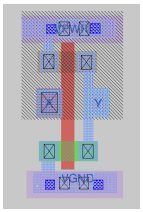
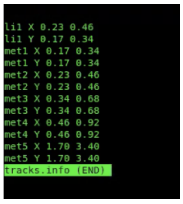
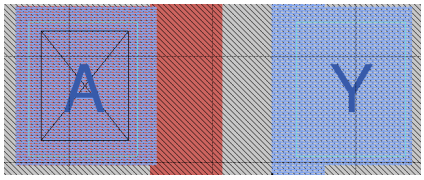
To open file:

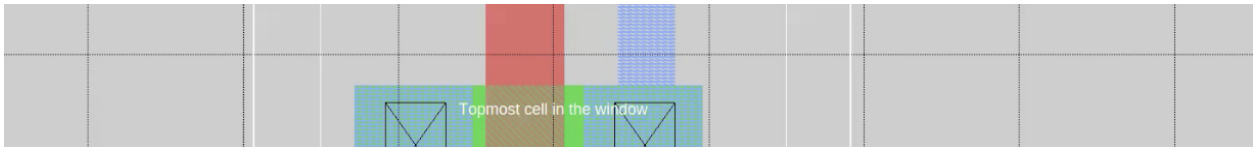
- cd vsdstdcelldesign
- magic -T sky130A.tech sky130_inv.mag &
- Now in:  (A = input, Y =output)(lef file only has input,output,power,ground - protection)
- Rules to follow in design:
    - Input + output must lie on intersection of horizontal + vertical design tracks
    - Width of standard cell should be in whole multiples of track pitch, and height augmented vertical track
- Go to /home/beaver/Desktop/work/tools/openlane_working_dir/pdks/sky130A/libs.tech/openlane/sky130_fd_sc_h
- Less tracks.info : (inside) 

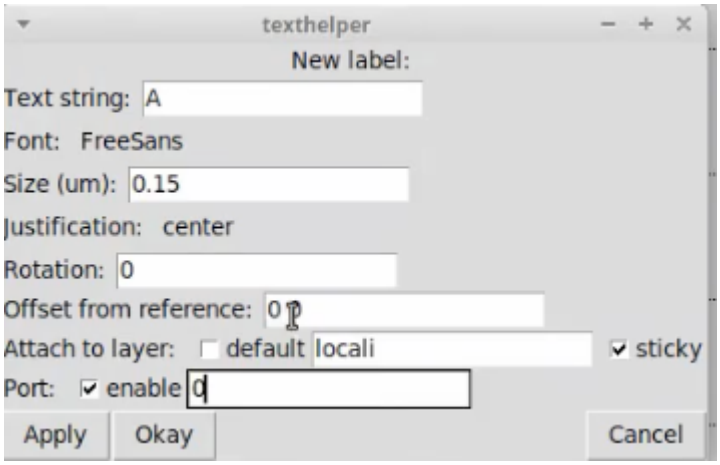
(for first input, x= horizontal, offset of 0.23, 0.46 in between each track in horizontal direction (pitch). Based on second line, vericle displacement=0.17, 0.34 in between each)
- Every metal layer x + y direction
- Tracks (metal places) used in routing - what to connect
- To check if follows track intersection rule:
    - In command line, grid on
    - Info on arguments needed in grid:

```
% help grid

Global Commands
---------------

Layout Commands
---------------
grid [xSpacing [ySpacing [xOrigin yOrigin]]]
                        toggle grid on/off (and set parameters)
scalegrid a b   scale magic units vs. lambda by a / b
snap [internal|lambda|user]
                        cause box to snap to the selected grid when moved
                        by the cursor
```

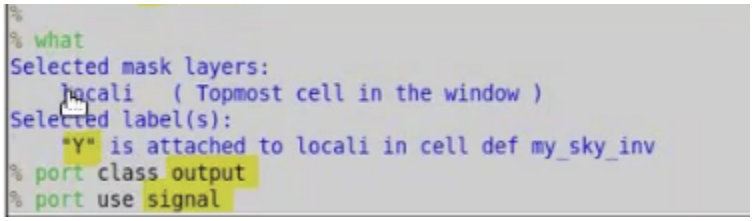
- ‘Grid 0.46um 0.34um 0.23um 0.17um’ — now new dimensions of tracks - adds tracks, we can see horizontal and vertical to intercept below input + output: 

- To check second rule (width of cell should be in odd multiples of x pitch) – width = 3.5 boxes,



  - Same rule for height
- To convert label to port:
  - Select label (where you want port)
  - Edit -> text 

(lower level enable = sooner on lef file)(power (port use power) + gnd(port use ground) should be attached to metal 1, input+output locali)
  - If you want to see this info, select label and type what:

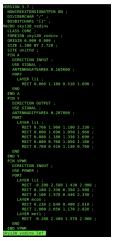


To extract lef file:

- Once port labels set
- Name cell ‘save sky130_vsdinv.mag’
- In vsdstdcelldesigns dir, magic -T sky130A.tech sky130_vsdinv.mag
- ‘Lef write’ in magic consul

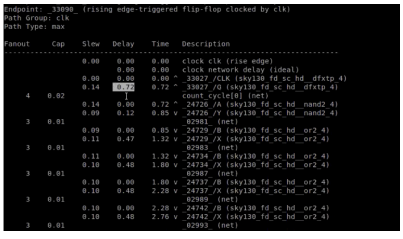
- New item has been added:

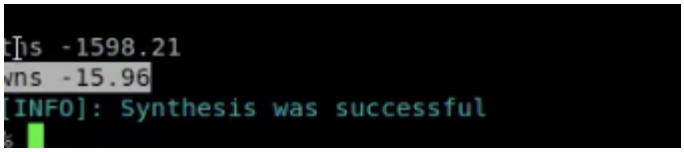
```
-rw-r--r-- 1 root root  1321 Jul 23 11:07 sky130_vsdinv.lef
```

- Inside lef (changes + modifications I've made):  (shows ports + info we declared)
- 'cp sky130_vsdinv.lef /home/beaver/Desktop/work/tools/openlane_working_dir/openlane/designs/ci/picorv32a/src' — makes sure synthesis maps to cells, has cell defs for synthesis, full characterization (cell rise, cell transition)(has both slow(max delay) and fast)
- Touch cong.tcl, copy in

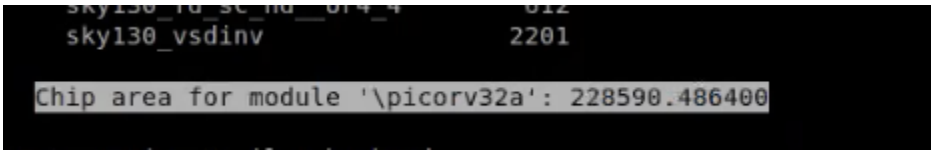
- In launcher:
  - ./flow.tcl -interactive
  - Package require openlane
  - prep -design designs/ci/picorv32a
  - If want continue old run:

```
0.9
% prep -design picorv32a -tag 15-09_06-28 -overwrite
```

  - set lefs [glob $::env(DESIGN_DIR)/src/*.lef]
  - add_lefs -src $lefs
  - output: 
  - Displays slew, delay
  - Report from opensd. Highlighted (vnxs)is maximum slack



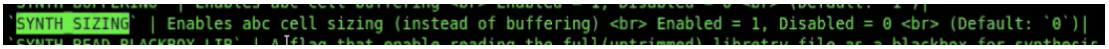
- tns= total negative slack
- To repair high slack:
  - Balance delay+area:



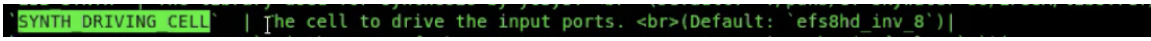
  - Synth strategy description:

  - (now set to 2, 0=most attention on delay)set $::env(SYNTH_STRATEGY) 1 (area will increase, but delay decrease)
  - Synth-buffering description: (good bc reduces wire delay by inserting buffers) (right now 1)

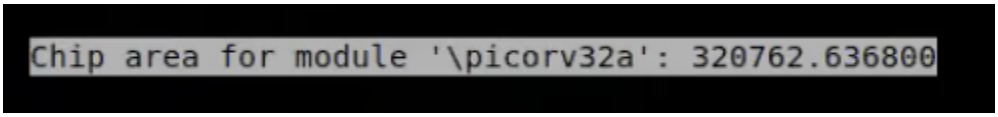
  - Synth-sizing description (right now 0) (size cell for delay minimization)

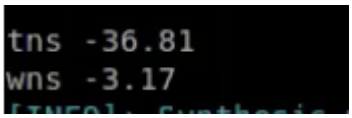


  - set $::env(SYNTH_SIZING) 1
  - Synth_driving_cell description (right now sky130_fd_sc_hs_inv_8) (more fan-outs = need for more drive-strength)



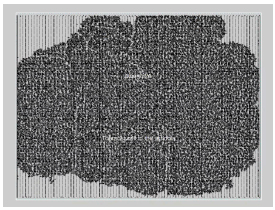
  - Running synthesis again, new area:



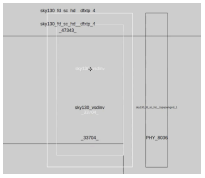
    new slack: 

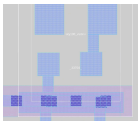
– much better
  - In merged.lef in latestrun/tmp, vsd there, confidence that in placement cell will be present
  - Run_placement
  - Go to results/placement in run, start up magic:

    - Magic - T /…(etc)…../sky130A.tech lef read /…(etc)…/merged.lef def read picorv32a.def &



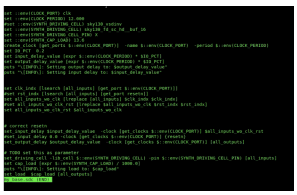
    - To find our cell: cat picorv32a.placement.def | grep sky130_vsdinv
    - Pick any instance, and in consul of magic
    - select cell xyz
    - Apartment - makes sure ground + power connected to cell

 (our cell)



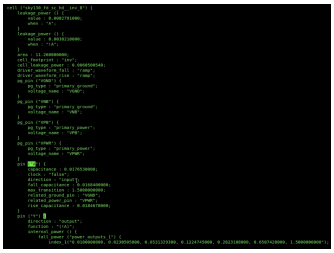
    - In consul: ‘expand’ (see layers of our cell)
- In openlane folder, new file pre_sta.conf:

(slow, fast, where read verilog)
- In certain_run/results/synthesis, one file, where logic mapped to gates
- In future stages will see clock tree in synthesis files, but not yet, created w/ netlists
- Inside my_base.sdc(referenced in pre_sta.conf): (define clock port, clock period)



- Go in via vi to change, update SYNTH_DRIVING_CELL to sky130_fd_sc_hd__inv_8
- SYNTH_DRIVING_CELL_PIN to Y

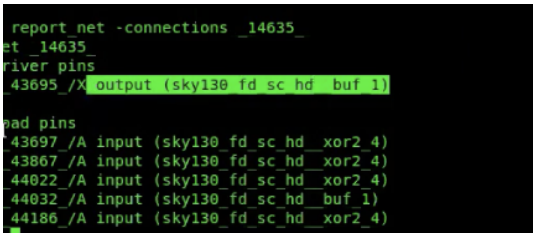
- Vi sky130_fd_sc_hd__typical.lib,(in openlane) search fir inv_8 to get cap load, which need change in my_base.sdc



- Get capacitance = 0.017653, modify in my_base.sdc
- Switch to my_base.sgc in read line so accessing right settings
- More skew + output capacitance = more delay
- SYNTH_MAX_FANOUT description:

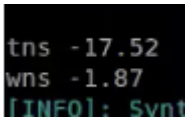
, set to 4 (too big fanout might cause delay – net delay w/ 5 fanouts):

- Run_synthesis
- Report_net -connections _14635_ (the net) – gives driver, Buffer , load pins – fanout increasing slew

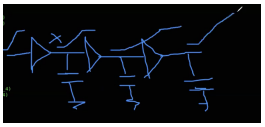


- New area: (slightly bigger)

- New slew (much smaller)



- Clack now -1.87 (desired time to arrive - actual time)
- Sta pre_sta.conf - where all info of individual pieces slew is stored
- In this file, we see buffer 1 adding a ton of delay: (line of increasingly bad slew - ability to change output based on input)



- To fix: replace_cell _41882_ sky130_fd_sc_hd__buf_4 (area grows, but should be less delay)

- Report_checks -fields {net cap clew input_pins}(fields we want to see) -digits 4(expand after decimal)
- Slack goes down to 1.4421 (upsizing cell decreases delay)
- Go through replacing long delay buffers w/ bigger buffers (signif. Timing betterment - just one more (44332) and -1.053 slack)
- Violation unless + value for slack, so keep upsizing
- Have to make sure openlane flow has new netlist post these buffer changes so,

```
write_verilog [-sort] [-remove_cells cells] filename
% write_verilog /home/nickson/Desktop/work/tools/openlane_working_dir/openlane/designs/picorv32a/runs/15-09_06-28/results/synthesis/picorv32a.synthesis.v
```

- Run_floorplan (no more synthesis, or will undo all we did)
- Run_placement
- Area (increased):

```
520.000000 : 10880.000000 - 1229120.000000 : 927520.000000
DieArea:
```

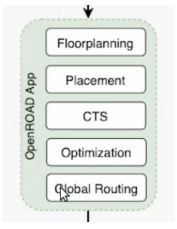
- Params in cts step: (target skew sets global skew, root buffer makes greater drive force, clock tree synth actually does it, tolerance int representing balance of QoR (quality of results) + runtime

```
### CTS

| Variable      | Description                                                                  |
|---------------|------------------------------------------------------------------------------|
| `CTS_TARGET_SKEW` | The target clock skew in picoseconds. <br> (Default: `20` ps)|
| `CTS_ROOT_BUFFER` | The name of cell inserted at the root of the clock tree. <br> (Default: `efs8hd_clkbuf_16`)|
| `CLOCK_TREE_SYNTH` | Enable clock tree synthesis for tirtonCTS. <br> (Default: `1`)|
| `CTS_TOLERANCE` | an integer value that represents a tradeoff of QoR and runtime. Higher values will produce smaller runtime but worse QoR <br> (Default: `100`) |

### Routing
```

- Run_cts
- In cts clock buffers get added, modifies netlist (cts file in synthesis)
- In openlane/scripts/tcl_commands, one for each status we've done so far
- Inside cts, procs (functions) name of proc + procedure

```
proc set_core_dims {args} {
    set options {{-log_path required}}
    parse_key_args "set_core_dims" args values $options
    set log_path $values(-log_path)
    set fpOutDef $::env(lpPlacer_tmp_file_tag).def
    set def_units $::env(DEF_UNITS_PER_MICRON)
    set coreinfo [join [exec $::env(SCRIPTS_DIR)/extract_coreinfo.sh $fpOutDef] " "]
    set sites_per_row [lindex $coreinfo 8]
    set step [lindex $coreinfo 9]
    set core_area_llx [expr { [lindex $coreinfo 4]/double($def_units) }]
    set core_area_urx [expr { ([lindex $coreinfo 6]+$step*$sites_per_row)/double($def_units) }]
    set core_area_lly [expr { [lindex $coreinfo 5]/double($def_units) }]
    set core_area_ury [expr { [lindex $coreinfo 7]/double($def_units) }]
    set ::env(CORE_WIDTH) [expr {$core_area_urx - $core_area_llx} ]
    set ::env(CORE_HEIGHT) [expr {$core_area_ury - $core_area_lly} ]
    puts "$::env(CORE_WIDTH) $::env(CORE_HEIGHT)"
}
```

- In run_cts, runs openroad (IDE tool in openlane), passes to script in openroad folder to cts.tcl



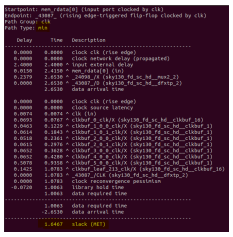
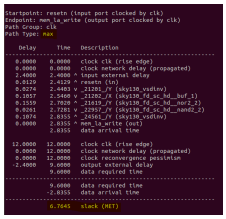
- All parts of ASIC flow done by openroad
- Max_slew = 10% clock cycle time
- Max_gap = max gap of output buffer

    - CTS-CLK_BUFFER_LIST - all clock buffers
    - root-buffer
    - Max cap - max capacitance of root-buffer
- ‘Openroad’ to launch openroad (note timing diffs., db made from lef + def files)
- Steps To create db file:
    - read_lef (place of merged lef –specific_run/tmp/merged.lef)
    - read def file from the CTS stage:

```
% read_def designs/picorv32a/runs/19-03_16-40/results/cts/picorv32a.cts.def
```

    - Create db: write_db pico_cts.db
    - Read :
        - db
        - verilog files
        - Library
        - Sdc
    - Set propagated delay for all clocks’ delay, w/ set_propogated_clock [all_clocks]
    - Check timing:

```
% report_checks -path_delay min_max -format full_clock_expanded -digits 4
```

    - Min: 
    - Max: 
- TritonCTS only optimizes once corner, but other libraries previously used do timing analysis based on min + max corners –inaccurate
- So need to leave + relaunch openroad to check timing for typical corner

```
% exit
%
% openroad
OpenROAD 0.9.0 1415572a73
This program is licensed under the BSD-3 license. See the LICENSE file for details.
Components of this program may be licensed under more restrictive licenses which must be honored.
%
% read_db pico_cts.db
%
% read_verilog designs/picorv32a/runs/19-03_16-40/results/synthesis/picorv32a.synthesis_cts.v
%
% read_liberty $::env(LIB_SYNTH_COMPLETE)
1
%
% link_design picorv32a
[WARNING ORD-1000] LEF master sky130_fd_sc_hd__tapvpwrvgnd_1 has no liberty cell.
%
% read_sdc designs/picorv32a/src/my_base.sdc
[INFO]: Setting output delay to: 2.4000000000000004
[INFO]: Setting input delay to: 2.4000000000000004
[INFO]: Setting load to: 0.01765
%
% set_propagated_clock [all_clocks]
%
```

- Steps to do this:
- Run check :

```
% report_checks -path_delay min_max -fields {slew trans net cap input_pin} -format full_clock_expanded -digits 4
```

- Hold + slack time:

```
   1.7707   library hold time
   1.7707   data required time
------------------------------
   1.7707   data required time
  -1.8506   data arrival time
------------------------------
   0.0799   slack (MET)
```

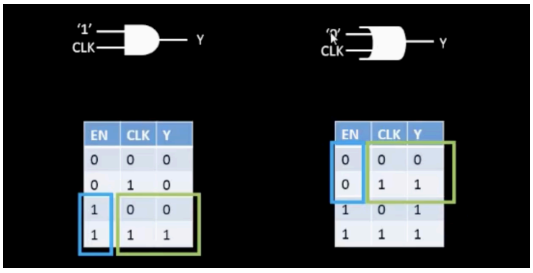
```
  13.2596   library setup time
  13.2596   data required time
------------------------------
  13.2596   data required time
  -9.3159   data arrival time
------------------------------
   3.9437   slack (MET)
```

- During cts, buffers from CTS_CLK_BUFFER_LIST are added to try and meet skew (can be modified based on requirements)
- As TritionCTS builds clock tree, tried to use all buffers in $::env(CTS_CLK_BUFFER_LIST) (sky130_fd_sc_hd__clkbuf_1 sky130_fd_sc_hd__clkbuf_2 sky130_fd_sc_hd__clkbuf_4 sky130_fd_sc_hd__clkbuf_8 from smallest to biggest, until skew is met (target = $::env(CTS_TARGET_SKEW)
- sky130_fd_sc_hd__clkbuf_1 is the most used buffer, can switch this val ($::env(CTS_CLK_BUFFER_LIST)) (we can do it via lreplace (ex below)) to see effect on STA + area

```
% echo $::env(CTS_CLK_BUFFER_LIST)
sky130_fd_sc_hd__clkbuf_8 sky130_fd_sc_hd__clkbuf_4 sky130_fd_sc_hd__clkbuf_2
%
% set ::env(CTS_CLK_BUFFER_LIST) [lreplace $::env(CTS_CLK_BUFFER_LIST) 0 1]
sky130_fd_sc_hd__clkbuf_2
%
```

- Want to set $::env(CURRENT_DEF) used by CTS in DEF file as placement's def file rather than that of previous cts. So, set $::env(CURRENT_DEF) to placement def and run_cts again
- Now only buf_2 clock buffer (before was buf_1)
- Hte + wns are much better bc bigger buffers

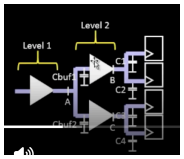
Power Aware CTS:

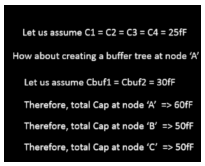
- Clock in and and or gate: 

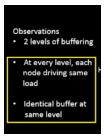
- And - only when clock on could go through gate
- Or - only opens when clock on – save power

Specific example:

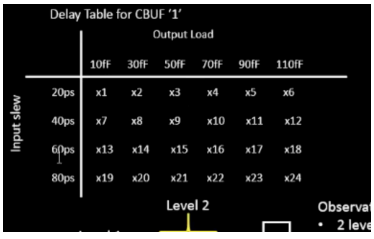
- Level one buffer drives load of 2 buffers, put load of four flops onto one buffer



- assumptions: 

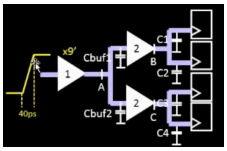
- Observations: 

(2 – level one same to self, level two each same)(3 – all at one level same size)
- Output capacitance for buffer is not constant for whole clock – load at output can vary
- Varying input transition + output load
- Delay tables:
    - To combat these issues
    - Specific range of inputs + outputs tp buffer put on one table – becomes timing model of specific size + threshold voltage:

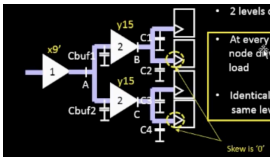


(values = delay numbers)

    - One for each type of cell, based on sizes + ration of pmos and nmos size (more = less resistance = varying delay) 1(smallest)-10(biggest)
    - Build equation from table, not just vals on table

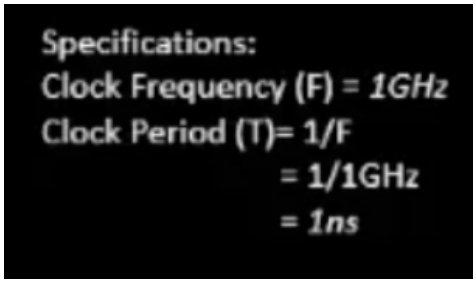
- With input 40ps, what will be clock latency (delay) at end? We must add all delays up. 

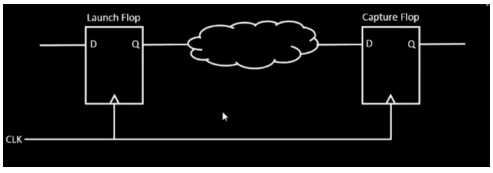
- Then take output to put input next buffer: 
- Although there are 2 buffers w/ y15 delay, on one branch only one factors into delay

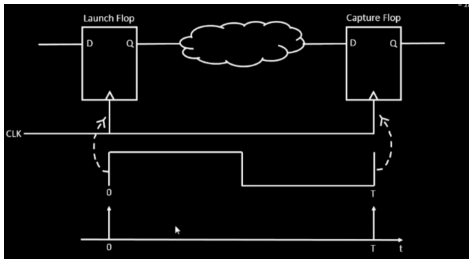


- Skew = 0 at these end points(at same time)
- So x9'+y15 = latency
- Last flipflops only on under certain conditions

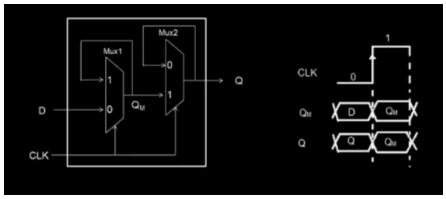
Setup timing analysis (w/ ideal clocks)(single clock):

- specs: 

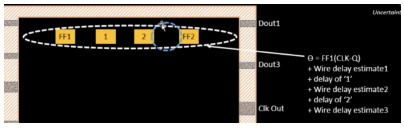
- Launch flop -> combinational logic -> capture flop (connected by ideal clock) 

- In one clock cycle, send posedge to launch flop. On 'T'th time, reaches to capture flop 

(bottom line on scale of time)
- Assume combinational logic delay of theta. This is less than clock cycle time. as time period goes up, frequency goes down (inverse)
- Image inside of flop. Mux1 delay of D to Q (when clock=0) output moves back to input of mux2 –output doesn't change until clock goes to logic 1

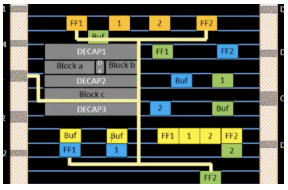
(output mux1 goes back in as input, and goes to second Q – this is a finite delay) 

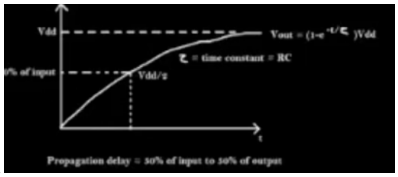
- Time it takes D to get to first Q subtracted from T (some amount of time before T needed for slop to transmit data within = mux1 delay, setup time) now theta < T-setup
- Some signal (phase log loops) that supplies clock, has bias, might not provide edges at exactly 0 or T
- There is a window within which clock edge can arrive – temporary variation (jitter)

- theta < T - setup - setup_uncertainty (jitter)
- Ex. combinational delay calculations: 

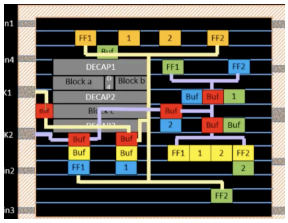
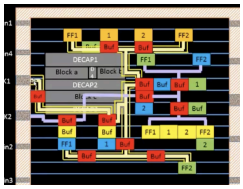
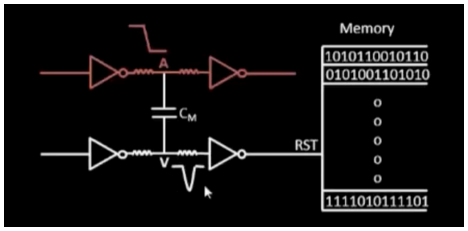
Clock Tree Synthesis:

- Example of skew: 

, want to be as little as possible
- H-Tree: take clock route, goes midpoint from distances to all next pieces, split in half and go to midpoint, etc etc 

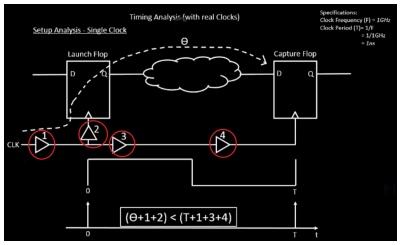
– hopefully reduces skew a lot
- Due to distances (capacitances + resistances), input to flops might not be same as output from clock 

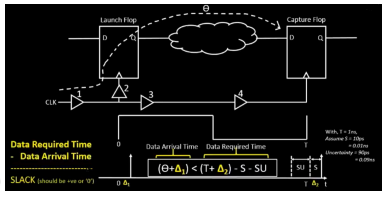
(get slope of waveform, not crisp edges)

- Add repeaters to mimic + maintain signal (for clocks must be equal rise + fall times)  (no skew)
- Protect from cross talk via Clock net shielding - protect from outside world 
- Danger of glitch (unwanted transfer of electrical energy between two conductors through electric field made by their capacitance) (capacitative coupling) 

(dip in bottom bc top goes down)
- So add wire of gnd or vdd - won't switch, so protection
- In hesitation bc of glitch, adds delay

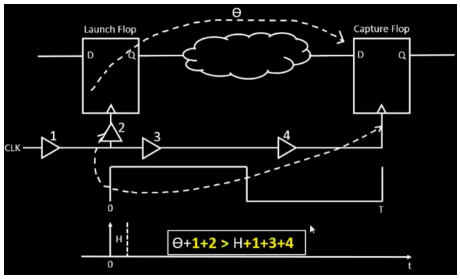
Setup timing analysis w/ real clocks:



- Buffers added to minimize signal distortion:
- Delta1 = delay of launch flop clock network, delta2 = delay of capture flop clock
- So, theta+delta1 < (T+delta2)-S(time for D to middle on capture flop)-SU(uncertainty) 

- Slack (signal required time - arrival time) must be 0 or positive
- Hold time – time need by mux2 to transmit data out. In this time, launch flop must retain the data. Occurs on same rising edge for both launch and capture flop (unlike setup, where on two different)

Hence finite Time 'H' required (after clk edge) for 'Q_M' to reach Q i.e. internal delay of Mux2 = Hold time

- Hold violation – path too fast, caused by:
    - combinational delay
    - clock buffer delays
    - hold time.
- time period and setup uncertainty don't matter, bc both flops on same posedge
- capture and flop get same clk 

-