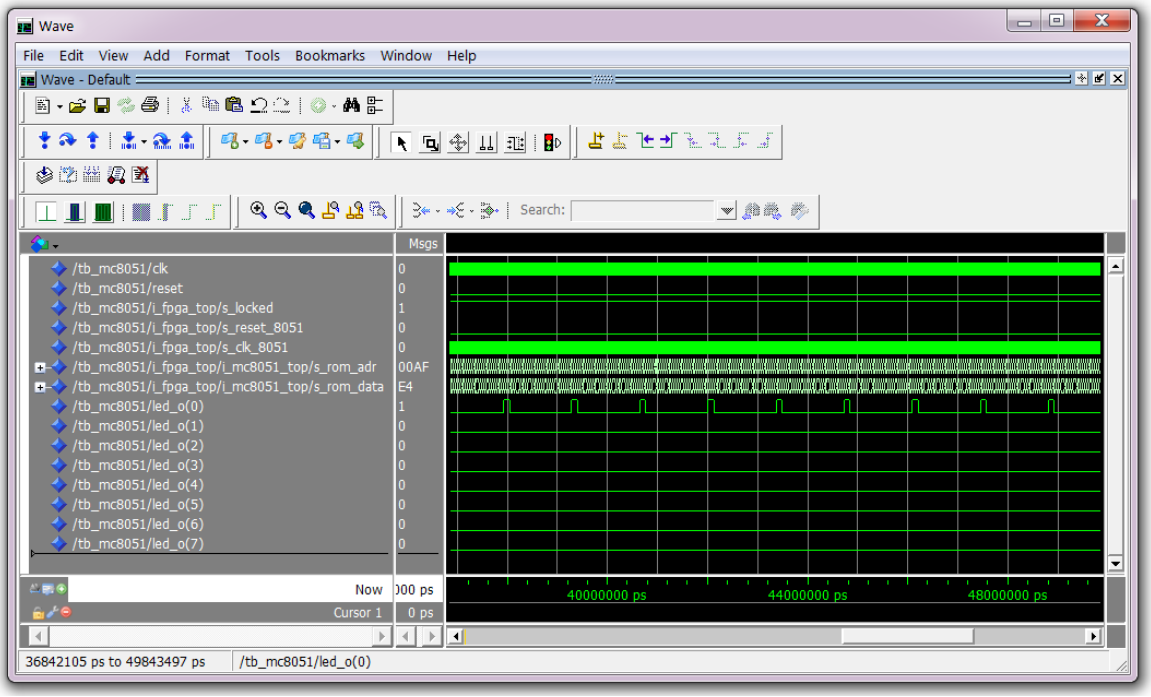

Figure 3: Simulation results of the mc8051 demo design

In the following, the three do-scripts that have been used previously are described in more detail.

compile.do:

Simulation models of generated RAM, ROM and PLL

```
vlog ../../generate/artix7_xc7a35tcpg263_1/prescaler/prescaler/prescaler_clk_wiz.v
vlog ../../generate/artix7_xc7a35tcpg263_1/prescaler/prescaler/prescaler.v
vlog ../../generate/artix7_xc7a35tcpg263_1/mc8051_rom/mc8051_rom/blk_mem_gen_v8_3_2/
  simulation/blk_mem_gen_v8_3.v
vcom ../../generate/artix7_xc7a35tcpg263_1/mc8051_rom/mc8051_rom/synth/mc8051_rom.vhd
vcom ../../generate/artix7_xc7a35tcpg263_1/mc8051_ram/mc8051_ram/synth/mc8051_ram.vhd
```

The vlog command compiles Verilog source code into the work library similar as it's VHDL counterpart vcom

```
vlog ../../generate/glbl.v

vcom ../../hdl/mc8051/mc8051_p.vhd

vcom ../../hdl/mc8051/control_mem_.vhd
vcom ../../hdl/mc8051/control_mem_rtl.vhd
vcom ../../hdl/mc8051/control_mem_rtl_cfg.vhd

vcom ../../hdl/mc8051/control_fsm_.vhd
vcom ../../hdl/mc8051/control_fsm_rtl.vhd
vcom ../../hdl/mc8051/control_fsm_rtl_cfg.vhd

                :         :         :
                :         :         :

vcom ../../hdl/fpga_top_.vhd
vcom ../../hdl/fpga_top_rtl.vhd
vcom ../../hdl/fpga_top_rtl_cfg.vhd

vcom ../../tb/tb_mc8051_.vhd
vcom ../../tb/tb_mc8051_sim.vhd
vcom ../../tb/tb_mc8051_sim_cfg.vhd
```

In order to properly reset the design in a simulation, the "glbl.v" module must be compiled and loaded along with the design

VHDL code of mc8051 IP core. The compile order is important!

Top-level unit of mc8051 demo design

Testbench of mc8051 demo design (is always compiled at last)

sim.do:

Maintains compatibility between ModelSim and QuestaSim

Desired time resolution (optional)

Specifies the library to search for design units instantiated from Verilog and for VHDL default component binding

Specifies the default working library where vsim will look for the design unit(s)

The vsim command invokes the simulator

```
vsim  -novopt -t ps -L unisims_ver -lib work work.tb_mc8051
work.glbl view *
do mc8051_wave.do
run 150 us
```

Starts the simulation and runs it for 150 us

Loads a predefined file which contains a list of signals that will be displayed in the "wave" window

build_mc8051_sw.do:

```
# compile software application for mc8051 using SDCC compiler
# and generate Intel HEX file using packihx_tcl.exe
sdcc ../../../sw/blink_led/src/main.c -o ../../../sw/blink_led/build/
echo "sdcc: main.c sucessfully compiled"
../../../tools/packihx_tcl/win/packihx_tcl.exe ../../../sw/blink_led/build/main.ihx
   ../../../sw/blink_led/build/mc8051_rom.hex

# generate MIF and COE files out of Intel HEX file
../../../tools/convhex/win/convhex.exe ../../../sw/blink_led/build/mc8051_rom.hex

# copy MIF file (required for simulation only) to ModelSim simulation directory
file copy -force ../../../sw/blink_led/build/mc8051_rom.mif
   ../../../hw/sim/tc_blink_led/

echo "mc8051 software application built successfully!"
```

Implementation of the mc8051 IP core demo design

In order to implement the mc8051 IP core demo design on the Artix-7 FPGA contained on the Digilent Basys3 board, simply double-click the file

`D:/work/mc8051_basys3/hw/impl/mc8051.xpr`

in Windows Explorer. Generate the bitstream (this takes about 5 minutes, depending on the performance of your PC) and download it to the FPGA. After downloading the design to the FPGA you will see LED0 blinking. You will notice a lot of warnings in Xilinx Vivado but they all can be ignored since most of them they do not indicate any flaws in the design or the mc8051 IP core.

If you would like to change the hardware of the mc8051 IP core demo design, a new synthesis and implementation run must be performed using Xilinx Vivado. Moreover, if you change the mc8051 application program, the design must also be re-synthesized and re-implemented. This is necessary, because the content of the mc8051 program memory will change. You have to execute the `build_mc8051_sw.do` script in advance. **It is also important that you close Xilinx Vivado before you execute the `build_mc8051_sw.do` script and re-open the Vivado project after the do-script was run to instruct Vivado to re-synthesize the mc8051 program memory! Otherwise Vivado does not notice that the content of the mc8051 program memory has changed!**

Abbreviations

FPGA	Field Programmable Gate Array
EDA	Electronic Design Automation
HDL	Hardware Description Language
IP	Intellectual Property
SDCC	Small Device C Compiler
Tcl/Tk	Tool command language / Tool kit
VHDL	Very High Speed Integrated Circuit Hardware Description Language

Questions

1) As shown above by the mc8051 IP core demo design, three essential steps are necessary to simulate a VHDL design. Name these steps!
2) What is the correct compile order for VHDL files?
3) Why is it useful to setup a clean directory structure for VHDL designs and related EDA tools?
4) Estimate the ratio between EDA tool knowledge and basic knowledge about digital design (VHDL, electronic circuits …)?
5) Which files have to be compiled first in a mixed-language (Verilog and VHDL) simulation?

Version

Version 1.0	Update to entire document for CHIP2

If you find errors or inconsistencies please report them to the supervisors of this course. Thank you!