Project 1 [ECE 720]

Digvijay Anand
200478940
[danand3@ncsu.edu]

Project Requirements:

(run '*maxr_lemt*' design for optimal metric **" A^2 * D ")**

1. [sufficient] viol_tot: **viol_tot < 10** [route_opt__drc__viol__tot]

2. [sufficient] hold slack: **hold_uncert + whs >= max_clk_trans**
   where,
   hold_uncert = clk_opt_cts__timing__hold_uncert__set
   whs = route_opt__timing__whs__worst
   max_clk_trans = route_opt__timing__clk_trans__max

3. Metrics: **A^2 * D**
   where,
   A = route_opt__area__cell__tot
   D = route_opt__timing__critpath__max

Setting up the silicon compiler:

proj1/pdrm/dcrm/rm_setup/common_setup.tcl: Modifications [**in bold**]

```
...
set DESIGN_NAME                    "maxr_lemt"

set DESIGN_REF_DATA_PATH           "../src/rtl/proj1"   ;#  Absolute path
prefix variable for library/design data.
...
set ADDITIONAL_SEARCH_PATH         "../src/rtl/proj1"   ;#  Additional search
path to be added to the default search path
...
```

proj1/pdrm/dcrm/rm_setup/dc_setup.tcl:

```
...
set RTL_SOURCE_FILES  "htm.v";# Enter the list of source RTL files if
reading from RTL
...
```

$ cp proj1/pdrm/template/counter.constratints.tcl proj1/pdrm/template/maxr_lemt.constraints.tcl

```
set clkname clk
...
```

proj1/pdrm/template/icc2_common_setup.tcl:

```
...
set DESIGN_NAME                 "maxr_lemt" ;# Required; name of the design to
be worked on; also used as the block name when scripts save or copy a block
...
```

proj1/pdrm/Makefile:

```
...
DESIGN=maxr_lemt

CLK_PER=5
UTIL=0.5
MAXLYR=met5
MAXTRANS=0.5
CLKUNCERT=0.2
...
...
setup:
    echo
synth__sched__date__begin,synth__sched__date__end,init_design__sched__date__be
gin,init_design__sched__date__end,place_opt__sched__date__begin,place_opt__sch
ed__date__end,clock_opt_cts__sched__date__begin,clock_opt_cts__sched__date__en
d,clock_opt_opto__sched__date__begin,clock_opt_opto__sched__date__end,route_au
to__sched__date__begin,route_auto__sched__date__end,route_opt__sched__date__be
gin,route_opt__sched__date__end,route_opt__sched__host__id,route_opt__sched__l
oadavg__end,synth__timing__clk_per__set,init_design__area__util__set,init_desi
gn__area__max_layer__set,place_opt__timing__clk_trans__set,clk_opt_cts__timing
__hold_uncert__set,route_opt__timing__whs__worst,route_opt__timing__tnhs__tot,
route_opt__timing__nhve__tot,route_opt__timing__ntv__tot,route_opt__timing__cr
itpath__max,route_opt__timing__clk_trans__max,route_opt__area__cell__tot,route
_opt__area__util__avg,route_opt__drc__viol__tot,route_opt__area__wirelength__t
ot,route_opt__area__li1_wirelength__tot,route_opt__area__met1_wirelength__to
t,route_opt__area__met2_wirelength__tot,route_opt__area__met3_wirelength__to
t,route_opt__area__met4_wirelength__tot,route_opt__area__met5_wirelength__to
t,route_opt__power__int__tot,route_opt__power__sw__tot,route_opt__power__lea
k__tot > results.csv
...
```

proj1/pdrm/set_constraints.py:

```
...
dest=open(f'src/rtl/proj1/{design}.constraints.tcl','w')
...
```

proj1/pdrm/parse_report.py:

```
...
  m=re.search(r'^\s*slack\s+\([A-Z]+\)\s+([-+0-9\.]+)',line)
  if m:
    if scenario=='mode_norm.fast.RCmin_bc' and group=='clk':
      whs=m.group(1)
      break
f.close()

# route_opt__timing__whs__worst
results.write(whs+',')
...
  m=re.search(r"^Critical Path Length:\s+([0-9\.\-]+)",line)
  if m:
    if scenario=='mode_norm.slow.RCmax' and group=='clk':
      crit_path_len=m.group(1)
    continue
  m=re.search(r"^Total Hold Violation:\s+([0-9\.\-]+)",line)
  if m:
    if scenario=='mode_norm.fast.RCmin_bc' and group=='clk':
      tnhs=m.group(1)
    continue
  m=re.search(r"^No. of Hold Violations:\s+([0-9]+)",line)
  if m:
    if scenario=='mode_norm.fast.RCmin_bc' and group=='clk':
      nhve=m.group(1)
    continue
...
...
  m=re.search(r'Total Routed Wire Length =\s+(\d+)',line)
  if m:
    wire_len=m.group(1)
    continue

  m=re.search(r'\s*Layer\s*li1\s*:\s*(\d+)',line)
  if m:
    li1_wire_len=m.group(1)
    print('#########################################')
    print('New statistics [wirelength -- route_opt.check_routes]')
    print(f'route_opt__area__li1_wirelength__tot = {li1_wire_len}')
    continue

  m=re.search(r'\s*Layer\s*met1\s*:\s*(\d+)',line)
  if m:
    met1_wire_len=m.group(1)
    print(f'route_opt__area__met1_wirelength__tot = {met1_wire_len}')
    continue
```

```
  m=re.search(r'\s*Layer\s*met2\s*:\s*(\d+)',line)
  if m:
    met2_wire_len=m.group(1)
    print(f'route_opt__area__met2_wirelength__tot = {met2_wire_len}')
    continue

  m=re.search(r'\s*Layer\s*met3\s*:\s*(\d+)',line)
  if m:
    met3_wire_len=m.group(1)
    print(f'route_opt__area__met3_wirelength__tot = {met3_wire_len}')
    continue

  m=re.search(r'\s*Layer\s*met4\s*:\s*(\d+)',line)
  if m:
    met4_wire_len=m.group(1)
    print(f'route_opt__area__met4_wirelength__tot = {met4_wire_len}')
    continue

  m=re.search(r'\s*Layer\s*met5\s*:\s*(\d+)',line)
  if m:
    met5_wire_len=m.group(1)
    print(f'route_opt__area__met5_wirelength__tot = {met5_wire_len}')
    print('#########################################\n\n')
    break

f.close()

# route_opt__drc__viol__tot
results.write(viol+',')
# route_opt__area__wirelength__tot
results.write(wire_len+',')
# route_opt__area__li1_wirelength__tot
results.write(li1_wire_len+',')
# route_opt__area__met1_wirelength__tot
results.write(met1_wire_len+',')
# route_opt__area__met2_wirelength__tot
results.write(met2_wire_len+',')
# route_opt__area__met3_wirelength__tot
results.write(met3_wire_len+',')
# route_opt__area__met4_wirelength__tot
results.write(met4_wire_len+',')
# route_opt__area__met5_wirelength__tot
results.write(met5_wire_len+',')

f=open('icc2rm/rpts_icc2/route_opt.report_power')
for line in f:
    m = re.search(r"^\s*Cell Internal Power\s*=\s*([\d.e+-]+)\s*nW", line)
```

```
    if m:
        power_int=m.group(1)
        print('########################################')
        print('New statistics [power --  route_opt.report_power] ')

        print(f'route_opt__power__int__tot = {power_int}')
        continue
    m = re.search(r"^\s*Net Switching Power\s*=\s*([\d.e+-]+)\s*nW", line)
    if m:
        power_sw=m.group(1)
        print(f'route_opt__power__sw__tot = {power_sw}')
        continue
    m = re.search(r"^\s*Cell Leakage Power\s*=\s*([\d.e+-]+)\s*nW", line)
    if m:
        power_leak=m.group(1)
        print(f'route_opt__power__leak__tot = {power_leak}')
        print('########################################')
        break

f.close()

# route_opt__power__int__tot
results.write(power_int+',')
# route_opt__power__sw__tot
results.write(power_sw+',')
# route_opt__power__leak__tot
results.write(power_leak+'\n')

results.close()
```

proj1/pdrm/run.sh:

```
#!/bin/bash
# change configurations for a valid set of values:
CLK_PER_VALUES=(1.0)
UTIL_VALUES=(0.5 0.52)
MAXTRANS_VALUES=(0.3 0.32 0.35 0.37 0.4 0.42 0.45 0.47 0.5 0.52 0.55 0.57
0.6)
CLKUNCERT_VALUES=(0.1 0.12 0.15 0.17 0.2 0.22 0.25 0.27 0.3)

for CLK_PER in "${CLK_PER_VALUES[@]}"
do
    for UTIL in "${UTIL_VALUES[@]}"
    do
        for MAXTRANS in "${MAXTRANS_VALUES[@]}"
        do
            for CLKUNCERT in "${CLKUNCERT_VALUES[@]}"
            do
```

```
                echo -e "Running make clean and make with the following
configuration:\nCLK_PER=$CLK_PER \nUTIL=$UTIL \nMAXLYR=met5
\nMAXTRANS=$MAXTRANS \nCLKUNCERT=$CLKUNCERT\n\n"
                make clean
                make CLK_PER=$CLK_PER UTIL=$UTIL MAXTRANS=$MAXTRANS
CLKUNCERT=$CLKUNCERT
            done
        done
    done
done
```

Observations:

Note:

- Please refer to *results.csv* for details on other invalid runs.
- The tables below include only valid observations according to project requirements.

Comment (1): [Runs for minimum delay]

- I conducted several experiments with a 5 ns clock and then gradually, reduced the clock period to 1 ns to verify if the design adhered to the constraints.
- The design ran successfully even at 1 ns.
- To determine the optimal set of parameters for these configurations, I created a run.sh script with varying values of {MAXTRANS, CLKUNCERT, UTIL} while keeping CLK_PER = 1 and MAXLYR = met5 constant. Since my metric is A^2 * D, I need to minimize both area and delay, with a greater emphasis on reducing the area.

Comment (2): [Runs for minimum area]

- Using the configurations defined in the run.sh script, I derived a preliminary set of values that can now be used to optimize the design's total cell area.
- After relaxing all the constraints to optimize for the minimum area, I found that the minimum possible total cell area is around ~8.8 - 9.1 E+04.
- These values come for CLK_PER = 3.0. For higher CLK_PER, this total cell area remained around the minimum value but the metric was affected adversely because of a slower clock.
- Further runs to obtain the best metric values for CLK_PER=3.0 and w/varying UTIL, MAXTRANS, and CLKUNCERT

Comment (3): [Runs for minimum metric]

- Best metric values obtained for CLK_PER=3.0, UTIL=0.6, MAXLYR=met5, MAXTRANS=0.3, CLKUNCERT=0.15.

- I tried to rerun the experiments with the best possible constraints, however, the metric values obtained were higher for later runs.

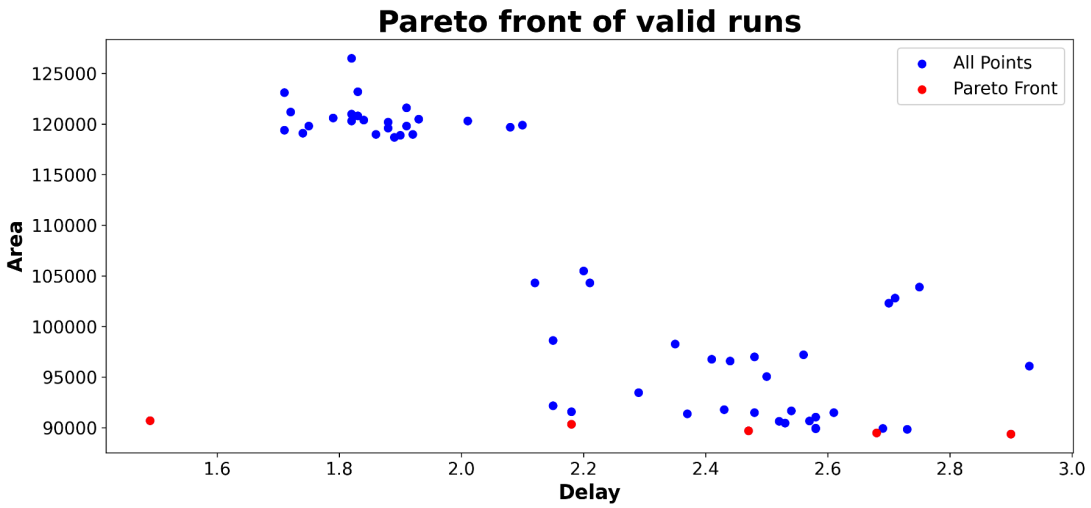

Fig. 1: Pareto front of valid runs

[w/varying {CLK_PER, UTIL, MAXTRANS, CLKUNCERT}; MAXLYR = met5]

Best constraints for Metric (A^2 * D), critpath (D), and area (A)

set constraints:				observed statistics:						Metric:
CLK_ PER	UTIL	MAX TR ANS	CLK UN CERT	whs	hold_ uncert + whs	max_ clk_ trans	viol _tot	cell_ area_ tot **(A)**	crit path **(D)**	**A^2 * D**
3	**0.6**	**0.3**	**0.15**	**0**	**0.15**	**0.12**	**0**	**9.069E+04**	**1.49**	**1.225E+10**
3	0.6	0.3	0.1	0.01	0.11	0.11	0	9.035E+04	2.18	1.779E+10
3	0.6	0.3	0.2	0	0.2	0.12	0	9.217E+04	2.15	1.826E+10
3	0.65	0.3	0.15	0	0.15	0.11	0	9.157E+04	2.18	1.828E+10
3	0.65	0.3	0.1	0.01	0.11	0.11	0	9.136E+04	2.37	1.978E+10
3	0.85	0.3	0.15	0	0.15	0.12	0	8.968E+04	2.47	1.986E+10
3	0.65	0.3	0.2	0	0.2	0.11	0	9.346E+04	2.29	2.000E+10
3	0.8	0.3	0.2	0	0.2	0.12	0	9.178E+04	2.43	2.047E+10

3	0.8	0.3	0.15	0	0.15	0.12	0	9.045E+04	2.53	2.070E+10
3	0.7	0.3	0.2	0	0.2	0.12	0	9.063E+04	2.52	2.070E+10
3	0.85	0.3	0.2	0	0.2	0.12	0	9.148E+04	2.48	2.075E+10
3	0.55	0.3	0.15	0	0.15	0.14	0	8.990E+04	2.58	2.085E+10
3	0.7	0.3	0.15	0.01	0.16	0.12	0	8.995E+04	2.58	2.087E+10
3	0.65	0.3	0.25	0	0.25	0.13	0	9.862E+04	2.15	2.091E+10
3	0.95	0.3	0.2	-0.01	0.19	0.11	0	9.068E+04	2.57	2.113E+10
3	0.75	0.3	0.2	0	0.2	0.12	0	9.168E+04	2.54	2.135E+10
3	0.55	0.3	0.2	0	0.2	0.14	0	9.106E+04	2.58	2.140E+10
3	0.9	0.3	0.15	0	0.15	0.12	0	8.949E+04	2.68	2.146E+10
3	0.75	0.3	0.15	0	0.15	0.12	0	8.994E+04	2.69	2.176E+10
3	0.9	0.3	0.2	-0.01	0.19	0.12	0	9.149E+04	2.61	2.185E+10
3	0.3	0.3	0.1	0	0.1	0.1	0	8.985E+04	2.73	2.204E+10
3	0.85	0.3	0.25	-0.01	0.24	0.12	0	9.676E+04	2.41	2.256E+10
3	0.7	0.3	0.25	0	0.25	0.12	0	9.506E+04	2.5	2.259E+10
3	0.8	0.3	0.25	0	0.25	0.12	0	9.828E+04	2.35	2.270E+10
3	0.55	0.3	0.25	0	0.25	0.14	0	9.660E+04	2.44	2.277E+10
3	0.65	0.3	0.3	0	0.3	0.11	0	1.043E+05	2.12	2.308E+10
3	0.95	0.3	0.15	0	0.15	0.12	0	8.937E+04	2.9	2.316E+10
3	0.6	0.3	0.25	0	0.25	0.12	0	9.700E+04	2.48	2.334E+10
3	0.8	0.3	0.3	-0.09	0.21	0.11	0	1.043E+05	2.21	2.403E+10
3	0.75	0.3	0.25	0	0.25	0.12	0	9.720E+04	2.56	2.418E+10
3	0.6	0.3	0.3	0	0.3	0.12	0	1.055E+05	2.2	2.449E+10
1	0.52	0.32	0.25	-0.09	0.16	0.12	0	1.191E+05	1.74	2.470E+10
1	0.52	0.32	0.17	-0.02	0.15	0.14	0	1.198E+05	1.75	2.512E+10
1	0.5	0.3	0.22	-0.09	0.13	0.12	0	1.212E+05	1.72	2.528E+10
1	0.5	0.3	0.3	-0.18	0.12	0.12	0	1.231E+05	1.71	2.591E+10

1	0.5	0.3	0.15	-0.02	0.13	0.12	0	1.206E+05	1.79	2.604E+10
1	0.52	0.32	0.15	-0.01	0.14	0.12	0	1.190E+05	1.86	2.633E+10
1	0.5	0.3	0.17	-0.03	0.14	0.11	0	1.203E+05	1.82	2.636E+10
1	0.52	0.3	0.1	0.01	0.11	0.11	0	1.187E+05	1.89	2.661E+10
1	0.52	0.3	0.25	-0.12	0.13	0.12	0	1.210E+05	1.82	2.663E+10
1	0.52	0.3	0.22	-0.1	0.12	0.12	0	1.204E+05	1.84	2.666E+10
1	0.5	0.3	0.25	-0.12	0.13	0.12	0	1.208E+05	1.83	2.672E+10
1	0.52	0.32	0.2	-0.08	0.12	0.12	0	1.189E+05	1.9	2.685E+10
1	0.52	0.3	0.12	-0.01	0.11	0.11	0	1.196E+05	1.88	2.691E+10
3	0.9	0.3	0.25	-0.05	0.2	0.12	0	9.608E+04	2.93	2.705E+10
1	0.5	0.3	0.12	0	0.12	0.12	0	1.202E+05	1.88	2.718E+10
1	0.5	0.32	0.17	-0.05	0.12	0.12	0	1.198E+05	1.91	2.740E+10
1	0.52	0.3	0.27	-0.14	0.13	0.12	0	1.232E+05	1.83	2.778E+10
1	0.52	0.3	0.17	-0.05	0.12	0.12	0	1.205E+05	1.93	2.804E+10
1	0.52	0.3	0.3	-0.16	0.14	0.12	0	1.216E+05	1.91	2.825E+10
3	0.85	0.3	0.3	-0.13	0.17	0.12	0	1.023E+05	2.7	2.825E+10
3	0.7	0.3	0.3	0	0.3	0.12	0	1.028E+05	2.71	2.865E+10
1	0.5	0.35	0.2	-0.05	0.15	0.14	0	1.203E+05	2.01	2.910E+10
1	0.5	0.3	0.2	-0.08	0.12	0.11	0	1.265E+05	1.82	2.912E+10
3	0.6	0.3	0.15	0	0.15	0.12	0	9.046E+04	2.6	2.912E+10
3	0.6	0.3	0.15	0	0.15	0.12	0	9.057E+04	2.57	2.912E+10
3	0.6	0.3	0.15	0	0.15	0.12	0	9.048E+04	2.27	2.912E+10
1	0.5	0.3	0.2	-0.08	0.12	0.11	0	1.265E+05	1.82	2.912E+10
3	0.75	0.3	0.3	0	0.3	0.12	0	1.039E+05	2.75	2.966E+10
1	0.5	0.37	0.27	-0.12	0.15	0.15	0	1.197E+05	2.08	2.978E+10
1	0.52	0.35	0.12	0	0.12	0.12	0	1.199E+05	2.1	3.020E+10

Table 1: Minimize metric [A^2*D] (sorted smallest to largest metric)

Result:

Best constraints:

```
CLK_PER=3.0
UTIL=0.6
MAXLYR=met5
MAXTRANS=0.3
CLKUNCERT=0.15
```

Minimum metric (A^2 * D):

- **1.225E+10** [best run]
- 2.912E+10 [2nd run with same constraints]
- 2.912E+10 [3rd run with same constraints]
- 2.912E+10 [4th run with same constraints]