1 Instructions to Run the Solvers on DE1-SoC Board

This section contains instructions to install necessary software, setup board connectivity, and to compile and run the different solver applications on the DE1-SoC board. The following instructions should guide to compile and run the source code written during the thesis time-frame.

1.1 Hardware Required

1. DE1-SoC development board.
2. Micro SD card (at least 4 GB).
3. Micro USB cable.
4. Ethernet cable.
5. Windows computer with at least 8GB RAM and 20GB disk space.

1.2 Software Installations

The following software are required for a successful development environment setup on the Windows operating system (Tested on Windows 10).

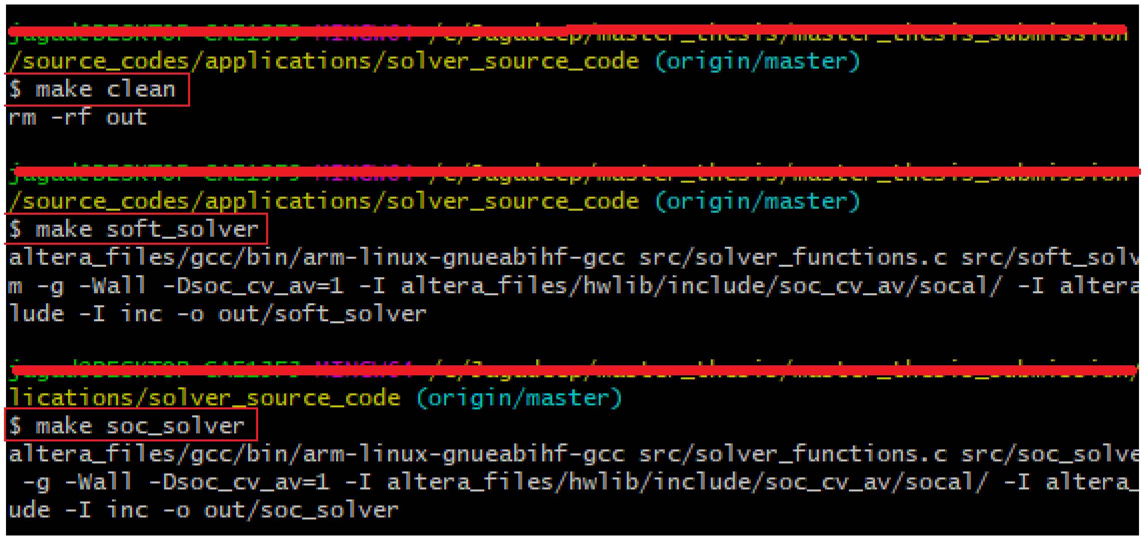
1. Quartus Prime Software: Download it from Intel's homepage and install it to design, synthesis, and program FPGA. The thesis project files are based on version 18.1 and it is recommended to install the same, but it might also work for the higher versions by upgrading the IP tools. The source code is designed using Lite edition which is available for a free of cost and does not require a license. One might choose any edition based on their needs (with subscription or license) and the project files are expected to work regardless of the editions. Download Cyclone V device support files to the same folder as the Quartus installation file in the host machine before installation.
2. Python: Install Python3 software, Matplotlib and Numpy packages. The packages can be installed using PIP tool (*$pip install matplotlib* and *$pip install numpy*) after installing Python3. They are required to verify the result files and plot timing graphs.
3. Putty: Install a terminal software to read/write a serial port. It is used as an command prompt interface towards the target Linux.
4. MinGW: Install Linux based GNU software on Windows OS to use Make tool for building applications. Remember to set its binary directory in the user path under environment variables.
5. Thesis report: Refer the master thesis report for design and implementation details.
   www.diva-portal.org/smash/record.jsf?pid=diva2%3A1340181dswid=7444

1.3 Compilation of MCU software

In the repository, navigate to the "applications/solver_source_code" folder in a terminal and execute the following commands.

- Clean application files using "make clean" command.
- Compile by typing "make" followed by an application name to compile a specific application and without an argument to compile all the three applications as shown in figure 1.

The ARM cross compiler is included in the "altera_files/gcc" folder which is used by the build system and necessary header files under "altera_files/hwlib" for successful compilation. It is highly recommended not to change these files and folders.



```
[redacted]
/source_codes/applications/solver_source_code (origin/master)
$ make clean
rm -rf out

[redacted]
/source_codes/applications/solver_source_code (origin/master)
$ make soft_solver
altera_files/gcc/bin/arm-linux-gnueabihf-gcc src/solver_functions.c src/soft_solv
m -g -Wall -Dsoc_cv_av=1 -I altera_files/hwlib/include/soc_cv_av/socal/ -I altera
lude -I inc -o out/soft_solver

[redacted]
lications/solver_source_code (origin/master)
$ make soc_solver
altera_files/gcc/bin/arm-linux-gnueabihf-gcc src/solver_functions.c src/soc_solve
 -g -Wall -Dsoc_cv_av=1 -I altera_files/hwlib/include/soc_cv_av/socal/ -I altera_
ude -I inc -o out/soc_solver
```

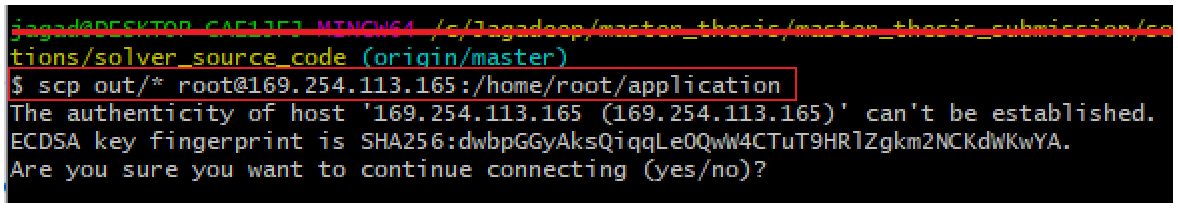
Figure 1: Make commands

1.4 Setting up Connectivity and Files Transfer

Download board manual, user guide, and all attachments from the home page of the DE1-SoC ?. Go through the user manual and set the dip switches for terminal-based Linux. Connect the USB terminal port from the board to the host computer and start the Putty software. Look for the comport number in device manager tool and open it as a serial port with baud rate 115200. Prepare an SD card containing Linux image as mentioned in the user manual and insert it before turning ON the board. Wait for 5 seconds during the Linux start-up for auto-boot process to complete and enter the login name **root**. This takes to the home directory of the root login.

Connect the host computer and the board through an Ethernet cable. Find the IP address of the host and set the IP address of the board with plus one in the last digit to that of the host's. For example, if the host has 169.254.113.164, then set the target as 169.254.113.165 using the command "**ifconfig eth0 169.254.113.165**". To confirm the connectivity, ping the address from the host.

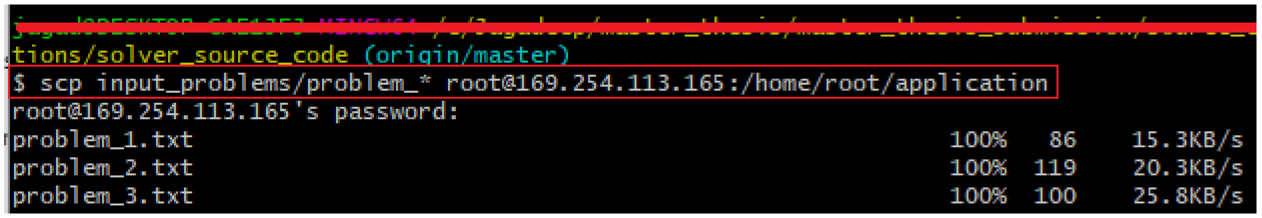
The application binaries created in the folder ”out” can be transferred to the target using SCP command as mentioned in figure 2. Create a folder called ”application” under the home directory in the target before the file transfer.



```
[redacted]
tions/solver_source_code (origin/master)
$ scp out/* root@169.254.113.165:/home/root/application
The authenticity of host '169.254.113.165 (169.254.113.165)' can't be established.
ECDSA key fingerprint is SHA256:dwbpGGyAksQiqqLeOQwW4CTuT9HRlZgkm2NCKdWKwYA.
Are you sure you want to continue connecting (yes/no)?
```

Figure 2: Command to transfer binary files to the target

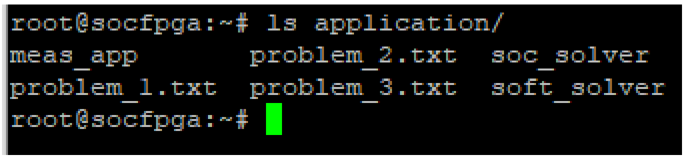
Type ”yes” to a security question and enter the password **terasic** (or find it in the board user manual) to complete the file transfer. Also transfer the input problem files in the same way.



```
[redacted]
tions/solver_source_code (origin/master)
$ scp input_problems/problem_* root@169.254.113.165:/home/root/application
root@169.254.113.165's password:
problem_1.txt                                                    100%   86    15.3KB/s
problem_2.txt                                                    100%  119    20.3KB/s
problem_3.txt                                                    100%  100    25.8KB/s
```

Figure 3: Command to transfer input files to the target

Now the ”application” folder in the target contains the three applications and the three input files.



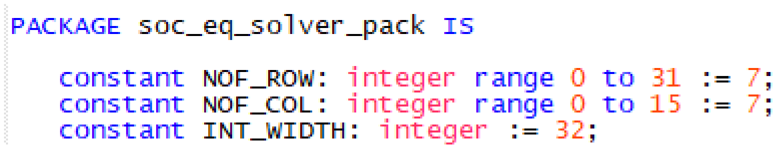
```
root@socfpga:~# ls application/
meas_app       problem_2.txt  soc_solver
problem_1.txt  problem_3.txt  soft_solver
root@socfpga:~#
```

Figure 4: List of files transferred to the target

1.5 Synthesis and Program FPGA designs

Find the folders ”soc_eq_solver_parallel” and ”soc_eq_solver_serial” under the repository root for the parallel and the serial solver’s design files. Open them in the Quartus IDE by double-clicking the ”soc_eq_solver.qpf” file under these folders. The folder “vhdl_design_files” contains VHDL design files that are common to the two projects and the individual design files are within the respective project folders. All the design and project files are submitted in the working condition without compilation and run-time errors.

The design properties can be changed in a VHDL package file before synthesis and compilation as in figure 5.



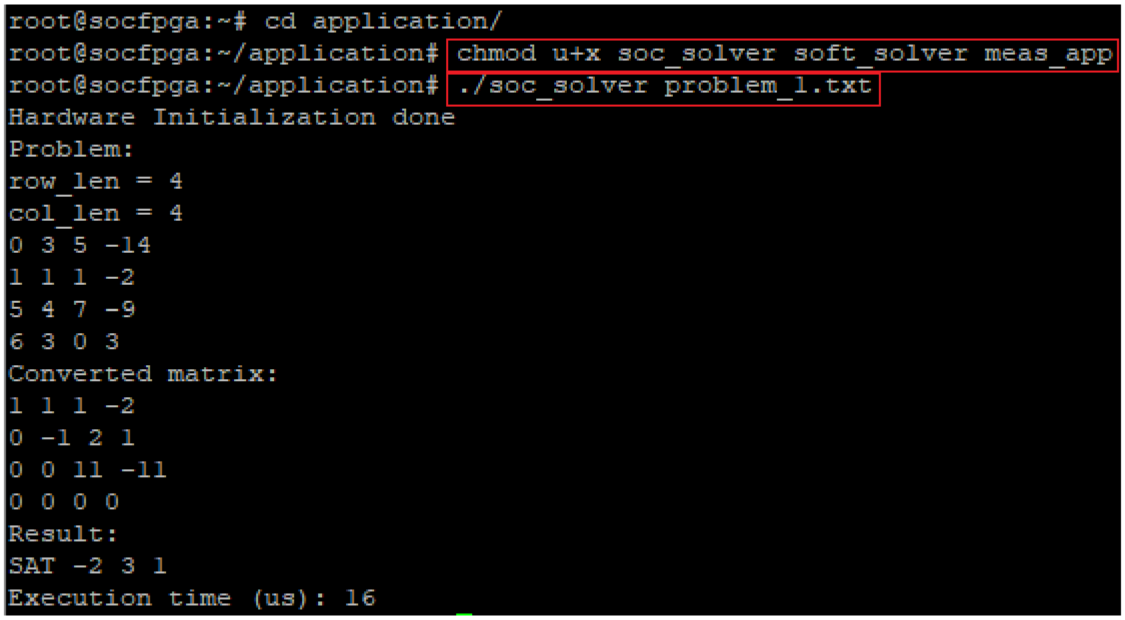
```
PACKAGE soc_eq_solver_pack IS

   constant NOF_ROW: integer range 0 to 31 := 7;
   constant NOF_COL: integer range 0 to 15 := 7;
   constant INT_WIDTH: integer := 32;
```

Figure 5: Hardware design properties in soc_eq_solver_pack.vhd

The package file is common for both the designs and the parameters should be changed before synthesis if they are different. Follow the Quartus user guide to modify, compile, and program the hardware converter design before running the applications.

1.6 Running Solver Applications

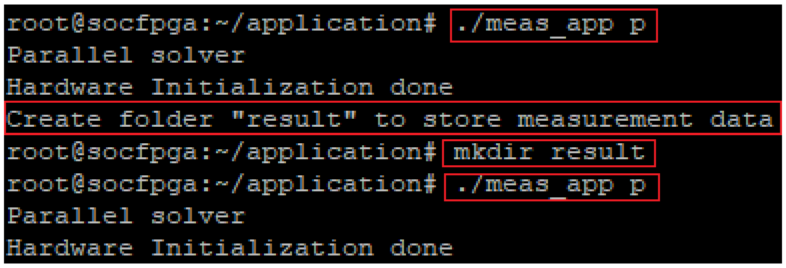
Change access permission of the application files to make them executable and execute the "soc_solver" (after programming FPGA) and the "soft_solver" followed by the name of an input text file as shown in figure 6.



```
root@socfpga:~# cd application/
root@socfpga:~/application# chmod u+x soc_solver soft_solver meas_app
root@socfpga:~/application# ./soc_solver problem_1.txt
Hardware Initialization done
Problem:
row_len = 4
col_len = 4
0 3 5 -14
1 1 1 -2
5 4 7 -9
6 3 0 3
Converted matrix:
1 1 1 -2
0 -1 2 1
0 0 11 -11
0 0 0 0
Result:
SAT -2 3 1
Execution time (us): 16
```

Figure 6: Run solver applications with an input text file

To execute the measurement application, type "meas_app" followed by a character 'p' or 's' based on the type of converter programmed. It is shown in the first LED on the target board. The application would throw an error saying that the "result" folder should be created. Re-run the application after creating the folder for a successful execution. It is designed on purpose that a user should be aware of results stored in this folder. Run this application again with other hardware converter to store all output files under the "result" folder. Note that the last two LEDs will display combinations of values which indicates row and column lengths of a matrix transferred for processing.

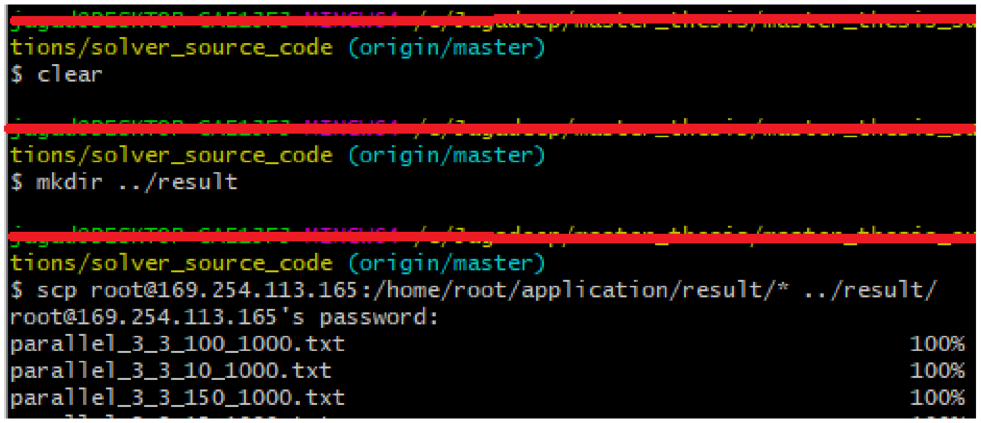


```
root@socfpga:~/application# ./meas_app p
Parallel solver
Hardware Initialization done
Create folder "result" to store measurement data
root@socfpga:~/application# mkdir result
root@socfpga:~/application# ./meas_app p
Parallel solver
Hardware Initialization done
```

Figure 7: Run measurement application

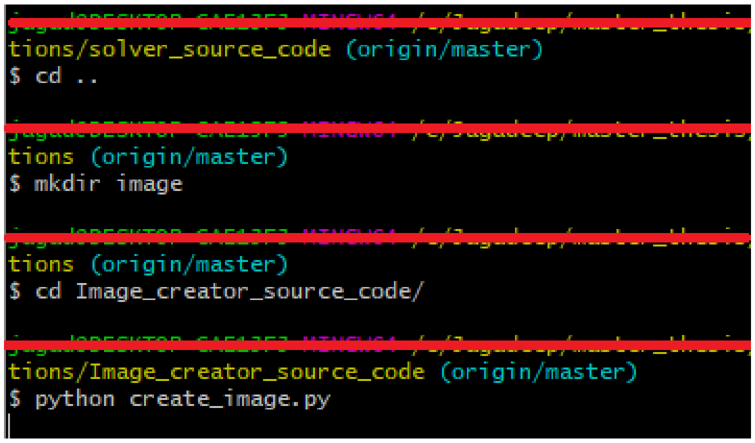
1.7 Analyzing Result Files

Transfer the ”result” folder from the target to the host machine to analyze and compare execution time between different solvers. Create a folder called ”result” under the ”applications” folder and execute the following SCP command to transfer all output files (as in figure 8). The name and location of the ”result” folder should not be changed as it is required by the ”image_create.py” script to read them. Run all the commands and the python script as shown in figure 9 to store graphs under the ”image” folder.



```
[redacted]
tions/solver_source_code (origin/master)
$ clear

[redacted]
tions/solver_source_code (origin/master)
$ mkdir ../result

[redacted]
tions/solver_source_code (origin/master)
$ scp root@169.254.113.165:/home/root/application/result/* ../result/
root@169.254.113.165's password:
parallel_3_3_100_1000.txt                                        100%
parallel_3_3_10_1000.txt                                         100%
parallel_3_3_150_1000.txt                                        100%
```

Figure 8: Transfer results from the target to the host



```
[redacted]
tions/solver_source_code (origin/master)
$ cd ..

[redacted]
tions (origin/master)
$ mkdir image

[redacted]
tions (origin/master)
$ cd Image_creator_source_code/

[redacted]
tions/Image_creator_source_code (origin/master)
$ python create_image.py
```

Figure 9: Create timing graphs