NC State University
ECE Department ECE 720 W. Rhett Davis

Homework #1 Solution

This solution begins by cloning the dcrm and icc2rm repositories as described in the lecture. Next, a Makefile was created to execute the solutions to all three problems, given below:

Makefile

```
.PHONY: synth


pnr: synth
	$(MAKE) -C icc2rm

synth:
	$(MAKE) -C dcrm

clean:
	cd dcrm && $(MAKE) clean
	cd icc2rm && $(MAKE) clean

qor:
	find . -type f -exec grep -q report_qor {} \; -print | wc

times:
	python difftime.py dcrm/synth.begin dcrm/synth
	python difftime.py icc2rm/setup icc2rm/route_opt
```

Note that problem 1 is solved by the default target "pnr", while problems 2 and 3 are solved by the targets "qor" and "times", respectively. A "clean" target is also added, for convenience. Note that the "times" target uses a python script to compare the times of the date-files generated by ICC2-RM. Next, a Makefile is created in the dcrm subdirectory, as shown below.

dcrm/Makefile

```
synth:
	date > $(@).begin
	dc_shell -topographical_mode -f rm_dc_scripts/dc.tcl |& tee dc.log
	date > $@


gui:
	dc_shell -topographical_mode -gui -f
rm_dc_scripts/dc_interactive.tcl

clean:
	-rm synth synth.begin
	-rm -rf WORK
	-rm -rf *_LIB
	-rm -rf reports
	-rm -rf results
	-rm *.log
	-rm *.svf
```

Note that dcrm/Makefile creates the date-files needed by the first Makefile. It also adds the clean target and a “gui” target for convenience. Next the Makefile is created for the icc2rm subdirectory:

icc2rm/Makefile

```
DESIGN = counter

pnr:
	$(MAKE) -f rm_setup/Makefile_pnr route_opt

clean:
	-rm *.log
	-rm *.ems
	-rm icc2_output.txt
	-rm setup init_design place_opt clock_opt_cts clock_opt_opto 
route_auto route_opt chip_finish icv_in_design write_data all 
redhawk_in_design_pnr pt_eco pt_eco_incremental_1 pt_eco_incremental_2 fm 
vc_lp summary
	-rm
	-rm -rf legalizer_debug_plots/
	-rm -rf logs_icc2
	-rm -rf rpts_icc2
	-rm -rf $(DESIGN)
```

The default “pnr” target executes the Makefile in the ICC2-RM through to the route_opt step, which is all that was required. The clean target removes all of the files that are generated by the “all” target in icc2rm/rm_setup/Makefile_pnr, just in case we want to run all of the steps later. Next, we return to the root directory and create the python script to compare the file-times.

difftime.py

```
import sys, os

t1=os.path.getmtime(sys.argv[1])
print('Start time in',sys.argv[1])
t2=os.path.getmtime(sys.argv[2])
print('End time in',sys.argv[2])
print('Duration',t2-t1,'seconds')
```

Finally, it can be confusing to remember how to setup the environment to execute the flow. For convenience, I like to create two setup files called “addsetup.sh” and “varsetup.sh”. The first “addsetup.sh” script calls the “add” command in the NCSU Linux environment, which starts a fresh shell with the specified tools added to the environment. The second “varsetup.sh” script adds all of the necessary environment variables (in this case, the variables needed to run Python3). These scripts can’t be combined on our system, because the “add” command always starts a new shell.

addsetup.sh

```
add synopsys2019
```

varsetup.sh

```
source /afs/eos.ncsu.edu/lockers/research/ece/wdavis/tools/anaconda3/setup.sh
```

Once all of these files are created, the solution to the homework is generated with the following steps:

```
$ source addsetup.sh
$ source varsetup.sh
$ make
$ make qor
$ make times
$ make clean
$ make qor
```

The first execution of "make" solves problem 1, while problems 2 and 3 are solved by the next two executions of make. Just in case some students execute problem 2 before problem 1, the directories are cleaned, and the qor target is executed again. "make qor" gives 51 when running before the reference methodologies, or 61 when running after the reference methodologies.