VLSI Final Project: J-K Flip-Flop w/ Clear

Stephen Glass
ECE 09.414-2
December 19th, 2018
Email: glasss6@students.rowan.edu

***Abstract*—A master-slave J-K Flip-Flop with clear functionality is implemented in CMOS technology using VLSI design process. The truth table is analyzed and minimized. All preliminary logic and circuits are analyzed and discussed. Logical effort is used to estimate least delay of the circuit, number of stages, and the sizes of the gates. A schematic design is developed using Cadence and simulated for delay. A layout design for the J-K Flip-Flop is developed. The layout satisfies design rule check (DRC) and Layout vs. Schematic (LVS). The delay of the layout and schematic are compared to the least delay computed by logical effort. The delay of the schematic had a 9.0% error compared to the logical effort. The delay of the layout had a 3.0% error compared to the logical effort. It is determined the J-K Flip-Flop design is accurate and that logical effort analysis can provide good approximation for delay.**

I. Introduction and Objectives

Complementary MOSFET (CMOS) technology is widely used today to form circuits in numerous applications. CMOS offers low power dissipation, high switching speed and operate over a wide range of input voltages [1]. This report involves the full design process and simulation of a J-K Flip-Flop using a CMOS implementation. Designing a JK-Flip-Flop using CMOS technology allows an engineer to realize the power and speed advantages. The CMOS design process involves logic design, simulations and layout design.

The MOSIS Scalable CMOS (SCMOS) design rules are used in this report to design a CMOS layout for fabrication using software. SCMOS is a set of logical layers and design rules which provide an interface to many CMOS fabrication processes [2]. A designer works in the abstract SCMOS layers and metric unit ("lambda"). Various fabrication processes specify feature size, design rules, and lambda.

A CMOS circuit can be converted to a layout design by first drawing a circuit diagram and stick diagram. The propagation delay of a CMOS circuit can be estimated using logical effort and compared to the propagation delay of schematic diagrams and layout design with added parasitic capacitance using software tools. The propagation delay calculated by logical effort is expected to be a close approximation to the delay found in simulation.

II. Background and Relevant Theory

A. CMOS technology

A CMOS circuit is composed of both types of MOSFETs: a PMOS type and a NMOS type. When two switches are connected in series, the resulting circuit conducts only if both switches conduct. Similarly, when two switches are connected in parallel, the resulting circuit conducts if either of the two switches conducts [3]. These circuits are implementations of the logical AND/OR operations. A logical function can be implemented with a CMOS circuit by utilized a pull-up circuit with PMOS transistors connected to a voltage source V_{DD} for a function f and a complementary pull-down circuit with NMOS transistors connected to ground for a function $\overline{f}$.

A CMOS inverter can be implemented where the body of each device is directly connected to the device's source, the gate of each device is connected to the input, and the drain of both devices are connected to the output. The logical expression for a inverter can be simply expressed as $F = \overline{A}$ where the logical input A will be inverted. In VLSI design, a CMOS inverter with a gate size of ratio of 2:1 is called the "unit inverter." A typical unit inverter is found to have a average propagation delay of approximately $64ps$. The properties of a unit inverter are used as a reference to calculate effective resistances and propagation delays using logical effort.

B. Logical effort

Logical effort is the ratio of the input capacitance of a gate to the input capacitance of a inverter delivering the same output current [4]. The method of logical effort can be used to:

1) Compute path effort
2) Estimate the best number of stages
3) Estimate least delay
4) Determine best stage effort
5) Find gate sizes

However, there are some limits of logical effort. It is a simplistic delay model which neglects input rise time effects. Logical effort also only considers the maximum speed. Logical effort is useful for approximating delay in CMOS circuits. A table of definitions to calculate logical effort is shown in Table 1 below.

TABLE I

Term	Stage	Path
number of stages	1	N
logical effort	g	$G = \Pi g_i$
electrical effort	$h = \frac{C_{out}}{C_{in}}$	$H = \frac{C_{out-path}}{C_{in-path}}$
branching effort	$b = \frac{C_{on-path}+C_{off-path}}{C_{on-path}}$	$B = \Pi b_i$
effort	$f = gh$	$F = GBH$
effort delay	f	$D_F = \Sigma p_i$
parasitic delay	p	$P = \Sigma p_i$
delay	$d = f + p$	$D = \Sigma d_i = D_F + P$

Effort delay f is stage effort and is the product of logical effort and electrical effort. Logical effort g measures the relative ability of a gate to deliver some specified current. Electrical effort h is the ratio of output capacitance to input capacitance (fanout). Parasitic delay p represents the delay of a gate driving no load and is set by internal parasitic capacitance.

The logical effort of an inverter is 1 and the logical effort of a n-input NAND gate can be calculated by $(n+2)/3$. Using logical effort, it is possible to estimate the best number of stages and find the gate sizes. These are crucial steps for designing a CMOS circuit with the least delay. The equations used to find crucial parameters for designing a CMOS circuit is listed in Table 2 below.

TABLE II

Term	**Equation**
Compute path effort	$F = GBH$
Estimate best number of stages	$\hat{N} = \log_4 F$
Estimate least delay	$D = \hat{N}F^{1/\hat{N}} + P$
Determine best stage effort	$\hat{f} = F^{1/N}$
Find gate sizes	$C_{in,i} = (g_i C_{out,i})/\hat{f}$

The delay that is provided from D is in reference to the unit inverter. To find the actual delay of a CMOS circuit, it must be normalized to the unit inverter output capacitance then multiplied by the inverter delay.

C. Transient analysis

Transient analysis shows the response of a signal over a period of time. Performing transient analysis can show circuit behavior and non-linearity of a signal. Transient analysis of a signal may provide rise time, fall time and propagation delays. These delays are important in assessing the performance of some circuit designs.

Rise time refers to the time it takes for a signal to fall from 80% to 80% of its minimum and steady-state value. Likewise, fall time refers to the time it takes for a signal to fall from 80% to 20% of its steady-state and minimum value. Rise time and fall time values primarily change due to capacitance.

Rising propagation delay is the input-to-output delay for low-to-high output transition. The rising propagation is the difference from the falling input crossing 50% steady-state to the rising output crossing 50% steady-state. Similarly, falling propagation delay is the time difference from the rising input crossing 50% steady-state to the falling output crossing 50%. The average propagation delay is found by averaging the falling and rising propagation delays.

D. CMOS layout design

The layout design is the diagram of a CMOS circuit to be fabricated. In layout design, the unit λ is used to refer to half technology [feature] size and is a standard unit for determining dimensions of layers in compliance with design rules. The unit lambda can be easily scaled to the different fabrication processes.

Design rules refer to the lambda unit of measurement for device mask dimensions, transistor spacing, metal spacing, etc. Layout design rules describe how small features can be and how closely they can be reliably packed in a particular manufacturing process [5]. A layout designer must follow design rules to allow successful manufacturing. The SCMOS design rules for layouts with two metal layers in an n-well are:

- Metal and diffusion have minimum width and spacing of 4λ
- Contacts are $2\lambda \times 2\lambda$ and must be surrounded by 1λ on the layers above and below
- Poly-silicon uses a width of 2λ
- Poly-silicon overlaps diffusion by 2λ where a transistor is desired and has spacing of 1λ away where no transistor is desired
- Poly-silicon and contacts have spacing of 3λ from other poly-silicon or contacts
- N-well surrounds pMOS transistors by 6λ and avoids nMOS transistors by 6λ.

Stick diagrams can be used to help a designer plan a layout design for a CMOS circuit. A stick diagram is a simple cartoon drawing which shows the topography and layer information of a circuit. Stick diagrams layout components, relative placement, routing, and connections between metal contacts and diffusion areas. A stick diagram is typically a good first step before attempting a layout design.

The design rule check tool (DRC) is used to analyze a layout and check if it is satisfies the design rules to ensure a layout can be fabricated. The layout versus schematic tool (LVS) allows the netlist of a netlist design to be compared against the netlist of a schematic design. If the two netlists do not match, then the layout-simulation results will not be accurate because it will not be an equivalent comparison.

E. J-K Flip-Flop

A J-K Flip-Flop is a two-input flip-flop where the output Q takes value of J at the next clock edge if J and K are different. No change will occur if J and K are both low. The output toggles stages if J and K are both high. This implementation of a J-K Flip-Flop will also contain a clear input-pin to clear the current state. A block diagram of a J-K Flip-Flop with clear functionality, circuit diagram, and a corresponding truth table is seen in Figure 1-2 and Table 3 below.

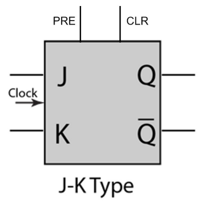


Fig. 1. Block diagram of a J-K Flip-Flop

While this implementation of the J-K Flip-Flop shown in Figure 2 works in principle, there are problems that arise with the timing. Short clock pulses can easily drive the circuit into

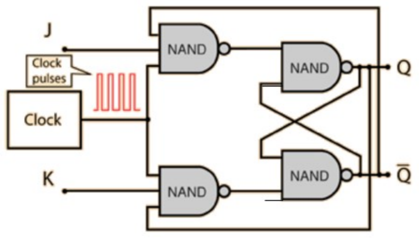


Fig. 2. Circuit implementation of a simple J-K Flip-Flop

TABLE III

CLK	J	K	Q	Q’	Qn	Qn’	State
0	X	X	X	X	X	X	Latched
1	0	0	0	1	0	1	No Change
1	0	0	1	0	1	0	No Change
1	0	1	0	1	0	1	Reset
1	0	1	1	0	0	1	Reset
1	1	0	0	1	1	0	Set
1	1	0	1	0	1	0	Set
1	1	1	0	1	1	0	Toggle
1	1	1	1	0	0	1	Toggle

oscillation. Modern ICs are so fast where the simple J-K flip-flop circuit is not practical as oscillation will occur [6]. In this case, the master-slave J-K Flip-Flop circuit design is used to fix the oscillation of the simple implementation. The circuit implementation of master-slave J-K Flip-Flop is seen in Figure 3 below.

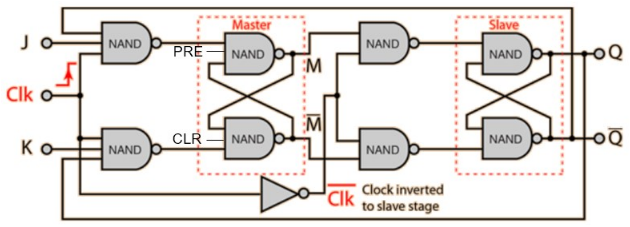


Fig. 3. Circuit implementation of a master-slave J-K Flip-Flop

The master-slave J-K Flip-Flop has two J-K Flip-Flops used as latches where the second is driven by an inverted clock signal. The addition of the second latch suppresses the oscillation behavior of the previous implementation.

III. Procedure

The logical design of a J-K Flip-Flop is first analyzed. K-map minimization is used to find a logical expression of the circuit. Various circuit implementations are compared. The logical effort of the chosen circuit implementation is found.

Following the logical effort process, the path effort is computed. The best number of stages of estimated. The least delay is estimated. The best stage effort is determined. And using backwards solving, the appropriate gate sizes are calculated with gate size ratios for equivalent effective resistance.

For design analysis, an inverter which presents a load of 3 units of transistor width on its input is assumed. Also, an output load equivalent to 90 units of transistor width is assumed. The overall input and output specifications of the project design is shown in Figure 4 below.

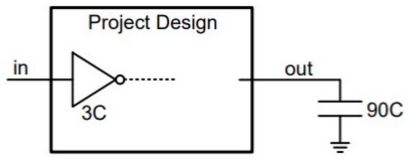


Fig. 4. Input-Output specifications of project design

The J-K flip-flop is implemented as a schematic using Cadence design. Transient (Spectre) simulations are performed to estimate the delay and verify correct circuit functionality.

Layouts are designed for each gate used in the J-K flip-flop schematic. The layouts are extracted to analyze parasitic capacitance. The layout designs are implemented in a overall layout design. The Design rule check (DRC) and Layout vs. Schematic (LVS) tool are used to verify the layout can be fabricated without error and corresponds to the schematic netlist. The overall layout is extracted and transient (Spectre) simulations are performed to estimate the delay and verify correct circuit functionality. The circuit delay of the layout and schematic simulations is compared against the estimated value from logical effort analysis.

IV. Results and Discussion

A. Truth table analysis

The J-K Flip-Flop from Table 3 is analyzed using a K-map to find the optimized logical equation. The K-map is shown in Figure 5 below.

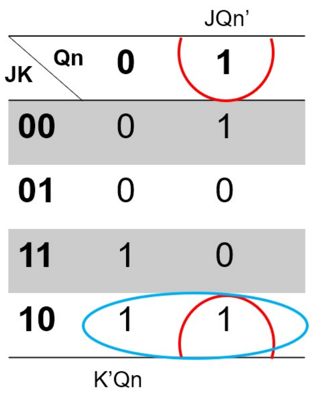

JQn’
JK
Qn
0
1
00
0
1
01
0
0
11
1
0
10
1
1
K’Qn

Fig. 5. K-map of J-K Flip-Flop truth table

The optimized logical equation for the J-K Flip-Flop is determined to be $Q_n = (\overline{K}Q_n + J\overline{Q_n}) \cdot CLK$ by the Kmap optimization. A circuit implementation of this logical equation is shown in Figure 6 below.

However, the circuit implementation found shown in Figure 6 is not optimized for minimum delay using AND/NOR gates. Additionally, it also contains the oscillation issue for a simple J-K Flip-Flop implementation as discussed in Background and Theory.

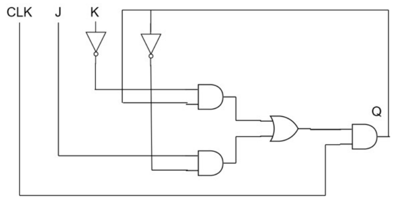


Fig. 6. Circuit implementation of the Kmap logical equation

B. Logic design

The master-slave J-K Flip-Flop design as discussed in Background and Theory section and shown in Figure 3 is chosen as the final design. This is because the master-slave configuration is the only practical solution as oscillation will not occur with short clock pulses. A preset input is added in addition with the clear input as to provide gate symmetry for easier logical effort calculations. The logical effort analysis of the master-slave J-K Flip-Flop implementation is shown in Table 4.

TABLE IV

N	**1**	**2**	**3**	**4**	**5**
p	1	3	3	2	2
g	1	5/3	5/3	4/3	4/3
h	w/3C	x/w	y/x	z/y	90/z
b	1	1	9/4	1	99/90

Recall the equations for calculating logical effort, electrical effort, and branching effort from Background and Theory section. The parasitic delay of each stage can be found by the number of gate input-pins n. The total parasitic delay can be found by summing each stage. The logical effort and branching effort can be found by multiplying each stage. The total electrical effort is found by dividing the total output capacitance by the total input capacitance. A summary of these results are shown in Table 5.

TABLE V

Parameter	**Value**
P	11
G	400/81
B	99/40
H	30

Recall the equations for computing path effort, estimating best number of stages, estimating least delay, determining best stage effort, and final gate sizes from Background and Theory section. A summary of these results are shown in Table 6.

TABLE VI

Parameter	**Value**
F	366.667
$\hat{N}$	4.25
D_4	28.5036
D_5	27.2864
$\hat{f}$	3.25727

It can be seen from Table 6 that the delay of a circuit with a path effort delay of 366.667 is fastest with five stages rather than four. The least delay for a five stage design is 27.2867 unit inverter delay. Dividing the result by 3C (unit inverter load capacitance) then multiplying the unit inverter delay of $64ps$ results in a least circuit delay of approximately $582.11ps$. The calculated best stage effort is found to be 3.25727. Using this information, it is possible to calculate the gate sizes for each stage. The CMOS circuit diagrams for a two-input and three-input NAND gate are shown in Figure 7.

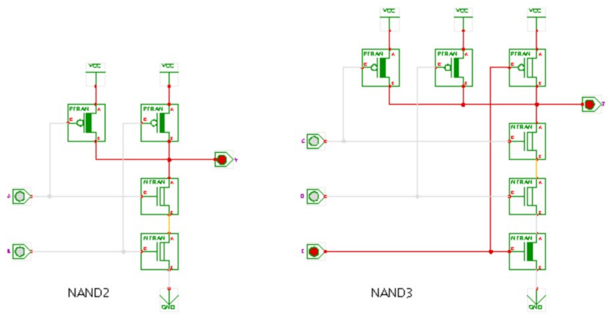


Fig. 7. CMOS circuit of NAND2 and NAND3 gates

The effective resistances of the circuit need be calculated such that the rise and fall delays are equivalent for the worst case charging and discharging scenarios. In the worst case charging scenario for a NAND2 either the PMOS or NMOS or exclusively on such that $(A, B) = (0, 1), (1, 0)$. In the worst case discharging scenario both NMOS are active such that $(A, B) = (1, 1)$ and can be modeled as two resistors in series. Therefore, the effective resistance of the PMOS R_p is equal to $2R/K_p$ and the effective resistance of the NMOS R_n is equal to R/K_n where K is the width of the device. To accomplish equivalent resistance, the NMOS device widths must be scaled by a factor of two so $R_p = 2R_n = 2R/K_n = 2R/K_p$. The gate ratio of PMOS to NMOS in terms of the unit inverter is finally calculated to be 2:2 for a NAND2 gate.

The worst case charging scenario for a NAND3 gate is when $(A, B, C) = (1, 1, 0)$. The worst case discharging scenario all three NMOS are active such that $(A, B, C) = (1, 1, 1)$. To accomplish equivalent resistance, the NMOS device widths must be scaled by a factor of three. The gate ratio of PMOS to NMOS in terms of the unit inverter is calculated to be 2:3 for a NAND3 gate.

Using the values from Table 4 and the equation $C_{in,i} = (g_i C_{out,i})/\hat{f}$ it is possible to backwards solve to find gate sizes. The gate sizes w, x, y, and z correspond to the gate sizes of the first, second, third, and fourth stage respectively. The equations below are used to find the gate sizes of each stage.

$$\frac{4}{3}\cdot\frac{90}{z}=\hat{f} \tag{1}$$

$$\frac{4}{3}\cdot\frac{z}{y}=\hat{f} \tag{2}$$

$$\frac{5}{3}\cdot\frac{y}{x}=\hat{f} \tag{3}$$

$$\frac{5}{3}\cdot\frac{x}{w}=\hat{f} \tag{4}$$

$$\frac{w}{3C}\cdot 1=\hat{f} \tag{5}$$

The calculated gate sizes using the equations above as well as the calculated sizes for each NMOS and PMOS device for the NAND2 and NAND3 gates are shown in Table 7 below.

TABLE VII

Gate	Calculated Size	Gate Ratio	PMOS	NMOS
w (NAND3)	3.94823	2:3	2.368um	4.553um
x (NAND3)	7.71627	2:3	4.629um	6.944um
y (NAND2)	15.0804	2:2	11.31um	11.31um
z (NAND2)	36.8407	2:2	27.63um	27.63um

C. Schematic design

The master-slave J-K Flip-Flop circuit is constructed using Cadence software. The appropriate gate sizes as calculated in Table 7 are selected. The overall top-level schematic of the J-K Flip-Flop is seen in Figure 8. The low-level transistor schematic of the J-K Flip-Flop is seen in Figure 9.

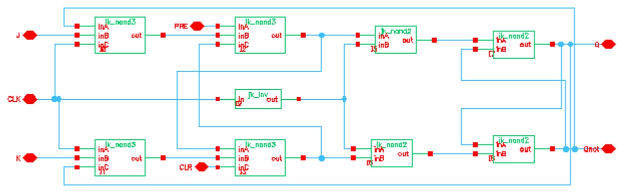


Fig. 8. Top-level schematic of master-slave J-K Flip-Flop

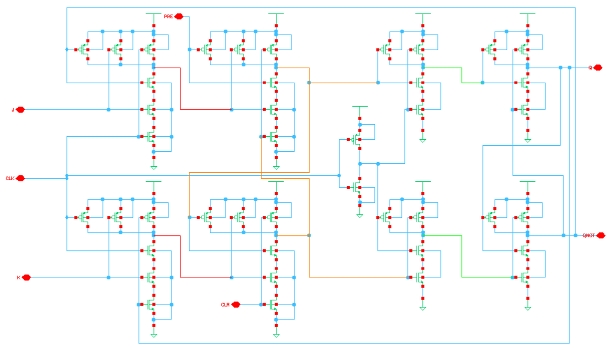


Fig. 9. Transistor-level schematic of master-slave J-K Flip-Flop

Spectre simulation is performed on the schematic design to obtain the input and output waveforms of the design and extract the propagation delay. An inverter with 3C capacitance and an output load of 90C is added to the final circuit. The simulation input and output waveforms are seen in Figure 10.

The inputs are varied as to see every combination and verify functionality of the circuit. Input J pulses with a period of 4ns and a pulse width of 2.5ns. Input K pulses with a period of 8ns and a pulse width of 4ns. The clock input pulses with a period of 2ns and a pulse width of 1ns.

The falling propagation delay of the schematic design is found to be 894.12ps. The rising propagation delay is found to be 164.25ps. The average propagation delay is calculated to be 529.19ps. Additionally, the functionality of the J-K Flip-Flop is verified by observation of inputs and outputs.

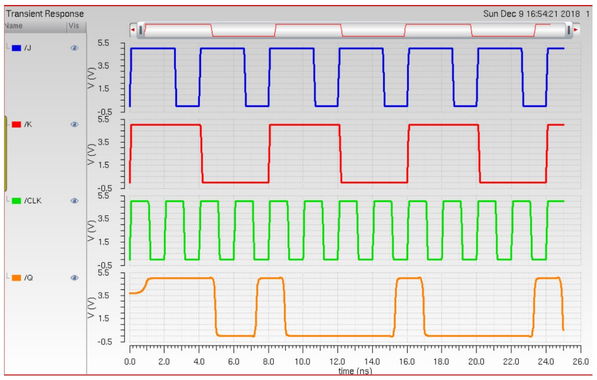


Fig. 10. Input and output simulation waveforms for schematic design

D. Layout design

As discussed in Background and Theory section, a recommended first step before attempting a layout design is to draw a stick diagram. In Figure 10 below the stick diagram of a inverter (NOT gate) is shown. In Figure 11 the stick diagrams for a NAND2 and NAND3 gate are shown.

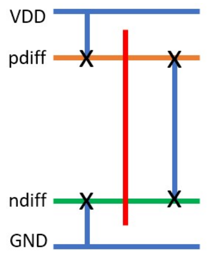


Fig. 11. Stick diagram of an inverter (NOT gate)

The stick diagrams show the topography, relative placement, and connections between metal contacts and diffusion areas. The layout design in this report uses 0.6um poly-gate CMOS lambda-based design rules. The layout design for a CMOS inverter is shown in Figure 13.

The layout design for each NAND3 gate and NAND2 gate must be slightly different to accommodate the various gate sizes calculated for each stage. E.g the first stage NAND3 gate will have different PMOS and NMOS sizes than the second

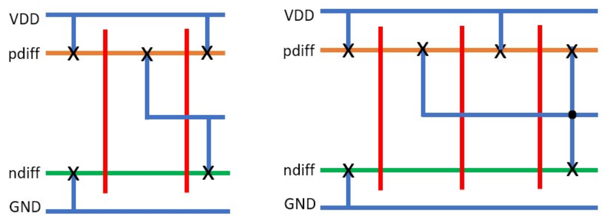


Fig. 12. Stick diagrams of NAND2 (left) and NAND3 (right) gates

stage NAND3 gate. Also, the third stage NAND2 will have different sizes than the fourth stage NAND2 gate. The layout designs for the first and second stage NAND3 gates are shown in Figure 14.

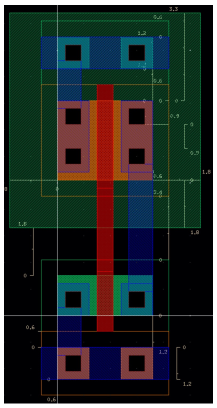


Fig. 13. Stick diagrams of NAND2 (left) and NAND3 (right) gates

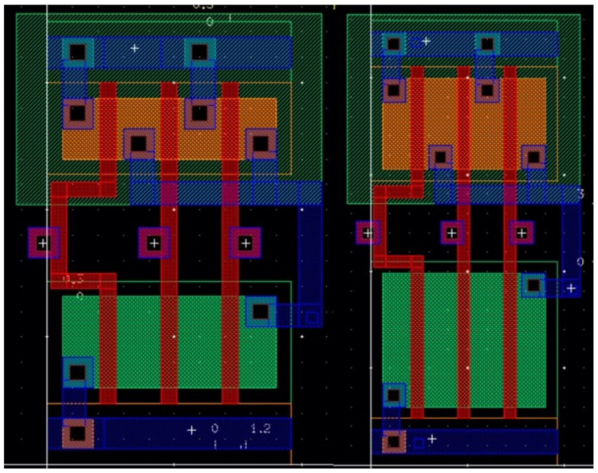


Fig. 14. Layout designs of first (left) and second (right) stage NAND3 gates

The first stage NAND3 gate has a PMOS to NMOS size ratio of 2.368um:3.553um. The second stage NAND3 gate has a ratio of 4.629um:6.944um. In the layout design, the polysilicon area is extended leftwards as to satisfy metal-spacing design rules. The layout designs for the third and fourth stage NAND2 gates are shown in Figure 15. The third stage NAND2 has a size ratio of 11.31um:11.31um. The fourth stage NAND2 gate has a ratio of 27.63um:27.63um.

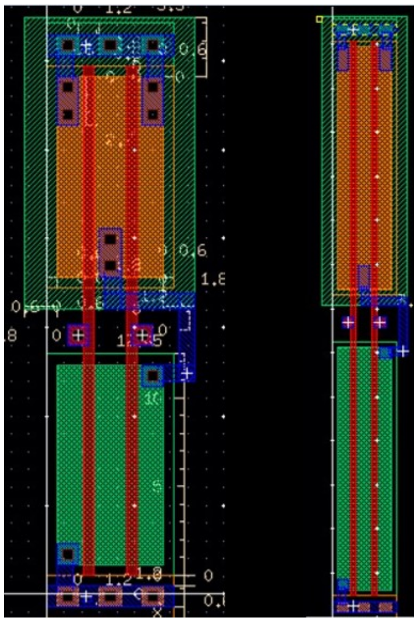


Fig. 15. Layout designs of third (left) and fourth (right) stage NAND2 gates

Each layout is individually tested for design rule errors using the Design rule check (DRC) tool. Additionally, the Layout vs. Schematic tool (LVS) is used to verify each individual design corresponds to the schematic netlist. Each layout satisfied the Design rule checker and LVS.

The parasitic capacitance's of each layout design can be extracted. In Figure 16 the extracted view showing the parasitic capacitance of the first and second stage NAND3 gates are displayed. In Figure 17 the extracted view showing the parasitic capacitance of the third and fourth stage NAND2 gates are displayed.

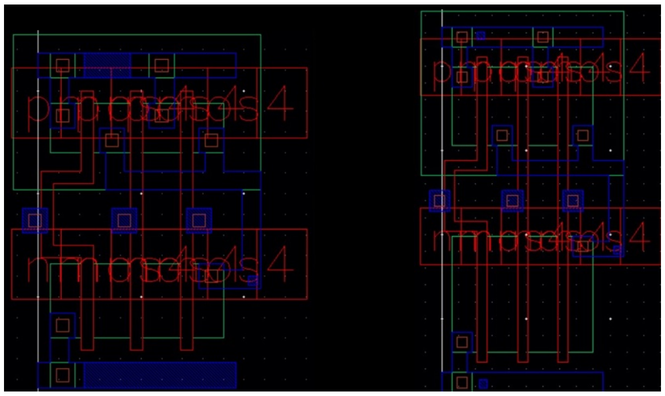


Fig. 16. Extracted views of first (left) and second (right) stage NAND3 gates

Each layout design can be implemented in a single layout to form the overall J-K Flip-Flop. Careful design considerations must be made in this part of the design process. Each stage must be implemented in the overall design as to minimize wasted space. Multiple metal layers should also be used to more easily optimize the design by having better routing and

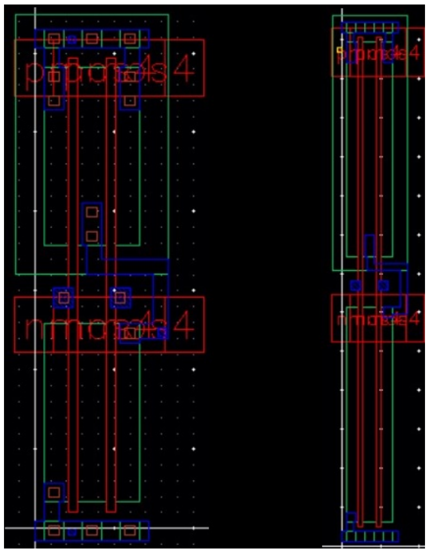


Fig. 17. Extracted views of third (left) and fourth (right) stage NAND2 gates

connection placement. The overall layout design of the J-K Flip-Flop is shown in Figure 18 below.

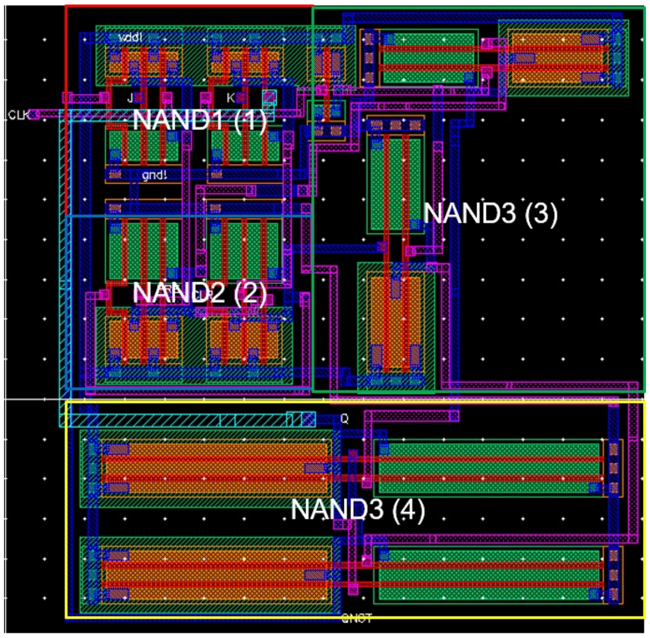


Fig. 18. Overall layout design of J-K Flip-Flop

The size of the layout design is approximately $78\mu m \times 78\mu m$ or $5700\mu m^2$. Expressed in the unit lambda, the size is $243\lambda \times 260\lambda$ or $63180\lambda^2$. Using the Design rule check (DRC) there are no design rules in the design. Also, the Layout vs. Schematic (LVS) tool verifies that the netlist of the J-K Flip-Flop layout matches the netlist of the schematic. The extracted view of the layout showing parasitic capacitance can be seen in Figure 19.

The overall extracted layout design of the J-K Flip-Flop is implemented as a symbol for Spectre simulation. The

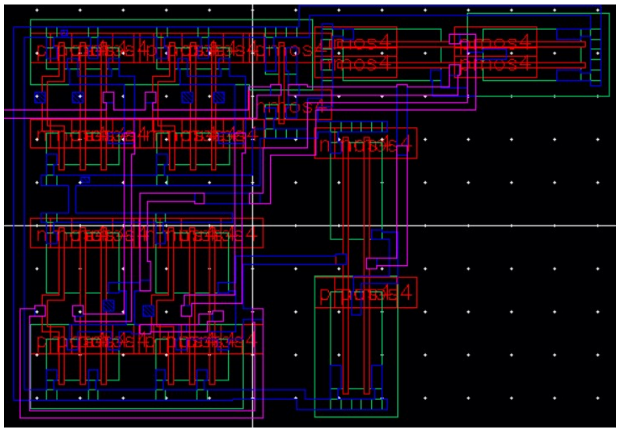


Fig. 19. Extracted view of the overall J-K Flip-Flop layout

simulation contains the 3C inverter input and the 90C output load as seen in Figure 20 below.

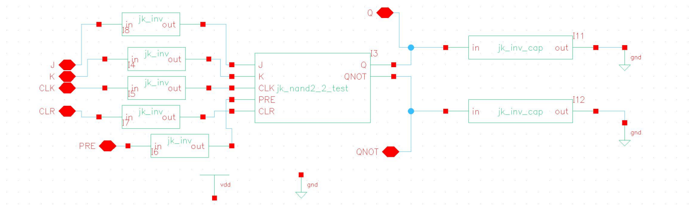


Fig. 20. J-K Flip-Flop with 3C input capacitance and 90C load

Spectre simulation is performed on the finalized layout for the J-K Flip-Flop. The inputs are varied as to see every combination and verify functionality of the circuit (same combinations as schematic simulation). Input J pulses with a period of 4ns and a pulse width of 2.5ns. Input K pulses with a period of 8ns and a pulse width of 4ns. The clock input pulses with a period of 2ns and a pulse width of 1ns. The input and output waveforms of the simulation are seen in Figure 21 below.

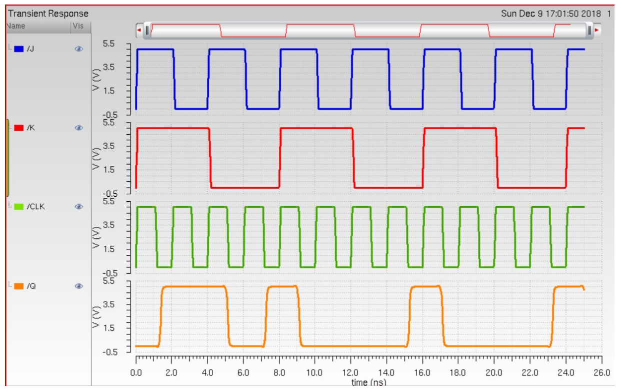


Fig. 21. Spectre simulation of layout design for J-K Flip-Flop (all inputs)

To more clearly visualize the delay, a simulation with only the clock input and the output Q is shown in Figure 22 below.

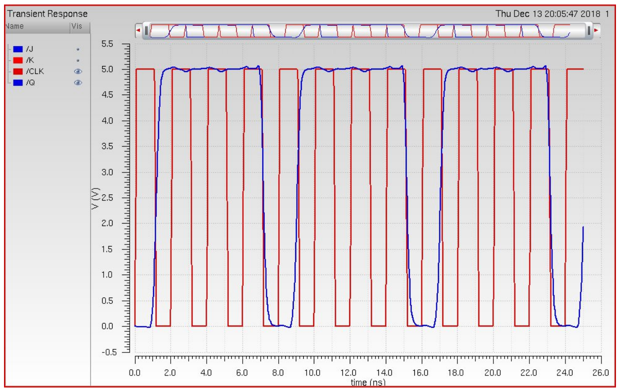


Fig. 22. Spectre simulation of layout design for J-K Flip-Flop (CLK and Q)

The falling propagation delay of the schematic design is found to be 1070ps. The rising propagation delay is found to be 125.0ps. The average propagation delay is calculated to be 600.0ps. Additionally, the functionality of the J-K Flip-Flop layout is verified by observation of inputs and outputs.

V. Conclusions

In this report the complete design process was exercised by making a J-K Flip-Flop. The design process included: truth table analysis, logical analysis, schematic design, layout design, and simulations. The master-slave J-K Flip-Flop design was chosen as it solves the issue of oscillation with short clock pulses. The logical effort of the circuit was found along with the least delays. Spectre simulation was performed on the schematic design and layout designs. The delays for each design are seen in Table 8 below.

TABLE VIII

	Rising Prop. Delay	**Falling Prop. Delay**	**Avg. Delay**
Logical Effort	X	X	582.11ps
Schematic	164.25ps	894.12ps	529.19ps
Layout	125.00ps	1070.0ps	600.00ps

Compared to the estimated least delay from logical effort, the schematic delay had a percent error of 9.0%. The layout delay had a percent error of 3.0%. The schematic design had less delay overall than the layout design. The layout design had more delay than the logical effort. The schematic design had less delay than the logical effort.

By the comparison of the estimated and calculated delays from logical effort, it can be determined that the design process and the layout design was accurate. The schematic simulation and layout simulation are reasonable in context of the logical effort calculated delay. It can be concluded that logical effort analysis provides good approximation for delay.

References

[1] [1] The CMOS Inverter Explained, Courseware.ee.calpoly.edu, 2018. [Online]. Available: https://courseware.ee.calpoly.edu/ dbraun/courses/ee307/F02/02 Shelley/Section2 BasilShelley.htm. [Accessed: 26- Sep-2018].

[2] [2] Mosis.com, 2018. [Online]. Available: https://www.mosis.com/files/scmos/scmos.pdf. [Accessed: 24- Oct2018].

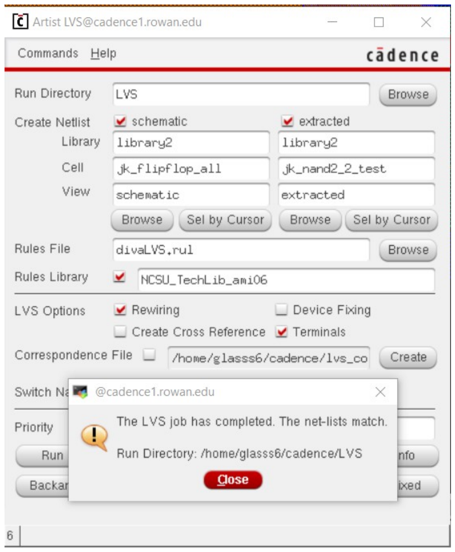
[3] [3] CMOS Digital Logic Circuits, Fourier.eng.hmc.edu, 2018. [Online]. Available: http://fourier.eng.hmc.edu/e84/lectures/ch4/node15.html. [Accessed: 26- Sep- 2018].

[4] [4] ”VLSI Design”, Dr. Sangho Shin, 2018. [Online].

[5] [5] The MOSIS Service: Vendors : ON Semiconductor: LM-AMI-C5N, Mosis.com, 2018. [Online]. Available: https://www.mosis.com/vendors/view/on-semiconductor/lm-ami-c5n. [Accessed: 24- Oct- 2018].

[6] [6] ”J-K Flip-Flop”, Hyperphysics.phy-astr.gsu.edu, 2018. [Online].

VI. Appendix



Artist LVS@cadence1.rowan.edu

Commands Help

Run Directory: LVS
Create Netlist: ✓ schematic / ✓ extracted
Library: library2 / library2
Cell: jk_flipflop_all / jk_nand2_2_test
View: schematic / extracted
Rules File: divaLVS.rul
Rules Library: ✓ NCSU_TechLib_ami06
LVS Options: ✓ Rewiring, Device Fixing, Create Cross Reference, ✓ Terminals
Correspondence File: /home/glasss6/cadence/lvs_co

@cadence1.rowan.edu

The LVS job has completed. The net-lists match.

Run Directory: /home/glasss6/cadence/LVS

Fig. 23. LVS verification of layout netlist matching schematic

```
// Generated for: spectre
// Generated on: Dec  6 01:45:12 2018
// Design library name: library2
// Design cell name: jk_flipflop_all
// Design view name: schematic
simulator lang=spectre
global 0 vdd!
include "/home/shin/pdk/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06N.m"
include "/home/shin/pdk/ncsu-cdk-1.6.0.beta/models/spectre/nom/ami06P.m"

// Library name: library2
// Cell name: jk_flipflop_all
// View name: schematic
P21 (QNOT net069 vdd! vdd!) ami06P w=27.6u l=600n as=4.14e-11 ad=4.14e-11 \
        ps=58.2u pd=58.2u m=1 region=sat
P20 (Q QNOT vdd! vdd!) ami06P w=27.6u l=600n as=4.14e-11 ad=4.14e-11 \
        ps=58.2u pd=58.2u m=1 region=sat
P19 (Q net070 vdd! vdd!) ami06P w=27.6u l=600n as=4.14e-11 ad=4.14e-11 \
        ps=58.2u pd=58.2u m=1 region=sat
P18 (QNOT Q vdd! vdd!) ami06P w=27.6u l=600n as=4.14e-11 ad=4.14e-11 \
        ps=58.2u pd=58.2u m=1 region=sat
P17 (net049 CLK vdd! vdd!) ami06P w=3u l=600n as=4.5e-12 ad=4.5e-12 ps=9u \
        pd=9u m=1 region=sat
P16 (net069 net028 vdd! vdd!) ami06P w=11.25u l=600n as=1.6875e-11 \
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region=sat
P14 (net070 net049 vdd! vdd!) ami06P w=11.25u l=600n as=1.6875e-11 \
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region=sat
P13 (net070 net059 vdd! vdd!) ami06P w=11.25u l=600n as=1.6875e-11 \
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region=sat
P15 (net069 net049 vdd! vdd!) ami06P w=11.25u l=600n as=1.6875e-11 \
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region=sat
P11 (net028 CLR vdd! vdd!) ami06P w=4.65u l=600n as=6.975e-12 ad=6.975e-12 \
        ps=12.3u pd=12.3u m=1 region=sat
```

```
P10 (net028 net07 vdd! vdd!) ami06P w=4.65u l=600n
        ad=6.975e-12 ps=12.3u pd=12.3u m=1 region=
P9 (net028 net059 vdd! vdd!) ami06P w=4.65u l=600n
        ad=6.975e-12 ps=12.3u pd=12.3u m=1 region=
P8 (net059 net028 vdd! vdd!) ami06P w=4.65u l=600n
        ad=6.975e-12 ps=12.3u pd=12.3u m=1 region=
P7 (net059 net2 vdd! vdd!) ami06P w=4.65u l=600n a
        ps=12.3u pd=12.3u m=1 region=sat
P6 (net059 PRE vdd! vdd!) ami06P w=4.65u l=600n as
        ps=12.3u pd=12.3u m=1 region=sat
P5 (net07 Q vdd! vdd!) ami06P w=2.4u l=600n as=3.6
        pd=7.8u m=1 region=sat
P4 (net07 K vdd! vdd!) ami06P w=2.4u l=600n as=3.6
        pd=7.8u m=1 region=sat
P3 (net07 CLK vdd! vdd!) ami06P w=2.4u l=600n as=3
        ps=7.8u pd=7.8u m=1 region=sat
P2 (net2 CLK vdd! vdd!) ami06P w=2.4u l=600n as=3.
        pd=7.8u m=1 region=sat
P1 (net2 J vdd! vdd!) ami06P w=2.4u l=600n as=3.6e
        pd=7.8u m=1 region=sat
P0 (net2 QNOT vdd! vdd!) ami06P w=2.4u l=600n as=3
        ps=7.8u pd=7.8u m=1 region=sat
N21 (QNOT Q net0102 net097) ami06N w=27.6u l=600n
        ps=58.2u pd=58.2u m=1 region=sat
N20 (net0102 net069 0 net097) ami06N w=27.6u l=600
        ad=4.14e-11 ps=58.2u pd=58.2u m=1 region=s
N19 (Q net070 net0101 net098) ami06N w=27.6u l=600
        ad=4.14e-11 ps=58.2u pd=58.2u m=1 region=s
N18 (net0101 QNOT 0 net098) ami06N w=27.6u l=600n
        ps=58.2u pd=58.2u m=1 region=sat
N17 (net049 CLK 0 0) ami06N w=1.5u l=600n as=2.25e
        ps=6u pd=6u m=1 region=sat
N16 (net069 net049 net078 net075) ami06N w=11.25u
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region
N15 (net078 net028 0 net075) ami06N w=11.25u l=600
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region
N14 (net070 net059 net080 net076) ami06N w=11.25u
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region
N13 (net080 net049 0 net076) ami06N w=11.25u l=600
        ad=1.6875e-11 ps=25.5u pd=25.5u m=1 region
N11 (net028 net059 net052 net110) ami06N w=6.9u l=600n
        ad=1.035e-11 ps=16.8u pd=16.8u m=1 region=
N10 (net052 net07 net051 0) ami06N w=6.9u l=600n a
        ps=16.8u pd=16.8u m=1 region=sat
N9 (net051 CLR 0 0) ami06N w=6.9u l=600n as=1.035e
        ps=16.8u pd=16.8u m=1 region=sat
N8 (net059 PRE net054 net110) ami06N w=6.9u l=600n as=1
        ps=16.8u pd=16.8u m=1 region=sat
N7 (net054 net028 net053 0) ami06N w=6.9u l=600n as=
        ps=16.8u pd=16.8u m=1 region=sat
N6 (net053 [illegible]) ami06N w=6.9u l=600n as=1.0
        ps=16.8u pd=16.8u m=1 region=sat
N5 (net07 CLK [illegible]) ami06N w=3.6u l=600n as=5.
        ps=10.2u pd=10.2u m=1 region=sat
N4 (net027 K net026 0) ami06N w=3.6u l=600n as=5.4
        pd=10.2u m=1 region=sat
```

```
N3 (net026 Q 0 0) ami06N w=3.6u l=600n as=5.4e-12 ad=5.4e-12 ps=10.2u \
        pd=10.2u m=1 region=sat
N2 (net2 QNOT net11 0) ami06N w=3.6u l=600n as=5.4e-12 ad=5.4e-12 ps=10.2u \
        pd=10.2u m=1 region=sat
N1 (net11 J net10 0) ami06N w=3.6u l=600n as=5.4e-12 ad=5.4e-12 ps=10.2u \
        pd=10.2u m=1 region=sat
N0 (net10 CLK 0 0) ami06N w=3.6u l=600n as=5.4e-12 ad=5.4e-12 ps=10.2u \
        pd=10.2u m=1 region=sat
include "./_graphical_stimuli.scs"
simulatorOptions options reltol=1e-3 vabstol=1e-6 iabstol=1e-12 temp=27 \
    tnom=27 scalem=1.0 scale=1.0 gmin=1e-12 rforce=1 maxnotes=5 maxwarns=5 \
    digits=5 cols=80 pivrel=1e-3 sensfile="../psf/sens.output" \
    checklimitdest=psf
tran tran stop=50n errpreset=moderate write="spectre.ic" \
    writefinal="spectre.fc" annotate=status maxiters=5
finalTimeOP info what=oppoint where=rawfile
modelParameter info what=models where=rawfile
element info what=inst where=rawfile
outputParameter info what=output where=rawfile
designParamVals info what=parameters where=rawfile
primitives info what=primitives where=rawfile
subckts info what=subckts where=rawfile
saveOptions options save=allpub
```