ECE 5745 Linux, Git, PyMTL, Verilog Cheat Sheet

Linux Commands

`# comment`	comment, does nothing
`man command`	display help for given `command`
`echo "string"`	display given `string`

`echo "string" > file`	create `file`
`cat a`	display file `a`
`less a`	display file `a` with paging and search
`ls`	list contents of current working dir
`ls -la`	list contents of current working dir (verbose)
`ls A`	list contents of dir `A`
`ls *.txt`	list files with `.txt` suffix in current working dir

`pwd`	display current working dir
`mkdir A`	make dir `A` to `B`
`mkdir -p A/B`	make all dirs in path `A/B`
`cd A`	change current working dir to `A`
`cd ..`	change current working dir to parent dir
`cd ~`	change current working dir to home dir
`tree`	recursively list contents of current working dir

`cp a b`	copy file `a` to `b`
`cp -r A B`	copy dir `A` to `B`
`mv a b`	move file `a` to `b`
`mv A B`	move dir `A` to `B`
`rm a`	remove file `a`
`rm -r A`	remove dir `A`

`wget url`	download file at `url`
`grep "string" a`	search file `a` for given `string`
`grep -r "string" A`	recursively search files in dir `A`
`find . -name "string"`	find files named `string` in dir `.`
`tar -czvf a.tgz A`	create archive `a.tgz` of dir `A`
`tar -xzvf a.tgz`	extract archive `a.tgz`
`top`	view what is running on system

`ENVVAR="string"`	set environment variable
`echo ${ENVVAR}`	display given environment variable
`cmd > a`	redirect output of `cmd` to newly created file `a`
`cmd >> a`	redirect output of `cmd` to append to file `a`
`cmd_a && cmd_b`	execute `cmd_a` and then execute `cmd_b`
`cmd_a | cmd_b`	send output from `cmd_a` to `cmd_b`

`source setup-ece2400.sh`	source setup script for course
`quota`	check disk usage
`trash`	move file to `${HOME}/tmp/trash`

Git Commands

`help cmd`	display help on git command `cmd`
`clone url`	clone repo at given URL
`add a`	add file `a` to index
`add A`	add directory `A` to index
`add -u`	add all tracked files
`commit`	commit indexed files
`commit -m "msg"`	commit indexedfiles w/ commit `msg`
`log`	show history log of previous commits
`status`	show status of local repo
`checkout a`	revert file `a` to last commit
`checkout A`	revert dir `A` to last commit
`pull`	pull remote commits to local repository
`push`	push local commits to remote repository
`whatchanged`	show incremental changes for each commit

`xstatus`	compact status display
`xlog`	compact log display
`xadd`	add all tracked, modified files to index
`xpull`	short for `pull --rebase`

Example RTL Development Session

```
% source setup-ece5745.sh
% cd ${HOME}/ece5745
% git clone \
   git@github.com:cornell-ece5745/ece5745-tut3-pymtl

% mkdir -p ece5745-tut3-pymtl/sim/build
% cd ece5745-tut3-pymtl/sim/build
% pytest ../tut3_pymtl/gcd
% pytest ../tut3_pymtl/gcd --verbose
% pytest ../tut3_pymtl/gcd -x -s --tb=long

% pytest ../tut3_pymtl/gcd/test/GcdUnitRTL_test.py
% pytest ../tut3_pymtl/gcd/test/GcdUnitRTL_test.py \
   -k basic_0x0 --dump-vcd
% gtkwave *.vcd &

% ../tut3_pymtl/gcd/gcd-sim --help
% ../tut3_pymtl/gcd/gcd-sim --impl rtl \
    --input random --stats --trace --dump-vcd
% gtkwave gcd-rtl-random.vcd &
```

GitHub and TravisCI URLs

```
http://github.com/cornell-ece5745/labY-groupXX
http://travis-ci.com/cornell-ece5745/labY-groupXX
```

Example ASIC Development Session

```
% source setup-ece5745.sh
% mkdir $HOME/ece5745
% cd $HOME/ece5745
% git clone \
git@github.com:cornell-ece5745/ece5745-tut6-asic-flow
% cd ece5745-tut6-asic-flow
% TOPDIR=$PWD

% mkdir -p $TOPDIR/sim/build
% cd $TOPDIR/sim/build
% pytest ../tut3_pymtl/gcd
% ../tut3_pymtl/gcd/gcd-sim \
   --impl rtl --translate --dump-vcd
% cat GcdUnitRTL_noparam__pickled.v

% mkdir -p $TOPDIR/asic/build
% cd $TOPDIR/asic/build
% pyhflow help
% pyhflow configure ../flow_tut6_gcd.py
% pyhflow list

% pyhflow run synth
% pyhflow run pnr
% pyhflow run pwr
% pyhflow run summary

design_name   = GcdUnitRTL_noparam
design_area   = 494.494 um^2
stdcells_area = 494.494 um^2
macros_area   = 0.0 um^2
chip_area     = 1240.198 um^2
core_area     = 727.776 um^2
constraint    = 1.0 ns
slack         = 0.186 ns
actual_clk    = 0.814 ns
exec_time     = 8684 cycles
power         = 0.4769 mW
energy        = 4.13966 nJ
```

ECE 5745 Linux, Git, PyMTL, Verilog Cheat Sheet

RegIncr.py

```
#==================================
# Registered Incrementer
#==================================

from pymtl3 import *

class RegIncr( Component ):

  def construct( s ):

    s.in_ = InPort(8)
    s.out = OutPort(8)

    # Sequential logic

    s.reg_out = Wire(8)

    @update_ff
    def block1():
      if s.reset:
        s.reg_out <<= 0
      else:
        s.reg_out <<= s.in_

    # Combinational logic

    @update
    def block2():
      s.out @= s.reg_out + 1
```

Example Line Trace for GCD

```
    src      in     A  B  ST out  sink
 1: 0f:05 > 0f:05(xx xx I )   >
 2: #     > #    (0f 05 C-)   >
 3: #     > #    (0a 05 C-)   >
 9: #     > #    (05 05 C-)   >
10: #     > #    (00 05 Cs)   >
11: #     > #    (05 00 C )   >
12: #     > #    (05 00 D )05 > 05
```

Legend for val/rdy interfaces in line traces

spaces	`!valid && ready`
`#`	`valid && !ready`
`.`	`!valid && !ready`
`msg`	`valid && ready`

RegIncr.v

```
//==================================
// Registered Incrementer
//==================================

`ifndef REGINCR_REG_INCR_V
`define REGINCR_REG_INCR_V

module tut4_verilog_regincr_RegIncr
(
  input  logic       clk,
  input  logic       reset,
  input  logic [7:0] in_,
  output logic [7:0] out
);

  // Sequential logic

  logic [7:0] reg_out;
  always_ff @( posedge clk ) begin
    if ( reset )
      reg_out <= 0;
    else
      reg_out <= in_;
  end

  // Combinational logic

  logic [7:0] temp_wire;
  always_comb begin
    temp_wire = reg_out + 1;
  end

  // Combinational logic

  assign out = temp_wire;

endmodule

`endif /* REGINCR_REG_INCR_V */
```

Valid PyMTL3 in Concurrent Blocks

```
Bits1Bits2
& | ^ ~
+ - *
<< >> == != <= < >=
reduce_and() reduce_or()
reduce_xor()
sext() zext() concat()
if else elif
s.signal[n] s.signal[n:m]
```

Coding Conventions

- Try to keep lines less than 74–80 chars
- Include header comment at top of each file
- Include comments to explain code
- Use only spaces, no tabs; use two-space indentation
- Use `CamelCase` for component/module names
- Use `under_scores` for var, port, instance names
- Use `clk` and `reset` for clock and reset
- Use informative variable names
- Separate sequential from combinational logic
- FSMs have 3 blocks: state, transitions, outputs
- Datapaths use structural composition

PyMTL3 Specific

- Use s instead of `self`
- All wires and modules must be members of parent component (i.e., `s.wire`, `s.component`)
- Only use `<<=` in `update_ff` blocks
- Only use `@=` in `update` blocks
- Only signals (e.g., `InPort`, `OutPort`, `Wire`) can be used to communicate between concurrent blocks
- Do not use `or`, `and`, `not` in boolean logic equations

Verilog Specific

- Use subdir prefix for macro and module names
- Explicitly include file dependencies
- Use include guards
- Use `ALL_CAPS` for macro names
- Use `p_` prefix for parameters
- Use `c_` prefix for constants
- Use localparams for local constants
- Always declare type of all ports
- Align port declarations, one per line
- Align port connections, one per line
- Align parameter config, one per line
- Avoid positional port connections & param config
- Use `always_ff` and `always_comb`
- Only use `<=` in `always_ff`, `=` in `always_comb`
- Prefer ternary operator over `if`
- Do not use `casex`, use `casez` very carefully
- Use tasks for compact state output table

Synthesizable Verilog Keywords

`logic`
`logic [N-1:0]`
`& | ^ ^~ ~` (bitwise)
`&& || !`
`& ~& | ~| ^ ^~` (reduction)
`+ -`
`>> << >>>`
`== != > <= < <=`
`{}`
`{N{}}` (repeat)
`?:`
`always_ff, always_comb`
`if else`
`case, endcase`
`begin, end`
`module, endmodule`
`input, output`
`assign`
`parameter`
`localparam`
`genvar`
`generate, endgenerate`
`generate for`
`generate if else`
`generate case`
named port connections
named parameter passing

Synthesizable Verilog Keywords with Limitations

`always`
`enum`
`struct`
`casez, endcase`
`task, endtask`
`function, endfunction`
`=` (blocking assignment)
`<=` (non-blocking assignment)
`typedef`
`packed`
`$clog2() $bits() $signed()`
`* for`