PRE-PLACEMENT & PLACEMENT STAGE – VIOLATION FIXES

1. Group Path Optimization

➢ What is Group Path?

Group path classifies timing paths into logical categories so the tool can **analyze and optimize them independently**, instead of treating all paths equally.

Common Path Groups

- Input → Register
- Register → Register
- Register → Output
- Clock → Register

Why Group Path is Used?

- Prioritizes **critical timing paths**
- Improves **timing convergence**
- Simplifies **timing analysis and reporting**

Where It Is Used?

- Timing analysis
- Placement and optimization stages

TCL Example – Group Path Creation

```
group_path -from [get_flat_cells -filter "is_sequential" I_BLENDER_0/*] \
        -to   [get_flat_cells -filter "is_sequential" I_BLENDER_0/*] \
        -weight 2 \
        -name I_BLENDER_0_path
group_path -from [get_flat_cells -filter "is_sequential" I_BLENDER_1/*] \
        -to   [get_flat_cells -filter "is_sequential" I_BLENDER_1/*] \
        -weight 4 \
        -name I_BLENDER_1_path
```

2. Magnet Placement

➢ What is Magnet Placement?

Magnet placement pulls **critical cells closer to reference objects** (flip-flops, macros, or ports) to reduce:

- Wire length
- Net delay
- Timing violations

Conceptual View

- Critical cell → Magnet
- Related datapath cells → Attracted toward the magnet

TCL Example – Magnet Placement

```
magnet_placement  [get_cells I_CONTEXT_MEM/I_CONTEXT_RAM_3_3]
```

3. Bound (Region) Creation

➢ **What is Bound Creating?**

Bound creation defines a **physical placement region** inside the core area where specific cells are allowed to be placed.

Types of Bounds

- **Hard Bound** – Strict placement region
- **Soft Bound** – Preferred but not enforced
- **Exclusive Bound** – Reserved only for specific cells

TCL Example – Exclusive Bound Creation

```
create_bound -name "movebound1" \
 -boundary {{{100 100} {200 200}} {{1000 1000} {2000 1000}} \
        {{2000 4000} {1500 4000}} {{1500 2000} {1000 2000}}} \
 -type hard -cells [get_cells I_CONTEXT_MEM/*]
```

CTS STAGE – SKEW & HOLD VIOLATION FIXES

➢ **Clock-path hold fixing using local clock delay insertion**

```
redirect ./reports/cts/rt_skew.rpt {
 report_timing -path_type full_clock_expanded \
 -max_paths 10 -input_pins -capacitance \
 -delay_type max -nets -nosplit \
 -significant_digits 3 -slack less_than 0 \
 -transition_time
}
```

Identifying Capture Flip-Flop Violations

Setup / Capture Analysis

```
report_timing -from [get_cells I_PCI_TOP/R_0] \
        -path_type end -max_paths 10 -nosplit
```

Hold Analysis

```
report_timing -to [get_cells I_PCI_TOP/R_0] \
        -path_type end -max_paths 10 \
        -delay_type min -nosplit
```

Buffer Insertion on Capture Clock Pin (Hold Fix)

```
insert_buffer [get_cells I_PCI_TOP/R_0/CLK] NBUFFX2_LVT
```

Purpose

- Fix hold violation
- Increase capture clock latency
- Maintain setup balance

Low-VT buffer is used for **controlled delay insertion**.

Verifying Slack After Fix

```
report_timing -from [get_cells I_PARSER/r_pcmd_out_reg[2]] \
        -to   [get_cells I_PCI_TOP/R_0] \
        -path_type full_clock
```

Confirms

- Hold violation fixed
- Positive slack achieved
- No setup degradation

Alternative Fix (If Delay Is Insufficient)

```
size_cell I_PCI_TOP/R_0/eco_cell DELLN1X2_LVT
```

Used When

- Buffer insertion alone is insufficient
- ECO-friendly higher-delay cell is required

Dump timing / constraint reports (ONE-TIME)

CTS skew / timing

Create routing report directory

```
set rpt_dir "./reports"
file mkdir $rpt_dir
redirect $rpt_dir/rt_skew.rpt {
  report_timing \
    -path_type full_clock_expanded \
    -max_paths 30 \
    -input_pins \
    -capacitance \
    -delay_type max \
    -nets \
    -nosplit \
    -significant_digits 3 \
    -slack_lesser_than 0 \
    -transition_time
}
```

Max transition violations → tran.txt

```
redirect $rpt_dir/tran.txt {
  report_constraints -all_violators -max_transition
}
```

Max capacitance violations → cap.txt

```
redirect $rpt_dir/cap.txt {
  report_constraints -all_violators -max_capacitance
}
```

Now these two files are your **single source of truth**.

VT SWAP ON TRANSITION VIOLATIONS

```
proc vt_swap nn {
  set dn [get_object_name [get_flat_cells -of_objects \
    [get_pins [all_connected $nn -leaf] -filter "direction == out"]]]
  set drn [get_attribute [get_flat_cells $dn] ref_name]
  puts "driver_name : $dn driver_ref_name : $drn"
  regexp -nocase {(.+X)([0-9]+)(.+)} $drn temp rn ds vt
  if {$vt == "_RVT" || $vt == "_LVT"} {
    set vt "_HVT"
  }
  size_cell $dn $rn$ds$vt
  set drn [get_attribute [get_cell $dn] ref_name]
  puts "driver_name : $dn new_ref_name : $drn"
}
```

#Apply VT swap using dumped file

```
set fp [open $rpt_dir/tran.txt r]
while {[gets $fp line] >= 0} {
   if {[llength $line] == 5} {
      set net_name [lindex $line 0]
      catch {vt_swap $net_name}
   }
}
close $fp
```

UPSIZE FOR CAPACITANCE VIOLATIONS

```
proc upsize_cell {n} {
   set dn [get_object_name [get_flat_cells -of_objects \
      [get_pins [all_connected $n -leaf] -filter "direction == out"]]]
   set drn [get_attribute [get_flat_cells $dn] ref_name]
   puts "driver_name : $dn  driver_ref_name : $drn"
   regexp -nocase {(.+X)([0-9]+)(.+)} $drn temp rn ds vt
   if {$ds == 0} {
      set ds 1
   } else {
      set ds [expr $ds * 2]
   }
   size_cell $dn $rn$ds$vt
   set drn [get_attribute [get_cell $dn] ref_name]
   puts "driver_name : $dn  new_ref_name : $drn"
}
```

#Apply upsizing using dumped file

```
set fp [open $rpt_dir/cap.txt r]
while {[gets $fp line] >= 0} {
   if {[llength $line] == 5} {
      set net_name [lindex $line 0]
      catch {upsize_cell $net_name}
```

```
    }
}
close $fp
```

BUFFER INSERTION FOR LONG NETS

```
proc insert_buffer nn {
    set le [get_attribute [get_nets $nn] dr_length]
    if {$le <= 101} {
        set di [expr {$le/2}]
        add_buffer_on_route -repeater_distance $di \
            -lib_cell NBUFFX2_HVT [get_nets $nn] \
            -cell_prefix user_buffer
    } elseif {$le < 300 && $le > 100} {
        set di [expr {$le/2}]
        add_buffer_on_route -repeater_distance $di \
            -lib_cell NBUFFX4_HVT [get_nets $nn] \
            -cell_prefix user_buffer
    } else {
        set di [expr {$le/2}]
        add_buffer_on_route -repeater_distance $di \
            -lib_cell NBUFFX8_HVT [get_nets $nn] \
            -cell_prefix user_buffer
    }
}
```

#Apply buffering using SAME cap report

```
set fp [open $rpt_dir/cap.txt r]
while {[gets $fp line] >= 0} {
    if {[llength $line] == 5} {
        set net_name [lindex $line 0]
        catch {insert_buffer $net_name}
    }
}
```

```
close $fp
```

FINAL LEGALIZATION

```
legalize_placement -incremental
```