JK Flip-Flop with Clear

Joshua Gould
VLSI 09414-1
December 19, 2018
Emails: gouldj5@students.rowan.edu

***Abstract*—JK Flip-Flop is a variation of the flip-flop that modifies the SR Latch. Flip-flops offer memory for logic operations. This logic is heavily used for data storage and transfer as well as in registers for the storage of binary data. This paper focuses on the optimization of a JK flip-flop delay; optimizing logical effort to pick a chosen topology and then designing said topology in Cadence.**

I. Introduction and Objectives

In this paper Cadence is used as a logic editor and simulator. Cadence is used to verify the architecture of the logic circuit. Using this interface for logic design, simulation and analysis allows a verification of an optimized JK flip-flop. JK flip-flops or multivibrators are memory devices used for alternating and storing a given signal between devices. JK's are set apart from other flip-flops by having a toggle state. JK flip-flops can be made from CMOS logic, pass transistors, transmission gates, and pseudo nMOS logic. CMOS-based JK flip-flops have a relatively moderate amount of delay, transistor count, surface area and static power draw in comparison to the pass transistors, transmission gates, and pseudo nMOS gates. This paper focuses on the fabrication of the JK flip-flop using the CMOS logic design.

II. JK Flip-Flop

The JK flip-flop possesses two user-controllable inputs, labeled J and K. If J and K are different then the output Q takes the value of J at the next clock edge. The inputs are labeled J and K in honor of the inventor of the device, Jack Kilby, but pocess similar input features of a SR flip-flop as Set is J and Reset is K. This behaviour is refelected in I

A JK flip-flop is a 6 terminal device and has a block symbol as shown in figure 2. The input terminals are labeled J, K , CLR , CLK , Q , and Q'. The select signal combinations corresponding to each output signal characteristic are shown in the truth table of Table I. Table I shows the states of the JK flip-flop with descriptions of each state. Since the states of the following state depend on the previous due to the feedback loop of the flip-flop, the "last" and "previous" states must be mentioned to describe the input to output relation.

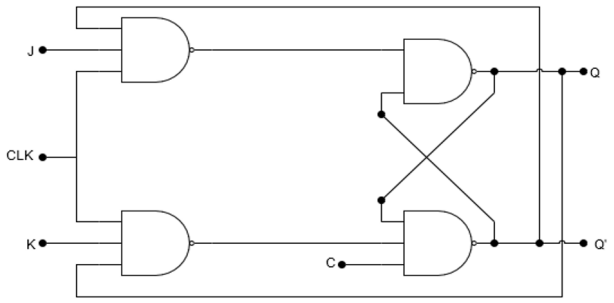


Fig. 1. Block diagram of a low-level JK Flip-Flop

J	K	CLR	CLK	Q	Q'	Action
0	0	1	Edge	Prev Q	Prev. Q'	No memory change
0	1	1	Edge	0	1	Q' set; Q low
1	0	1	Edge	1	0	Q set; Q' low
1	1	1	Edge	Last Q	Last Q'	toggle
X	X	0	X	0	1	Clear

TABLE I
Truth/Excitation table representing the JK Flip-Flop with clear

As described by Table I, the first state occurs if J and K are both low where no memory change moves states. If J and K are both high at the clock edge then the output will toggle from one state to the other. It can perform the functions of the SR flip-flop and has the advantage that there are no inherit ambiguous states [1].

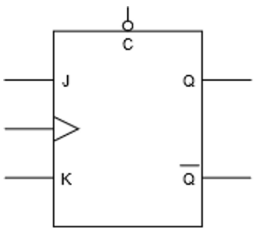


Fig. 2. JK Flip-Flop Symbol

A. Master-Slave JK Flip-Flop

In this paper, CMOS technology is used for designing the JK flip-flop with a clear. Simple designs with the JK flip-flop, however, contain run errors at the output of the circuit. As depicted by Figure 3, the output switches multiple times at the clock output. A common remediation to this problem is

a high clock speed that would increase the dynamic power consumption of the system.

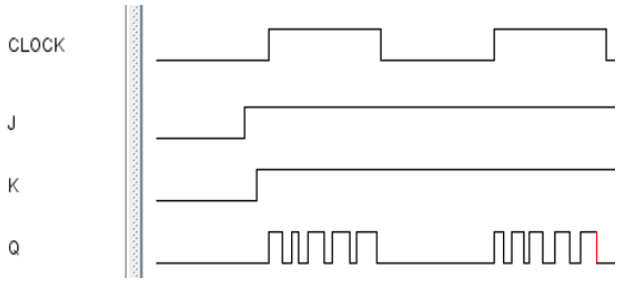


Fig. 3. Run time error on the one-stage JK flip-flop

Master-slave flip-flops are a remediation to this run time error, allowing the clock to control the slave cycle before the data is transferred at the falling edge of the clock. Due to this, the master-slave flip-flop does possess more delay due to doubling the stages of the design, and that the design must wait a half-clock period to progress to the next stage.

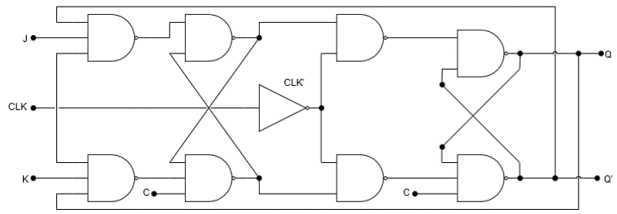


Fig. 4. Logic diagram of master-slave JK flip-flop

Noted within-in the design are the clear stages at the beginning and end of the CMOS stage, as well as the NOT gate placed to alternate the master and slave latch. The clear placed at the master and slave latch, instead of solely the master latch, allows for clear to force to both stages without a conflict of data loading to the last stage. Here, the clear does not have to wait a clock cycle to clear the circuit.

B. Topology Specification

To determine which topology presented has the least delay, rules behind the logical effort design model will compare to the designs with a overview of the designs within the circuit.

The given designs run through the differences in delay for the simple and master-slave flip-flop designs as well as a 5-stage NOR-based design. These designs possess the same features as far as satisfying the JK flip-flop characteristics with a clear. However, designs shown below have distinctive differences with regard to delay.

1) Design 1 - Simple JK Flip-flop: For this simple JK flip-flop, the logic based design uses three 3-input NAND gates and a 2-input NAND gate on the latch. Additionally, the circuit contains a given NOT gate for clock inverter. As described by Section II-A the drawback to the JK flip-flop is that is has a run timing error on the output as indicated by Figure 3. On the plus side, the simple JK flip-flop has less delay due to having one master latch in comparison to the two of the master-slave.

2) Design 2 - Master-Slave Flip-flop: The Master-Slave flip-flop, as depicted by Figure 4, contains a solely NAND-based design to allow for greater symmetry and smaller design area in comparison to a NOR design. The limitation on the Master-slave is that it requires the output to be reliant on the falling clock. This output characteristic slows the delay to half a clock cycle time inherently, even with zero delay in the input.

3) Design 3 - NOR based design: As a final design specification, the NOR-based master-slave flip-flop is provided to include NOR logic gates before the master latch that connects the clock to the inputs. This will cause the output to oscillate continuously until the one of the input signals goes low. Noticeably with this design, stages increase, inputs must be stable for some time before clock goes low, and the overall design area for NORs are larger than NAND only. This design possessed the highest number of transistors required.

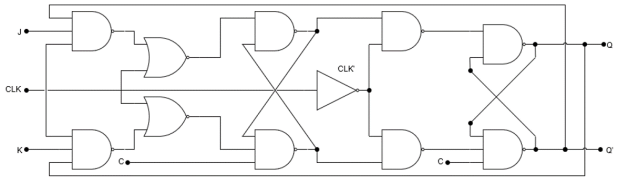


Fig. 5. NOR based topology

As a summary, the topology provided in each case is reflective of pros and cons at each junction.

Design	Major Concern	Trans.	Stages
1 - Simple	Run time error	20	2
2 - MastSlav	Higher Delay / Clock limitation	46	4
3 - NOR	Inputs must be stable	54	5

TABLE II
DESIGN OVERVIEW COMPARISON OF MAJOR CONCERNS, NUMBER OF TRANSISTORS WITHIN THE TOP-LEVEL TRANSISTOR DESIGN, AND NUMBER OF STAGES THEORETICALLY WITHIN THE DESIGN.

The summary of Table II displays the varying characteristics of each design. Nevertheless, this paper will detail optimization of the second design of the NAND-based master-slave JK flip-flop for the lowest delay without errors at an optimal four stage design.

C. Delay Specifications & Equations

To find the delay of the master-slave JK (MSJK) flip-flop design , the components g, h, p and N must be found at each of the four stages within the design. Notably, the JK flip-flop contains a few feedback loops within the design which requires experimental calculations for each latch. Additionally, the MSJK in this design possesses a separate path for the top and bottom of the design. Section IV-B overviews these paths in more detail. In summary however, each stage will behave differently on the top path and the bottom path.

Equations 1 and 2 are used to find g and p. The calculations for these equations are normalized by the unit inverter capacitance to find the proper delay output. $C_{In-Unit}$ is the

input capacitance of the unit inverter and $C_{Diff-Unit}$ is the diffusion capacitance of the unit inverter.

$$g = \frac{C_{in}}{C_{in-invunit}} \tag{1}$$

$$p = \frac{C_d}{C_{d-invunit}} \tag{2}$$

As for branching, the branching effort b of each stage is found by determining how many gates in the next stage are inputs of the next stage. Branching efforts of each stage are slightly skewed with the latch at stage two and four. Reinforcing this, b would normally equal 1 at each path (top and bottom), however, the output of the latches and the entire circuit, Q, output feeds back to another component.

$$G = \prod_{k=1}^{n} g_k \tag{3}$$

$$P = \sum_{k=1}^{n} p_k \tag{4}$$

$$B = \prod_{k=1}^{n} b_k \tag{5}$$

The path logical effort G is defined in equation 3, the path branching effort B is defined in equation 5, and the path parasitic delay P is defined in equation 4. Product of these efforts finds the path effort F with equation 6 where H is defined as equation 7. In this paper, a 90C driver with 3C load is used to give H a total of 30, however, the equation is provided as a reference.

$$F = G * H * B \tag{6}$$

$$H = \frac{C_{Driver}}{C_{Load}} \tag{7}$$

With F having been obtained using equation 6 the best stage effort $\hat{f}$ must be obtained using equation 8 where N is the number of stages of the topology (+1 for the driver).

$$\hat{f} = F^{\frac{1}{N}} \tag{8}$$

The path delay D is found with equation 9 by $\hat{f}$. The delay provided by equation 9 is delay in terms of unit inverter delay τ found with a reference inverter.

$$D = N\hat{f} + P \tag{9}$$

$$D_{time} = D * \tau \tag{10}$$

The delay of the topology allows the transistor sizes of each stage to be calculated and provides worst case delay detailed in Section IV-B. At each stage, the input capacitance must be found at h. These sizes are calculated with the latches in mind. Found with equation 11 and with (equation 12) of path effort $\hat{f}$.

$$h = \frac{C_{Load}}{C_{gate}} \tag{11}$$

$$\hat{f} = ghb \tag{12}$$

For the first stage, C_{gate} is known. For the remaining gates C_{gate} must be calculated. Starting from the last stage with driven load, Equation 11 is entered to obtain h.

$$C_{gate} = \frac{C_{load}bg}{\hat{f}} \tag{13}$$

The gate capacitance of each stage is steps through from the last to first stage. Then, the scaling multiplier k is found.

$$C_{gate} = kC_{gate-unit-size} \tag{14}$$

D. Delay calculations

Delay of each circuit input-output characteristics must be quantified by the rise, fall, and propagation delay as reflected in the time delay equations 15 to 19 with Equation 19 being the final deciding factor of comparisons in schematic, calculations, and layout designs.

$$t_r = 80\% * V_{dd} - 20\% * V_{dd} \tag{15}$$

$$t_f = 20\% * V_{dd} - 80\% * V_{dd} \tag{16}$$

Furthermore, the t_{pd} delay can be directly compared to D_{time} provided by τ.

$$t_{pdr} = .5 * V_{dd}out - .5 * V_{dd}in \tag{17}$$

$$t_{pdf} = .5 * V_{dd}in - .5 * V_{dd}out \tag{18}$$

$$t_{pd} = \frac{t_{pdf} + t_{pdr}}{2} \tag{19}$$

E. Latch Calculations

The calculations for the latches within the circuit are the most important calculations in the MSJK flip-flop as they possess the largest logical effort in this design. The latches at stages two and four, being the same, can be accounted for the same logical effort.

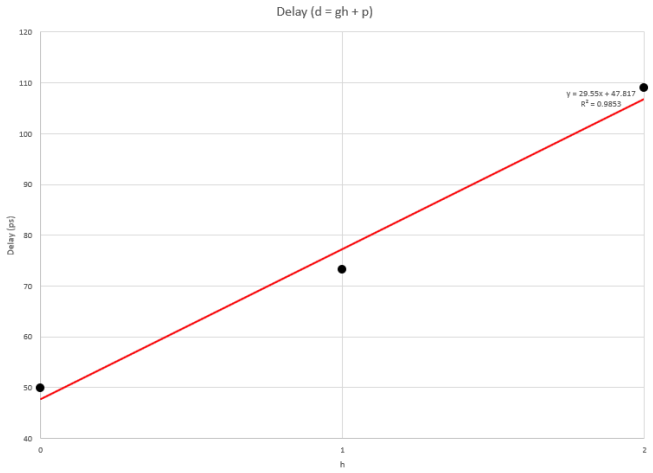


Fig. 6. Latch plot used for calculations

Latch plot trends shown above allow for the abuse of the equation $d = gh + p$ to find the logical effort of the latch after normalization. The latch was measured at the output Q and for the following fan-out stages, they were placed after the Q output. Here, the t_{pd} at each fan-out allowed for equation manipulation.

h	g
0	212.11
1	261.30
2	297.01

TABLE III
LATCH DELAY DATA

The data obtained from this plot is provided in Table III

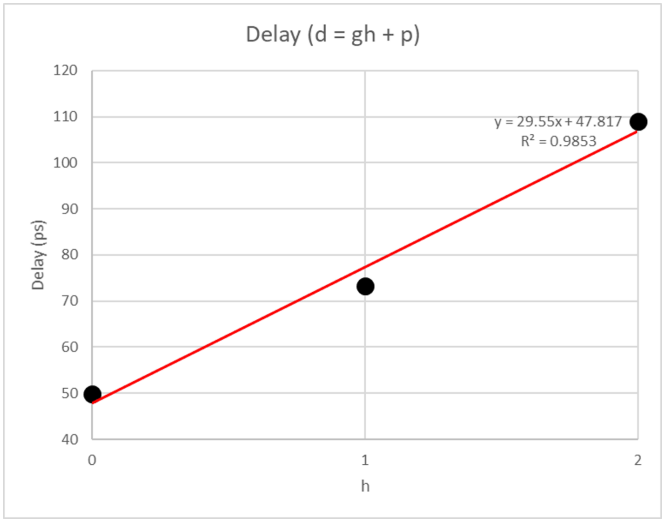


Fig. 7. Inverter delay plot to determine how to normalize

The data obtained from this plot is provided in Table IV. Here, the normalization of the inverter can be performed for the rest of the circuit at the delay trend of the inverter plot provided in 7. Here, the inverter reference came to 77.367 with a fan-out h of 1.

h	g
0	49.9
1	73.2
2	109

TABLE IV
INVERTER DELAY DATA

Normalization was then applied to the latch where the g and p values came to be 1.436 and 4.2506 respectively to be then applied to the logical effort calculation table.

III. Procedure

In order to determine the optimized MSJK flip-flop that provides the lowest delay a number of different topologies were created and are listed in Section II-B. The delay for the chosen topology was found using the equations provided in section II in which the best topology was chosen with optimized size and stages in mind. A transient analysis was then performed to determine the worst case delays with latch calculations being a focus for the given stages in the logical top-level circuit. After the transient analysis was performed the layout of that topology was created to be extracted and compared with a theoretical circuit simulation. The schematic level can be inaccurate and slower or faster with left-out parasitic resistances and capacitances. Transient analysis was then performed to compare each design and find the percent difference between each design simulated.

IV. Proposed JK Flip-Flop CMOS Design

Calculating delays for the MSJK in table IV-B allows for a standing point to see the theoretical worst-case design for the MSJK and verify the design. Design calculations at h junctions of each stage are detailed in Section IV-A

A. Gate Optimization

Gate optimization was found by performing the process of reverse logic area through Equation 12. As an automatic process, when the logic gates of the circuit, through a reverse logic is found by using the graphical process found in Figure 14.

This process allows for the h at each stage to be calculated. Notably, the area of each logic gate should theoretically increase as the data progresses through the MSJK. Gate optimization is provided in Figure 15 and 16 to show the area of each gate.

Gate optimization was found using the code found in the Github repository https://github.com/rjweld21/VLSI_Lab/tree/master/Final_JKFlipFlop for custom latch characteristics at cinCalcs.py.

B. Worst Case Path Delay

With the stage logical efforts g and stage parasitic delays p found, to find the path logical effort G and path parasitic delay P to be found, the path of the circuit can be calculated for worst case path.

Without calculations, the worst case path is estimated to be as in Figure 8.

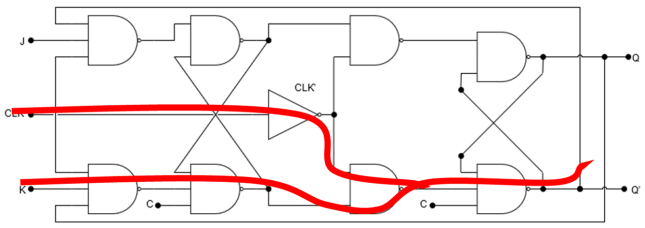


Fig. 8. Worst path shown by the red line of data through input to output of the MSJK

Given the path in question represented by the red line of figure 8 the path followed by the design flows through the greatest amount of 3-input NAND gates as well as relying on the pulse-width of the clock. With assumptions that the data flows from the input to output, and branching effort for each stage maximized, the path logical effort G, path parasitic delay P, and path branching effort B can be estimated to give the largest delay within the design.

Delay calculations at the worst case are provided below in Table IV-B where the calculated delay time come to about 2.635 ns.

Var.	**1**	**2**	**3**	**4**	**Path**
b	1	1.51	1	1.562	B = 5.072
g	5/3	1.440	4/3	1.440	G = 4.587
h	3.276	3.800	4.095	3.800	H = 90/3 = 30
p	3	4.251	2	4.251	P = 13.501
F	5.46	5.46	5.46	5.46	F = GBH = 697.93
$\hat{f}$					$\hat{f} = 5.1394$
D					D = N$\hat{f}$+P = 34.06
D_{time}				D_{time}	$= 34.06 * \tau$ = 2.635 ns

TABLE V
PATH AND DELAY CALCULATIONS FOR MSJK FLIP-FLOP

C. Schematic Design

The schematic design of the MSJK flip-flop used transistors to form the circuit. Sizes of these transistors are reflective of the numbers provided by Figure 15 and 16. The top-level design of the transistor circuit is provided in 9

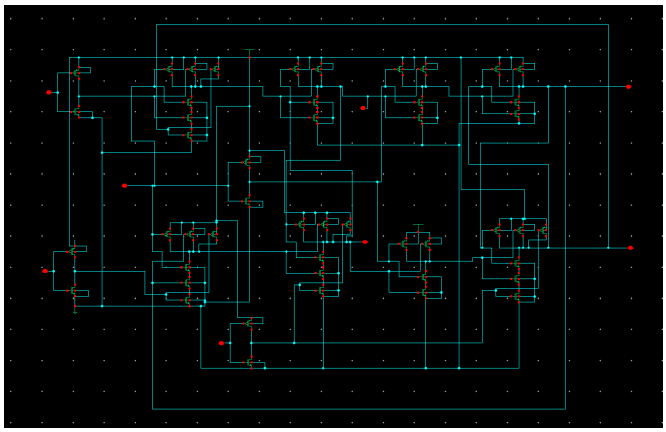


Fig. 9. Top-level transistor schematic design

The transistor design requires pMOS and nMOS components. Input and output capacitance at the beginning and output junctions were added as well to base simultaneously with the calculations provided in Table IV-B.

D. Schematic Transient Analysis

Transient analysis was performed to calculate delay and the input waveform was measured on the input pins of J,K, CLR, and CLK with driving inverters of 3C to give varying outputs to verify that a MSJK flip-flop was created within the schematic. On the output Q and Q', the load inverter of 90 was inputted and the output of the circuit was measured for comparisons.

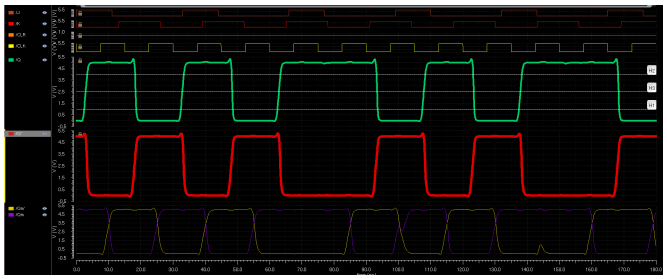


Fig. 10. Schematic simulation with measurement markers placed at Q.

The transient analysis was performed over a 4ns period clock with the correct select pins held high, with clock low (due to inverters on each input. These simulations were given a 0.1ps rise and fall time and pulse width of 2ns clock. The results of these simulations are plotted in Figures 10 and summarized in Table VI.

(ns)	**J-CLK-Q**	**K-CLK-Q'**
pdr	2.6678	3.0139
pdf	2.8476	3.2837
pd	2.7577	3.1488
tr	0.9281	0.941
tf	0.3431	0.452

TABLE VI
SUMMARIZED RESULTS OF PRE-LAYOUT SCHEMATIC SIMULATIONS

As reflective of the worst case design of Section IV-B, the worst case delay path was found from K-CLK to Q'. Here the output to bring Q' high and low is significant in the NAND3-NAND3-NAND2-NAND3 path in comparison to the NAND3-NAND2-NAND2-NAND2 path.

E. Layout Design and Standard Cell Library

With the stick diagrams created, the layouts for each stage are created. These layout designs take individual components at each stage and apply a logic package-based design to patch the Top-level design of Figure 11.

Figure 17 and 18 shows the layout of each stage path with the size of the layouts increasing from left to right simultaneously with the data path through the design. At the right end of Figure 18, the load and driving inverters are included. Using minimum design rules, the expected length

of the poly gate was calculated to 1.5μm providing a total area of 65.55 X 90.75 microns or 196.65λ by 272.25λ for the top-level design.

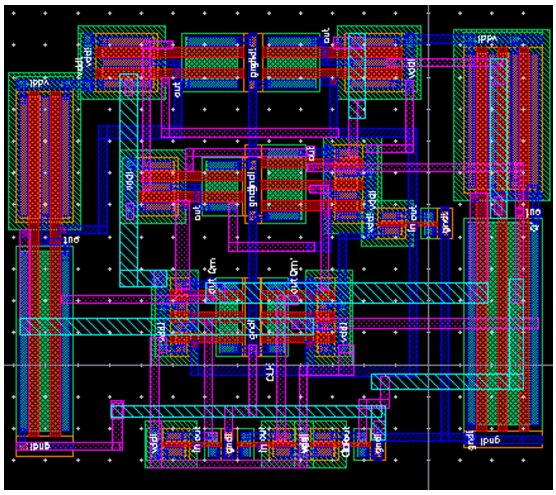


Fig. 11. Layout of top level entity

1) Top Level: The final layout is shown in figure 11 shows the driving inverters providing 3C at the bottom center of the design to travel through the center and loop back to the begging after reaching the final transistor in the stage. The top row and bottom row of the logic design are reflective of the left and right of the layout. Notably, the right side of the Layout is disproportionally larger than the left with the greater number of 3-input NAND gates. When constructing this top level design, metal 3 was avoided as much as possible with metal one being the main path of transfer. The total area of the top level layout was calculated to be $5948.66\mu m^2$.

F. Extracted View

With the top level layout complete an extracted view was generated that contained the parasitic capacitances and is shown in Figure 12. Visuals provided in Figure 12 allow for a one-to-one comparison to the layout image.

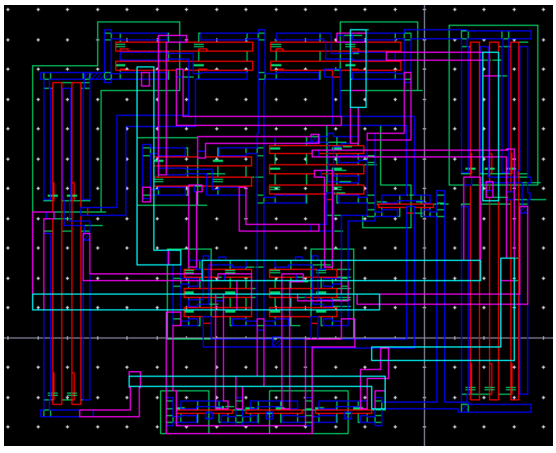


Fig. 12. Extracted view of top level entity containing transistors and parasitic capacitances

With the extracted view a post layout simulation was ran to get a more represented measure of delay that accounts for parasitic capacitances. In order to run the layout simulation under the same pretense of Figure 9 to ensure the circuit was created again, an LVS design check was used. The simulation was ran using extracted views of the MSJK layout and applied to a symbol design. Transient analysis was performed and the results are shown in Table VII. LVS matched with proof provided in Section VI-D.

G. Post-Layout Simulation

After LVS was verified and symbol correlated to the extracted view, a 90C inverter was place at the output of the design to match the schematic simulation. To perform the transient analysis to calculate delay, the input characteristics matched that of the Pre-Layout Simulations.

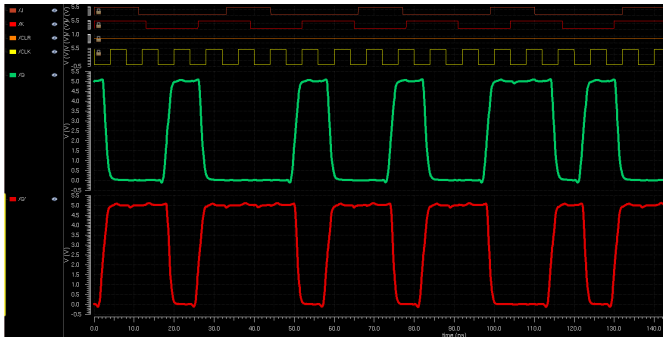


Fig. 13. Caption

As shown with the results of Figure 13, the outputs of the system are less stable then their respective schematic counterpart found with the simulation of Figure 10. This is to prove that the simulations shown are different and from the layout design. Calculations of the simulation should reflect with the Layout design differences as well in Table VII.

(ns)	J-CLK - Q	K-CLK - Q'
pdr	2.3335	2.31858
pdf	2.9282	2.80130
pd	2.63085	2.55994
tr	1.2389	1.31517
tf	0.9174	0.8305

TABLE VII
POST-LAYOUT SIMULATION RESULTS FOR MSJK FLIP-FLOP

Reflective of the table provided and detailed in Section IV-H, the layout delay characteristics improved from both the schematic and calculated designs to be reflective of the use behind gate optimization at the worst case path. Shown by an improved delay calculation at K-CLK-Q' over J-CLK-Q, the gate optimizations of each gate improved the delay along with the compacted layout and metal design choices.

H. Delay Comparison

Table VIII shows the percent difference between the post and prelayout simulations. Table VIII shows that there was

a significant change in fall time between post and prelayout simulations. The fall time increase by roughly double for the given path. Additionally it shows that the propagation delay decreased for all paths, the decrease in propagation delays is obviously due to the large decreases in rising propagation delay especially in the worst case path across the NAND3-NAND3-NAND2-NAND3. This large reduction in propagation delays is due to the combining of active regions, metal design, and gate optimization in the layout.

(%)	**J-CLK-Q**	**K-CLK-Q'**
pdr	-12.5	-23.1
pdf	2.8	-14.7
pd	-4.6	-18.7
tr	33.5	39.8
tf	167.4	83.8

TABLE VIII
PERCENT DIFFERENCES BETWEEN POST AND PRELAYOUT SIMULATIONS WITH IMPROVEMENTS DESIGNATED BY THE NEGATIVE(-)

As compared to the calculations provided in the Background of the MSJK, the Table IX displays the differences in this design.

(%)	**Lay-Q**	**Lay-Q'**	**Schem-Q**	**Schem-Q'**
pd	3.455	6.057	1.200	15.55

TABLE IX
PERCENT DIFFERENCES BETWEEN CALCULATED DELAY AND THE PRE AND POST SIMULATIONS WITH IMPROVEMENTS DESIGNATED BY THE NEGATIVE(-)

In this table, the calculations are shown to be relatively close to calculations with no improvements to the calculations with exception to the larger Schematic Q' output. Reasons to this design would likely fall to misplaced gate optimization and the limitation to optimization in schematic design.

I. Discussion

Improvements can fall with area and even gate design where the different sized paths provide uneven data transfer and difficulty in layout design. As such, in the future, a recommendation would be to add a preset to allow for even 3-input NAND gates in both rows. Additionally, if the spacing between regions were reduced or taken by a "Snap-together" design, the layout size would be reduced and easier to use. Finally, folding within the design for the final stage gates and restructuring to remove the metal-3 layer would inprove the size and delay overall. Time constraints were a holding factor to this design.

V. CONCLUSIONS

This report shows shows that for CMOS logic Master-Slave JK Flip-flop, design optimization and clock limitations can be found when pursuing the layout design process. To further this point, Cadence simulations show parasitic capacitances can have a positive effect on the delay of the design. Comparisons to the original design allow for multiple steps, following this project, to be taken to further optimize the JK flip-flop with clear.

REFERENCES

[1] Y. Dai and J. Shen, "An explicit-pulsed double-edge triggered jk flip-flop," in *2009 International Conference on Wireless Communications Signal Processing*, pp. 1–4, Nov 2009.

VI. Appendix

A. Gate Optimization Plot

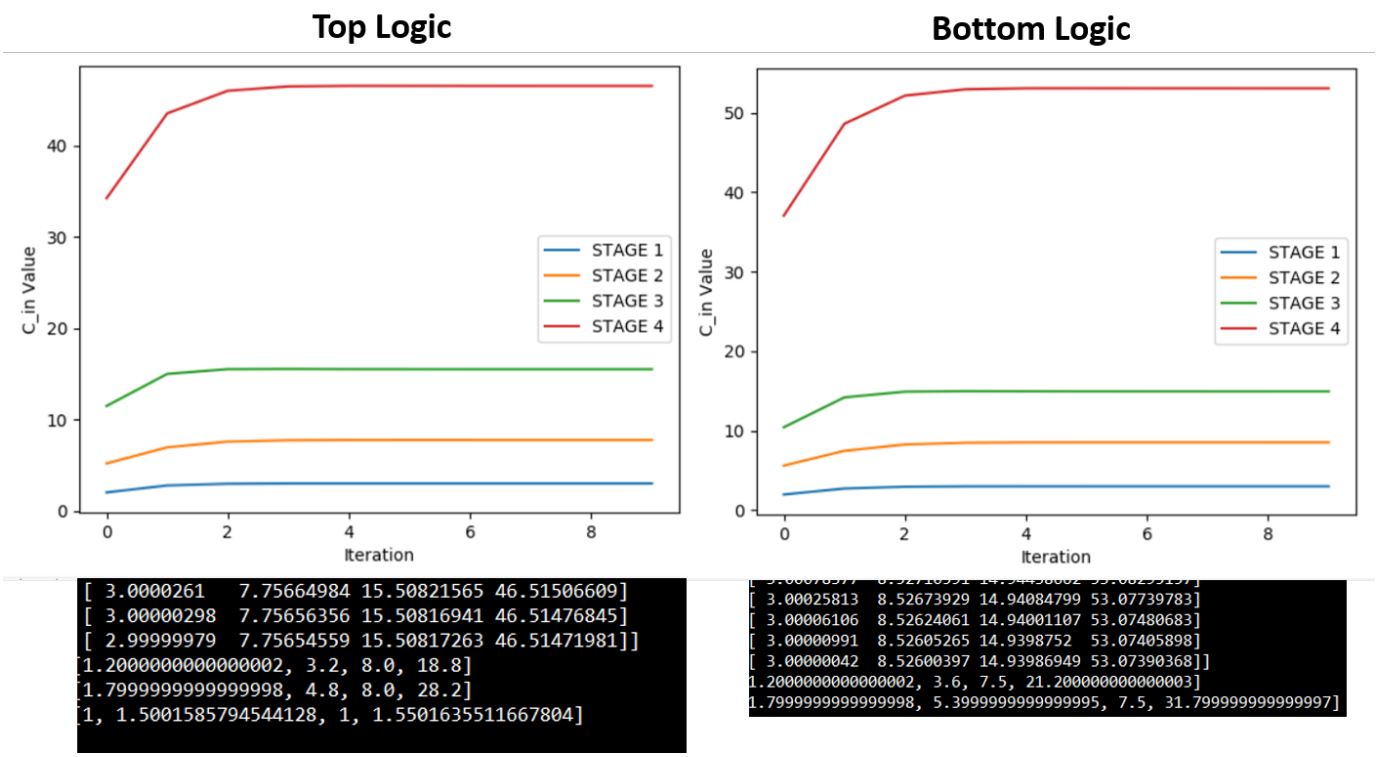
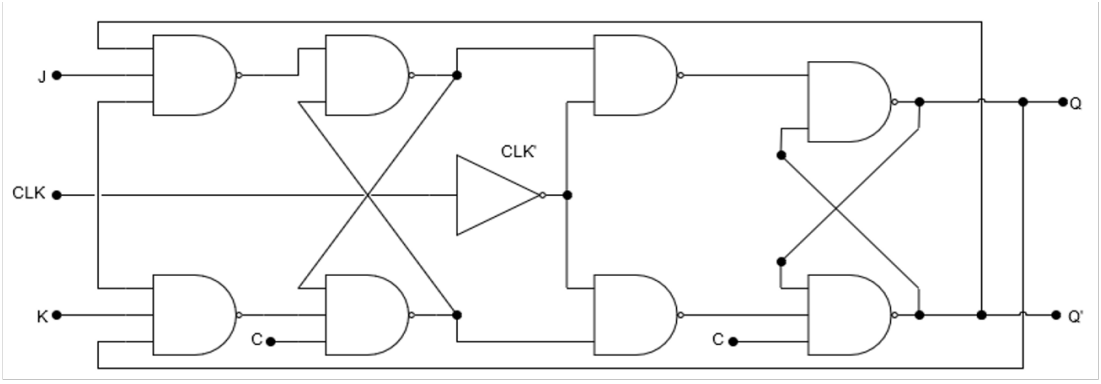


Fig. 14. Gate optimization of top and bottom logic paths used to determine gate sizes of each gate within the MSJK



Stage	Size	Width pMOS		Width nMOS		Length (μm)
R,V	3.00	1.8	1.8	2.7	1.8	1.5
S	7.757	3.2	3.3	4.8	4.8	1.5
T	15.508	8	8.1	8	8.1	1.5
U	46.515	18.8	18.9	28.2	28.2	1.5

Fig. 15. Gate sizes of top logic paths within the MSJK

Stage	Size	Width pMOS		Width nMOS		Length (μm)
R,V	3.00	1.8	1.8	2.7	1.8	1.5
W	8.52	3.6	3.6	5.4	5.4	1.5
X	14.940	7.5	7.5	7.5	7.5	1.5
Y	53.074	21.2	21.3	31.8	31.8	1.5

Fig. 16. Gate sizes of bottom logic paths within the MSJK

B. Schematic Netlist Output

```
Cadence (R) Virtuoso (R) Spectre (R) Circuit Simulator
Version 14.1.0.804.isr12 32bit -- 20 Aug 2015
Copyright (C) 1989-2015 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence, V

Includes RSA BSAFE(R) Cryptographic or Security Protocol Software from RSA Security, Inc.

User: gouldj5   Host: cadence1.rowan.edu   HostID: FA96E34B   PID: 26384
Memory  available: 2.4131 GB  physical: 8.3898 GB
CPU Type: Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
          Processor PhysicalID CoreID Frequency Load
              0         0        0     2600.0    34.2
              1         2        0     2600.0    38.5
              2         4        0     2600.0    45.7
              3         6        0     2600.0    32.5


Simulating 'input.scs' on cadence1.rowan.edu at 2:25:21 PM, Wed Dec 19, 2018 (process id: 2638
Current working directory: /home/gouldj5/cadence/simulation/Project/Overview-inv/adexl/results
Command line:
    /opt/cadence/installs/MMSIM141/tools/bin/spectre input.scs  \
        +escchars +log  \
        ../../simulation/Overview-inv/spectre/schematic-invset/psf/spectre.out  \
        +inter=mpsc +mpssession=spectre0_25407_2 -format psfxl -raw  \
        ../../simulation/Overview-inv/spectre/schematic-invset/psf  \
        +lqtimeout 900 -maxw 5 -maxn 5
spectre pid = 26384

Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libinfineon_sh.so ...
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libphilips_o_sh.so ...
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libphilips_sh.so ...
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libsparam_sh.so ...
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libstmodels_sh.so ...
Reading file:  /home/gouldj5/cadence/simulation/Project/Overview-inv/adexl/results/data/.tmpADI
Reading file:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/configs/spectre.cfg
Reading file:  /home/shin/pdk/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06N.m
Reading file:  /home/shin/pdk/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06P.m
Reading file:  /home/gouldj5/cadence/simulation/Project/Overview-inv/adexl/results/data/.tmpADI
Time for NDB Parsing: CPU = 81.987 ms, elapsed = 87.5429 ms.
Time accumulated: CPU = 114.981 ms, elapsed = 87.5499 ms.
Peak resident memory used = 29.7 Mbytes.

Reading link:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/discipline.h
```

```
Reading file:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/disciplines.vams
Reading link:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/constants.h
Reading file:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/constants.vams
Time for Elaboration: CPU = 27.996 ms, elapsed = 28.5411 ms.
Time accumulated: CPU = 142.977 ms, elapsed = 116.385 ms.
Peak resident memory used = 32.6 Mbytes.

Time for EDB Visiting: CPU = 0 s, elapsed = 1.10602 ms.
Time accumulated: CPU = 143.977 ms, elapsed = 117.8 ms.
Peak resident memory used = 33.1 Mbytes.


Global user options:
             reltol = 0.001
            vabstol = 1e-06
            iabstol = 1e-12
               temp = 27
               tnom = 27
             scalem = 1
              scale = 1
               gmin = 1e-12
             rforce = 1
           maxnotes = 5
           maxwarns = 5
             digits = 5
               cols = 80
             pivrel = 0.001
           sensfile = ../psf/sens.output
      checklimitdest = psf
               save = allpub
               tnom = 27
             scalem = 1
              scale = 1

Circuit inventory:
              nodes 29
            bsim3v3 48
            vsource 5

Analysis and control statement inventory:
               info 7
               tran 1

Output statements:
             .probe 0
           .measure 0
               save 0

Time for parsing: CPU = 3.999 ms, elapsed = 9.00817 ms.
Time accumulated: CPU = 148.976 ms, elapsed = 127.077 ms.
Peak resident memory used = 34 Mbytes.

~~~~~~~~~~~~~~~~~~~~~~
Pre-Simulation Summary
~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~
```

```
Entering remote command mode using MPSC service (spectre, ipi, v0.0, spectre0_25407_2, ).

Warning from spectre.
    WARNING (SPECTRE-16707): Only tran supports psfxl format, result of other analyses will be


************************************************
Transient Analysis `tran': time = (0 s -> 180 ns)
************************************************
Trying `homotopy = gmin' for initial conditions.
Trying `homotopy = source' for initial conditions.
Trying `homotopy = dptran' for initial conditions.

Notice from spectre during IC analysis, during transient analysis `tran'.
    GminDC = 1 pS is large enough to noticeably affect the DC solution.
        dV(net133) = 41.2354 mV
        Use the `gmin_check' option to eliminate or expand this report.
    Bad pivoting is found during DC analysis. Option dc_pivot_check=yes is recommended for pos

DC simulation time: CPU = 86.987 ms, elapsed = 87.2951 ms.
Important parameter values:
    start = 0 s
    outputstart = 0 s
    stop = 180 ns
    step = 180 ps
    maxstep = 3.6 ns
    ic = all
    useprevic = no
    skipdc = no
    reltol = 1e-03
    abstol(V) = 1 uV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    errpreset = moderate
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS


Output and IC/nodeset summary:
                save   5       (current)
                save   29      (voltage)

    tran: time = 4.751 ns    (2.64 %), step = 268.3 ps     (149 m%)
    tran: time = 13.54 ns    (7.52 %), step = 67.8 ps     (37.7 m%)
    tran: time = 22.5 ns     (12.5 %), step = 578.3 ps     (321 m%)
    tran: time = 31.52 ns    (17.5 %), step = 185.2 ps     (103 m%)
    tran: time = 40.54 ns    (22.5 %), step = 154.2 ps    (85.6 m%)
    tran: time = 49.73 ns    (27.6 %), step = 256.6 ps     (143 m%)
    tran: time = 59.03 ns    (32.8 %), step = 638.5 ps     (355 m%)
    tran: time = 67.5 ns     (37.5 %), step = 168.3 ps    (93.5 m%)
    tran: time = 76.62 ns    (42.6 %), step = 260.6 ps     (145 m%)
```

```
    tran: time = 85.55 ns    (47.5 %), step = 166.1 ps    (92.3 m%)
    tran: time = 94.61 ns    (52.6 %), step = 230.3 ps     (128 m%)
    tran: time = 103.7 ns    (57.6 %), step = 271.1 ps     (151 m%)
    tran: time = 112.5 ns    (62.5 %), step = 451.9 ps     (251 m%)
    tran: time = 121.5 ns    (67.5 %), step = 267 ps       (148 m%)
    tran: time = 130.5 ns    (72.5 %), step = 42.55 ps    (23.6 m%)
    tran: time = 139.6 ns    (77.5 %), step = 229 ps       (127 m%)
    tran: time = 148.9 ns    (82.7 %), step = 737.7 ps     (410 m%)

Notice from spectre at time = 150 ns during transient analysis 'tran'.
    Found trapezoidal ringing on node net0133.

    tran: time = 157.5 ns    (87.5 %), step = 124.2 ps      (69 m%)
    tran: time = 166.6 ns    (92.5 %), step = 190 ps       (106 m%)
    tran: time = 175.7 ns    (97.6 %), step = 313 ps       (174 m%)
Number of accepted tran steps =             2322

Notice from spectre during transient analysis 'tran'.
    Trapezoidal ringing is detected during tran analysis.
        Please use method=trap for better results and performance.


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Post-Transient Simulation Summary
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   -   To further speed up simulation, consider
          add ++aps on command line
          add +cktpreset=sampled on command line for ADC/DAC simulation
          add +cktpreset=pll on command line for PLL simulation
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Initial condition solution time: CPU = 86.987 ms, elapsed = 87.405 ms.
Intrinsic tran analysis time:    CPU = 743.887 ms, elapsed = 751.386 ms.
Total time required for tran analysis 'tran': CPU = 833.874 ms, elapsed = 841.969 ms.
Time accumulated: CPU = 990.849 ms, elapsed = 1.34327 s.
Peak resident memory used = 37.6 Mbytes.

finalTimeOP: writing operating point information to rawfile.
modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.
```

C. Standard Cell Library for Top and Bottom Paths

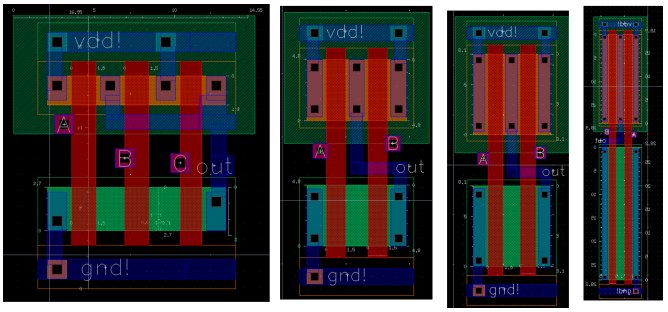


Fig. 17. Layout of Upper-Level Path of Design

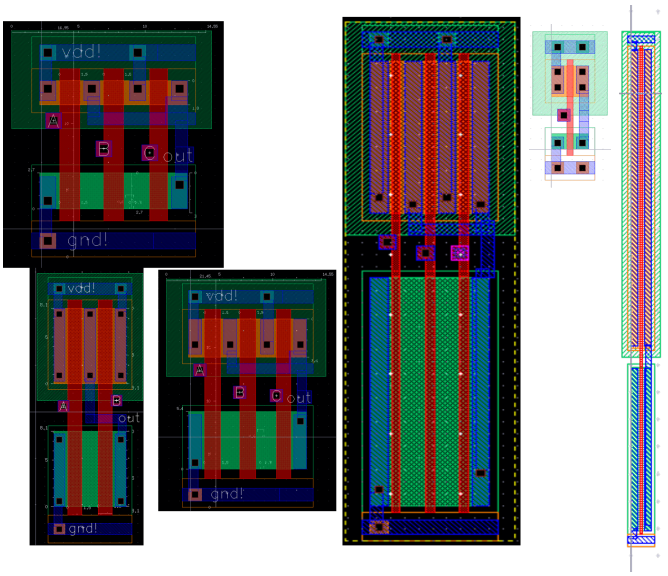


Fig. 18. Layout of Lower-Level Path of Design with Inverter Design

D. LVS Design Rule Match Output

```
      @(#)$CDS: LVS version 6.1.6 09/01/2015 15:36 (sjfnl125) $

Command line: /opt/cadence/installs/IC616/tools.lnx86/dfII/bin/32bit/LVS -dir /home/gouldj5/ca
Like matching is enabled.
Net swapping is enabled.
Using terminal names as correspondence points.
Compiling Diva LVS rules...

    Net-list summary for /home/gouldj5/cadence/LVS/layout/netlist
       count
        30              nets
        10              terminals
        24              pmos
        24              nmos

    Net-list summary for /home/gouldj5/cadence/LVS/schematic/netlist
       count
        30              nets
        10              terminals
```

```
        24              pmos
        24              nmos


    Terminal correspondence points
    N26       N4        CLK
    N22       N17       CLR
    N25       N3        J
    N23       N15       K
    N21       N5        Q
    N24       N19       Q'
    N28       N29       Qm
    N29       N30       Qm'
    N20       N1        gnd!
    N27       N0        vdd!

Devices in the rules but not in the netlist:
        cap nfet pfet nmos4 pmos4

The net-lists match.

                             layout  schematic
                                instances
        un-matched              0       0
        rewired                 0       0
        size errors             0       0
        pruned                  0       0
        active                  48      48
        total                   48      48

                                  nets
        un-matched              0       0
        merged                  0       0
        pruned                  0       0
        active                  30      30
        total                   30      30

                                terminals
        un-matched              0       0
        matched but
        different type          0       0
        total                   10      10


Probe files from /home/gouldj5/cadence/LVS/schematic

devbad.out:

netbad.out:

mergenet.out:

termbad.out:

prunenet.out:
```

```
prunedev.out:

audit.out:


Probe files from /home/gouldj5/cadence/LVS/layout

devbad.out:

netbad.out:

mergenet.out:

termbad.out:

prunenet.out:

prunedev.out:

audit.out:
```

E. Post-layout Netlist

```
Cadence (R) Virtuoso (R) Spectre (R) Circuit Simulator
Version 14.1.0.804.isr12 32bit -- 20 Aug 2015
Copyright (C) 1989-2015 Cadence Design Systems, Inc. All rights reserved worldwide. Cadence, V

Includes RSA BSAFE(R) Cryptographic or Security Protocol Software from RSA Security, Inc.

User: gouldj5   Host: cadence1.rowan.edu   HostID: FA96E34B   PID: 11742
Memory  available: 2.2451 GB  physical: 8.3898 GB
CPU Type: Intel(R) Xeon(R) CPU E5-2670 0 @ 2.60GHz
          Processor PhysicalID CoreID Frequency Load
              0         0        0     2600.0    34.3
              1         2        0     2600.0    38.5
              2         4        0     2600.0    45.7
              3         6        0     2600.0    32.5


Simulating 'input.scs' on cadence1.rowan.edu at 3:49:00 PM, Wed Dec 19, 2018 (process id: 1174
Current working directory: /home/gouldj5/cadence/simulation/Project/sim_Overview/adexl/results
Command line:
    /opt/cadence/installs/MMSIM141/tools/bin/spectre input.scs  \
        +escchars +log  \
        ../../simulation/sim_Overview/spectre/config/psf/spectre.out  \
        +inter=mpsc +mpssession=spectre5_25407_7 -format psfxl -raw  \
        ../../simulation/sim_Overview/spectre/config/psf +lqtimeout 900  \
        -maxw 5 -maxn 5
spectre pid = 11742

Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libinfineon_sh.so ...
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libphilips_o_sh.so ...
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libphilips_sh.so ...
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libsparam_sh.so ...
```

```
Loading /opt/cadence/installs/MMSIM141/tools.lnx86/cmi/lib/5.0/libstmodels_sh.so ...
Reading file:  /home/gouldj5/cadence/simulation/Project/sim_Overview/adexl/results/data/.tmpADI
Reading file:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/configs/spectre.cfg
Reading file:  /home/shin/pdk/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06N.m
Reading file:  /home/shin/pdk/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06P.m
Reading file:  /home/gouldj5/cadence/simulation/Project/sim_Overview/adexl/results/data/.tmpADI
Time for NDB Parsing: CPU = 73.989 ms, elapsed = 79.1401 ms.
Time accumulated: CPU = 94.984 ms, elapsed = 79.1461 ms.
Peak resident memory used = 29.7 Mbytes.

Reading link:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/discipline.h
Reading file:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/disciplines.vams
Reading link:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/constants.h
Reading file:  /opt/cadence/installs/MMSIM141/tools.lnx86/spectre/etc/ahdl/constants.vams
Time for Elaboration: CPU = 25.996 ms, elapsed = 26.2091 ms.
Time accumulated: CPU = 120.98 ms, elapsed = 105.64 ms.
Peak resident memory used = 32.7 Mbytes.

Time for EDB Visiting: CPU = 0 s, elapsed = 1.08695 ms.
Time accumulated: CPU = 121.98 ms, elapsed = 106.992 ms.
Peak resident memory used = 33.2 Mbytes.


Global user options:
             reltol = 0.001
            vabstol = 1e-06
            iabstol = 1e-12
               temp = 27
               tnom = 27
             scalem = 1
              scale = 1
               gmin = 1e-12
             rforce = 1
           maxnotes = 5
           maxwarns = 5
             digits = 5
               cols = 80
             pivrel = 0.001
           sensfile = ../psf/sens.output
     checklimitdest = psf
               save = allpub
               tnom = 27
             scalem = 1
              scale = 1

Circuit inventory:
              nodes 29
            bsim3v3 48
            vsource 5

Analysis and control statement inventory:
               info 7
               tran 1

Output statements:
             .probe 0
```

```
            .measure 0
                save 0

Time for parsing: CPU = 3 ms, elapsed = 8.22783 ms.
Time accumulated: CPU = 125.98 ms, elapsed = 115.489 ms.
Peak resident memory used = 34.1 Mbytes.

~~~~~~~~~~~~~~~~~~~~~~
Pre-Simulation Summary
~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~
Entering remote command mode using MPSC service (spectre, ipi, v0.0, spectre5_25407_7, ).

Warning from spectre.
    WARNING (SPECTRE-16707): Only tran supports psfxl format, result of other analyses will be


************************************************
Transient Analysis `tran': time = (0 s -> 180 ns)
************************************************
Trying `homotopy = gmin' for initial conditions.
Trying `homotopy = source' for initial conditions.

Notice from spectre during IC analysis, during transient analysis `tran'.
    GminDC = 1 pS is large enough to noticeably affect the DC solution.
        dV(I0.29) = 47.8729 mV
        Use the `gmin_check' option to eliminate or expand this report.

DC simulation time: CPU = 63.99 ms, elapsed = 63.9348 ms.
Important parameter values:
    start = 0 s
    outputstart = 0 s
    stop = 180 ns
    step = 180 ps
    maxstep = 3.6 ns
    ic = all
    useprevic = no
    skipdc = no
    reltol = 1e-03
    abstol(V) = 1 uV
    abstol(I) = 1 pA
    temp = 27 C
    tnom = 27 C
    tempeffects = all
    errpreset = moderate
    method = traponly
    lteratio = 3.5
    relref = sigglobal
    cmin = 0 F
    gmin = 1 pS


Output and IC/nodeset summary:
                save   5       (current)
                save   29      (voltage)
```

```
    tran: time = 4.564 ns    (2.54 %), step = 89.43 ps    (49.7 m%)
    tran: time = 13.52 ns    (7.51 %), step = 25.08 ps    (13.9 m%)

Notice from spectre at time = 22.3321 ns during transient analysis 'tran'.
    Found trapezoidal ringing on node I0.29.

    tran: time = 22.55 ns    (12.5 %), step = 221.9 ps     (123 m%)
    tran: time = 31.51 ns    (17.5 %), step = 245.3 ps     (136 m%)
    tran: time = 40.57 ns    (22.5 %), step = 78.23 ps    (43.5 m%)
    tran: time = 49.61 ns    (27.6 %), step = 127.4 ps    (70.8 m%)
    tran: time = 58.51 ns    (32.5 %), step = 67.87 ps    (37.7 m%)
    tran: time = 67.64 ns    (37.6 %), step = 238.4 ps     (132 m%)
    tran: time = 76.52 ns    (42.5 %), step = 57.14 ps    (31.7 m%)
    tran: time = 85.53 ns    (47.5 %), step = 31.21 ps    (17.3 m%)
    tran: time = 94.51 ns    (52.5 %), step = 212.1 ps     (118 m%)
    tran: time = 103.6 ns    (57.6 %), step = 371.4 ps     (206 m%)
    tran: time = 112.5 ns    (62.5 %), step = 52.25 ps      (29 m%)
    tran: time = 121.6 ns    (67.5 %), step = 125.6 ps    (69.8 m%)
    tran: time = 130.6 ns    (72.5 %), step = 73.34 ps    (40.7 m%)
    tran: time = 139.7 ns    (77.6 %), step = 249.7 ps     (139 m%)
    tran: time = 148.5 ns    (82.5 %), step = 58.16 ps    (32.3 m%)
    tran: time = 157.6 ns    (87.6 %), step = 129.7 ps    (72.1 m%)
    tran: time = 166.6 ns    (92.5 %), step = 121.9 ps    (67.7 m%)
    tran: time = 175.6 ns    (97.6 %), step = 300.7 ps     (167 m%)
Number of accepted tran steps =             2919

Notice from spectre during transient analysis 'tran'.
    Trapezoidal ringing is detected during tran analysis.
        Please use method=trap for better results and performance.


~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Post-Transient Simulation Summary
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
   -   To further speed up simulation, consider
          add ++aps on command line
          add +cktpreset=sampled on command line for ADC/DAC simulation
          add +cktpreset=pll on command line for PLL simulation
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Initial condition solution time: CPU = 63.99 ms, elapsed = 64.0781 ms.
Intrinsic tran analysis time:    CPU = 951.855 ms, elapsed = 958.813 ms.
Total time required for tran analysis 'tran': CPU = 1.01784 s, elapsed = 1.0259 s.
Time accumulated: CPU = 1.15282 s, elapsed = 1.43831 s.
Peak resident memory used = 37.9 Mbytes.

finalTimeOP: writing operating point information to rawfile.
modelParameter: writing model parameter values to rawfile.
element: writing instance parameter values to rawfile.
outputParameter: writing output parameter values to rawfile.
designParamVals: writing netlist parameters to rawfile.
primitives: writing primitives to rawfile.
subckts: writing subcircuits to rawfile.
```