Lab 2 Report
Accumulator (ECE558)

Luan Vo (30553600), October 13th 2020

***Purpose & Introduction* — In this lab, we will focus on how to make an accumulator, this accumulator is made up of a flipflop and half adder. The goal is to design an accumulator so that it is optimized with low rise and fall clock cycles.**

I. System Functions and Truth Table (STEP 1):

A. Truth tables

Table 1. *Half adder*

CIN	SOUT	S	COUT
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Table 2. *NAND gate*

S	RST	D
0	0	1
0	1	1
1	0	1
1	1	0

Table 3. *Flip flop and inverter truth table*

D	PHI	OUT	SOUT
0	Rising	0	1
1	Rising	1	0
X	Falling	X	X

B. Design Foundations:

In our design, we want when RST = 0, the system SOUT will be 0, this means that, when RST equal to zero, we completely ignore whatever the input of S is, and by looking at the highlighted rows in table 1 and 3, we see that this is indeed the case. However, this means that, whenever RST is high, the system will continue its operation normally.

Note: I will be using "~=" to denote very closely equal.

II. Layout Requirements (INTERMEDIATE STEP):

Below is a comprehensive list of the requirements set forth by this lab:

- 45nm technology, with L = 50nm => λ=25nm.
- Cell height = 140λ, with 20λ-high rails.
- The nominal supply voltage is 1.1V.
- Use β = 2 (the P/N width ratio).
- Upsize appropriately for series transistors while ensuring that the smallest devices satisfy the process minimum rules (NMOS=90nm, PMOS=180nm).
- Except where stated otherwise, assume that inputs, including the clock, have 30 ps rise time and fall times (tr and tf, using 20%-80% metric from pg 141 of book)

III. Create Schematics (STEP 2):

Important textbook pages (pdf): pg 498 (XOR), pg 423 (flipflop)

A. Design of Half Adder

Half adder can be made using various combinations of logic gates, however, for this assignment, our half adder will be made of: an XOR gate with 2 (inverted) inputs NOR gate. For a visual check, observe the table below:

Table 4. *XOR and NOR truth tables*

XOR			NOR (inverted inputs)		
CIN	SOUT	Y	$\overline{CIN}$	$\overline{SOUT}$	K
0	0	0	1	1	0
0	1	1	1	0	0
1	0	1	0	1	0
1	1	0	0	0	1

We see that Y is the sum of the half adder, whereas, K is Cout. This means that the circuit would look something like this:

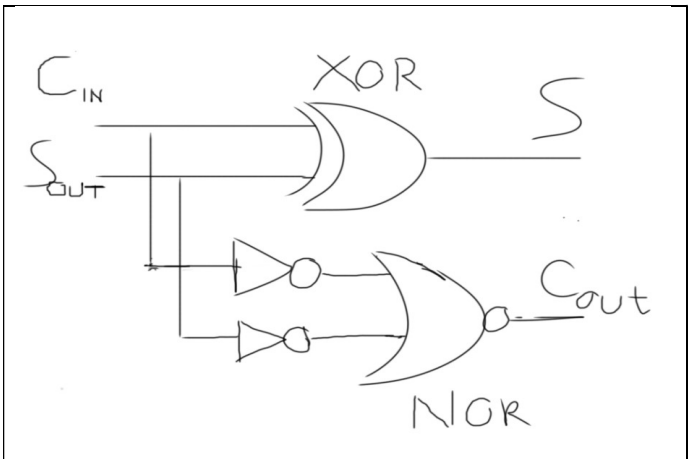


Figure 1. *Half Adder Implementation*

Where XOR gate with the output of Y, can be implemented as follow:

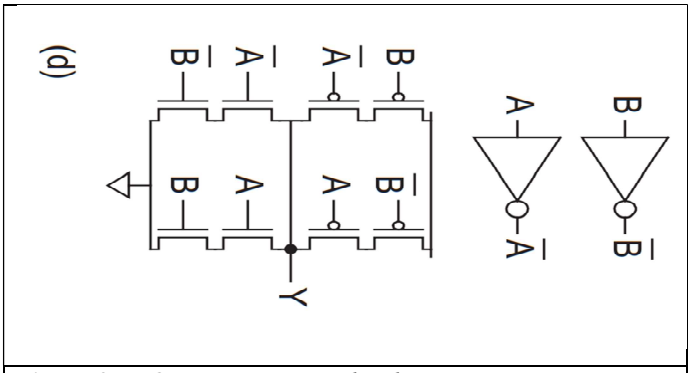


Figure 2. *XOR gate transistor level*

Implementing half adder transistor level (Check appendix A for enlarged version):

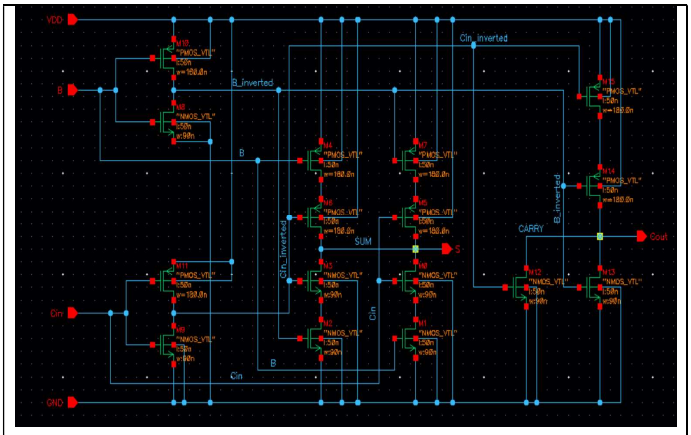


Figure 3. *Half Adder transistor level (zoom in to see)*

--------Half adder test run--------

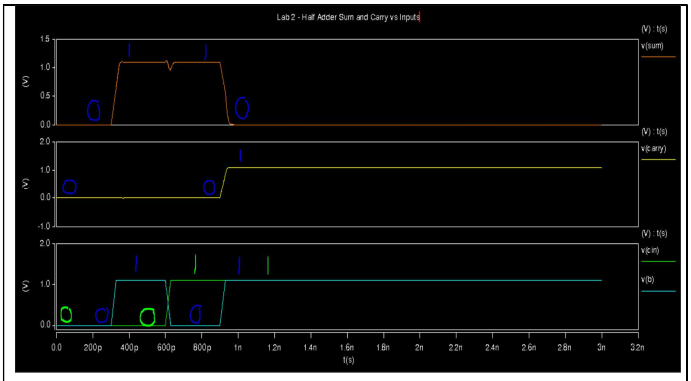


Figure 4. *Half Adder's preliminary outputs*

We see that the output of the half adder's sum and carry meets the expected value of the truth table for a typical half adder (as stated in table 2). With this, we can move on to examine the layout for the flip flop, as NAND gate layout is provided in lab 1 assignment. According to this graph, we have the following:

Table 5. *CScope truth table for HA*

S (sum)	0	1	1	0
Cout	0	0	0	1

B. Synchronous Flip flop

Below is the C2MOS flip-flop model that we will be using:

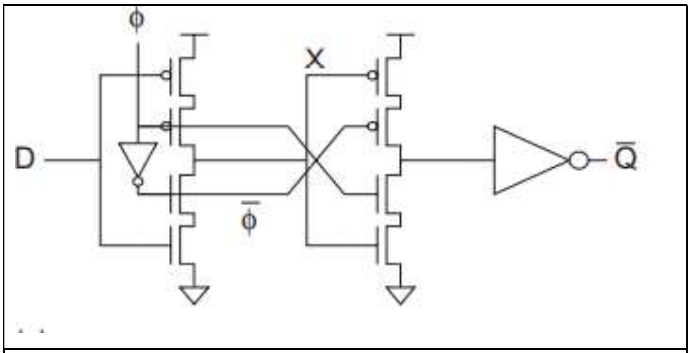


Figure 5. *C2MOS (NORA) flipflop (Page 394)*

Implementing C2MOS (NORA) flipflop transistor level (Check appendix A for enlarged version):

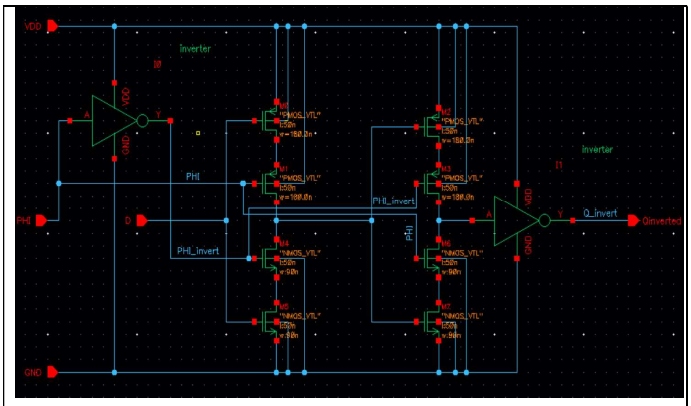


Figure 6. *Flipflop transistor level (zoom in to see)*

--------Flipflop test run--------

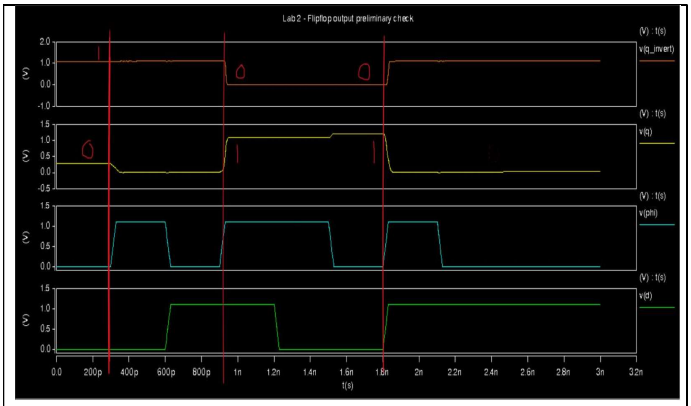


Figure 7. *Flipflop transistor level (zoom in to see)*

Flipflop only stores new values when clock PHI is at the rising edge (i.e: from 0 to 1), else, flipflop does not change what is currently being output. This can be seen at the red lines drawn at PHI rising edge. According to the graph, we have the following table for flipflop output with inverter (where X stands for unchanging value).

Table 6. *CScope flip flop and inverter truth table*

D	PHI	OUT	SOUT
0	Rising	0	1
1	Rising	1	0
X	Falling	X	X

This table agrees with table 1 output for a conventional flipflop, where SOUT is OUT inverted. With this, we can move on to the final phase of building the entire accumulator.

C. *Accumulator*

By combining everything we have done previously in part A and B (with NAND gate from lab 1), the circuit for the accumulator is (larger picture will be in appendix A):

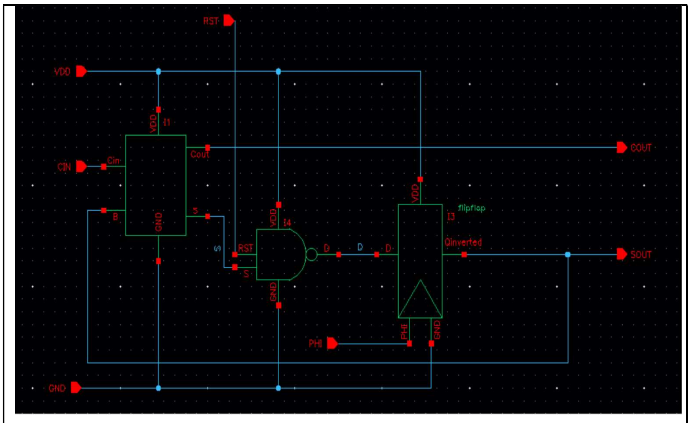


Figure 8. *Accumulator Schematic gate level (zoom in to see)*

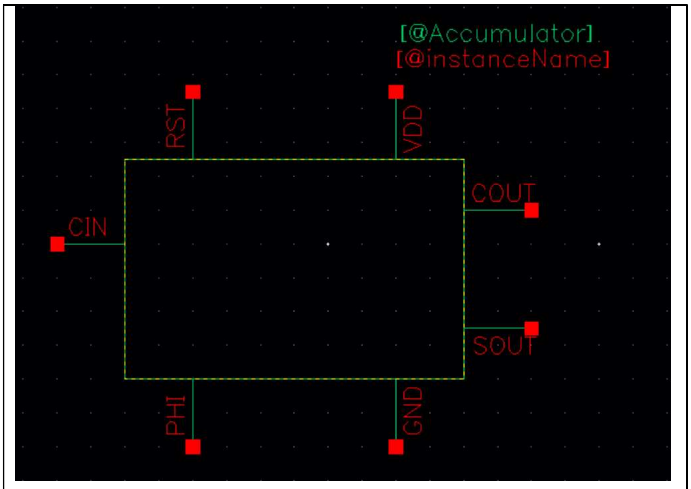


Figure 9. *Accumulator Symbol graph*

IV. CHECK FUNCTIONALITY OF SCHEMATIC (STEP 3):

A. *Checking with truth table, stable PHI*

For this check, we must note some peculiarity of the accumulator, which lies in the flipflop and active-low reset NAND gate, and inputs are CIN, RST, and PHI.

- NAND gate: as mentioned in table 3, ***when RST equals to 0 and when S isn't 0, the output D will always be 1.*** (which means that SOUT will be reversed, i.e SOUT is 0 when D is 1)
- Flipflop: flipflop will sample input (D) when it is on the rising edge, meaning that ***flipflop will only be storing whatever the input of D is whenever it is transitioning from 0 to 1***.
- This means that the output is dependent on the input of PHI, thus, we have to TIME it so that the input is correctly examined at the rising edge only. Therefore, any values at non-rising edge state will be denoted as X (i.e: don't care or values that don't impact results).
- In addition, RST = 0, makes SOUT = 0 on the NEXT rising edge, not the one currently on.
- ***The output (SOUT) is D inverted.***

Thus, we have the following table:

Table 7. *Accumulator truth table with stable PHI*

Line	CIN	SOUT	COUT	S	RST	PHI	D
1	X	0	0	X	0	0	1
2	X	0	K	X	X	1	X
3	1	0	0	1	1	0	0
4	X	1	K	X	X	1	X
5	1	1	1	0	X	0	1
6	X	0	K	X	X	1	X
7	0	0	0	0	X	0	1
8	X	0	K	X	X	1	X
9	1	0	0	1	0	0	1
10	X	0	K	X	X	1	X

Where ***Black denotes "don't care" and won't impact SOUT***. ***Yellow is the active-low test***, if changed to 1, SOUT will change. ***Reds are values that are at rising edge and will get forwarded to SOUT*** and will impact results if changed. ***Greens are values of SOUT that is D inverted on next clock*** and the ***next SOUT is dependent on the previous SOUT (sout line 2nd depends on 1st)***

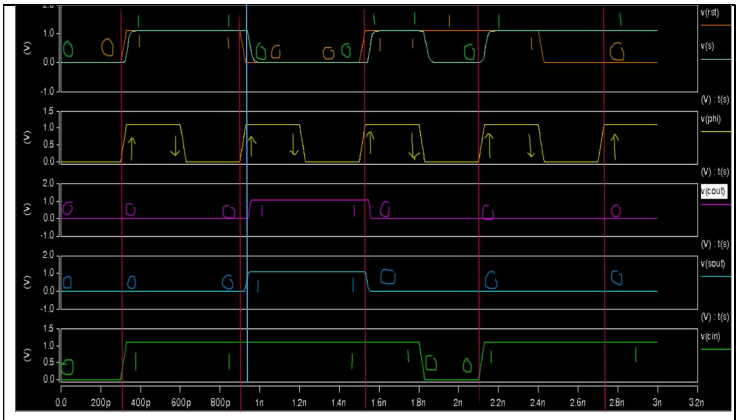


Figure 9. *Accumulator with Table 7 inputs graph*

Inputs used in .vec file (where Xs from table are filled randomly)

CIN = 0111110111 | RST = 0110011100 | PHI = 0101010101

B. Analyze observations

As shown in Table 7 and Fig. 9, we see that the results for S, SOUT, and COUT match perfectly. This means that table 7 is correct. Here, the most important aspect is that the flipflop only samples inputs when the clock is rising, it means ***that D is related to SOUT (which it should be) and that relation is highlighted green***. For example, SOUT in line 3 is D in line 1 inverted, and so on. We see that D does not perfectly match with the current SOUT because the flipflop needs to complete 1 cycle to release D to its output. ***This adheres to the principle of the flipflop as it takes 1 cycle (i.e: PHI = 0 1) to complete the load input and flush the current value out***. This is because we only see that SOUT is only ever being updated when PHI is rising.

Changing in CIN will undoubtedly change COUT, as COUT is a direct computational step of HA, which means that the value of ***COUT is dependent on whatever the input CIN and the current value that is stored in SOUT by flipflop, however, while CIN may change, it will not impact the result of SOUT (which is the output that we care about).*** These values are denoted as K in table 7 which stands for K = CIN + SOUT.

The flipflop only accepts at the rising edge, which means that any other transitions (11 or 10) values will not be loaded to flipflops. ***This is represented as the black highlight and X which means that these values do not matter and will not impact results if changed***. This is proven by my random assignments of 0 and 1 at these X positions for the input.vec file.

Lastly, ***the inputs of PHI are highlighted blue, for better visualization*** when reading the table. Besides, ***the yellow highlight is meant to test the active-low reset function of the accumulator***, if this bit is set to 1, the result will change accordingly and SOUT will be 1 instead of 0 (no longer being reset). Vice versa, if this bit is set to 0 (when S not 0), it should reset SOUT back to 0. ***Where reds denote the values that are currently at the rising edge of PHI clock (from 0 to 1)***.

C. How is this input set cover all cases?

This input set covers all cases because it covers all behaviors of RST and CIN while properly generate a correct PHI clock signal (pulse).

As shown in Fig. 9, PHI is a pulse signal alternating between 0 and 1, with this setup, we have both behaviors of rising and falling edge that is going into flipflop. In this case, we do not need to perform 11 and 00 for PHI because no edges are occurring at these points. ***For a clock, PHI must remain consistent and consist of 2 edges***.

For RST, we see that the input sequence covers all cases of resetting and not resetting, during and not during the rising edge of PHI. For example, at line 1, RST = 0 at the rising edge of PHI and line 3 where RST = 1 at the rising edge of PHI. In addition, I randomly assigned 1 and 0 bits for RST at the non-rising edge of PHI and the result for SOUT doesn't change (refer to part D below). This means that this RST sequence covers all cases for resetting the NAND gate.

Lastly, For CIN, it also covers any additions at the rising edge from CIN either 1 or 0, and again, any X noted in table 7 can be interchanged. Further proofs are shown in Part D and E.

D. Device behavior when X positions are flipped (EXTRA)

The purpose is to examine whether SOUT will change these X (don't care) values are modified. If my thinking is correct, ***SOUT will remain the same as in Fig. 9, however, there will be changes in COUT***. Our modified inputs are (with PHI and RST are the same): CIN = 1010100010

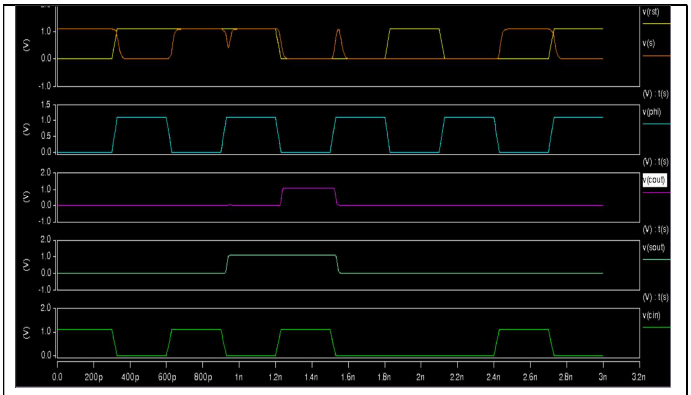


Figure 10. *Accumulator with modified CIN*

Now, I will change the inputs of RST at X positions are flipped compared to part A, and our modified RST and CIN:

CIN = 1010100010 | RST = 0011100001

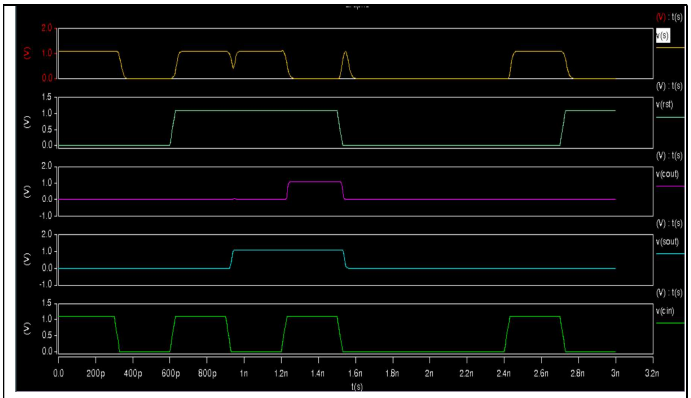


Figure 11. *Accumulator with modified CIN and RST*

We see that, the behavior of the accumulator aligned with what we have theorized in part A and part B ***since SOUT remains consistent even when changing the values at X positions***. However, we can see that ***CIN at K positions changed***.

E. Active-low reset test (RST in line 9) (EXTRA)

This is an extra test to confirm the functionality of RST, which is in line 9 in table 7 – by setting that RST to 1.

Part A inputs test: CIN = 0111110111 | RST = 0110011110

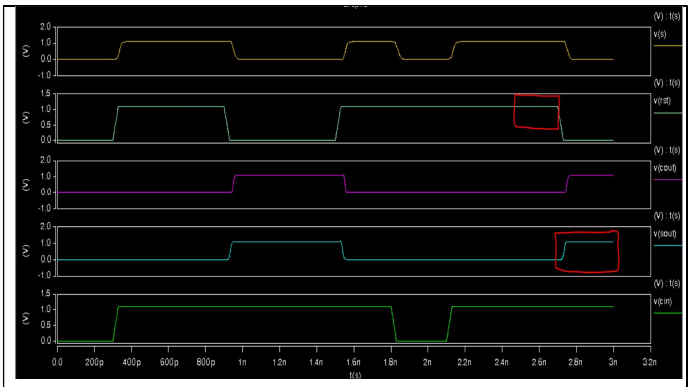


Figure 12. *Accumulator's active-low reset part A's inputs*

Part C inputs test: CIN = 1010100010 | RST = 0011100011

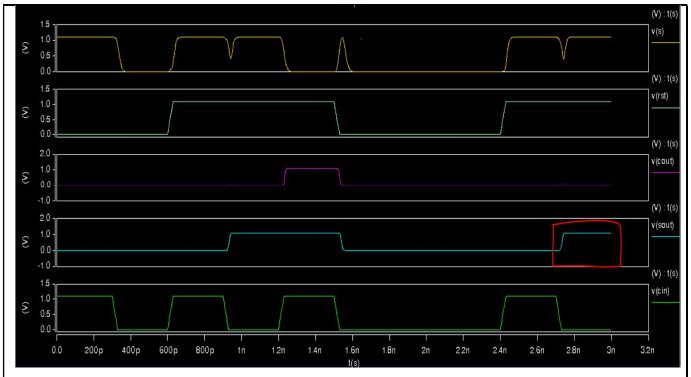


Figure 13. *Accumulator's active-low reset part D's inputs*

Fig. 12 and 13 show that the active-low reset logic work at line 9 (and consequentially, line 1 and 3) in table 7.

V. DETERMINE MAXIMUM CLOCK FREQUENCY (STEP 4):

Since frequency is 1/time. Therefore, the higher the frequency, the smaller the clock time. Also, to avoid triangle-shaped waveforms, we want the rise and fall time to be 10% of the period. Input sequences are the same as STEP 3.

A. Maximum clock

Initially: Rise/Fall = 30ps, Period = 300ps

Maximum: Rise/Fall = 3ps, Period = 30ps => f = 33.3GHz

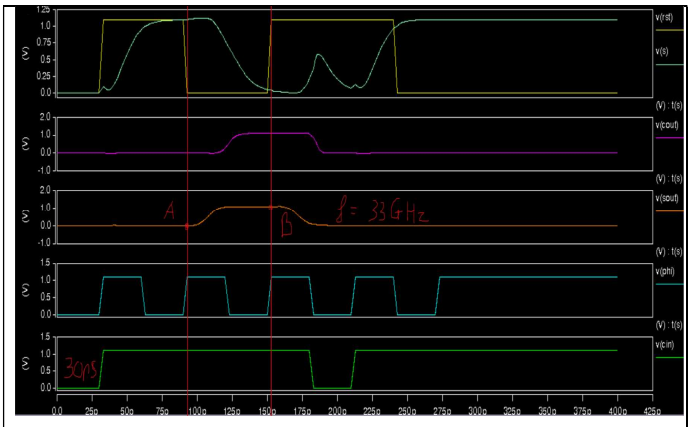


Figure 14. *Accumulator with clock speed of 33.3 GHz (30ps)*

As shown in the graph, SOUT and COUT remains the same and stays consistent with what we have in part 3 (Fig. 9). I think that this is the maximum/optimal operational frequency for this system. If we further increase the frequency, the value detected at rising edge A will be further to the left, which means that the value detected at point A will be 0. This is different than what we have in part 3, as in part 3, at this rising edge, the value should already be at the increasing slope (from 0 to 1) in SOUT.

The only effect that I noticed so far by increasing the frequency is that the values of SOUT and COUT are being shifted further to the right of the graph. While my inputs do not have any values followed SOUT and COUT after point B, nonetheless so long that the ***values detected at the rising edge are either inconsistent or delayed from what we would expect in normal circumstances (step 3, Fig. 9), then it is safe to say that such behaviors mean that the circuit does not behave as expected*** (since the clock cycle is faster than what the switching could handle). ***The correct result is captured between A and B, this will be the reference point to compare with part B.***

B. When the system is unstable and 10% above

The system is, in my opinion, considered unstable at 11.1 GHz. We see that, while the change is very small, nonetheless, at 11.1 GHz (90ps) in Fig. 15, we see that rising edge at point A is not within the rising edge of SOUT anymore and the value at that point is completely zero. It means that the circuit is lagging and could not catch up with the clock. Where Fig. 16 is where the period is 10% above the unstable value.

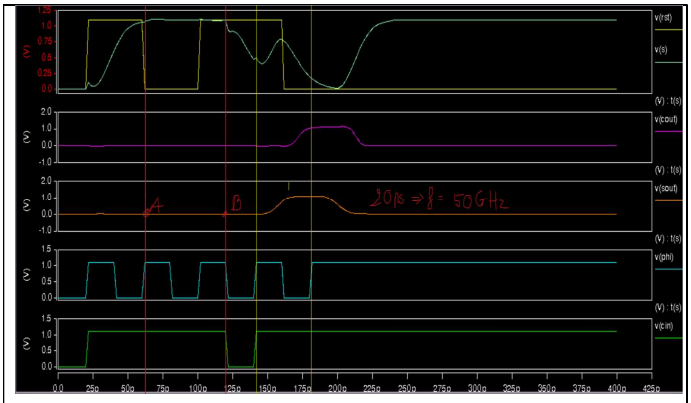


Figure 15. *Accumulator with clock speed of 50 GHz (20ps)*

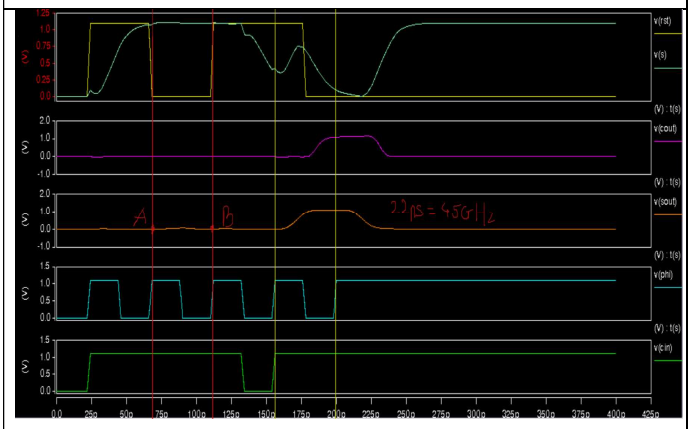


Figure 16. *Accumulator with clock speed of 45 GHz (22ps)*

From Fig. 15 and 16, we can see that the point at which the output seems to differ is when the period is at around 20ps because, at this point, we were expecting SOUT to be 1 (or at the rising edge of SOUT), however, at between point A and B, SOUT of 20ps outputs 0. With this, I concluded that the time after 30ps the output no longer behaves as expected, thus making around 30ps the maximum period of the system – this corresponds to around 33.3GHz in frequency.

Besides, we see that Fig. 16 shows an extra 10% of 20ps, which shows that the system still outputs the incorrect SOUT, which means that the assuming 33.3GHz to be the maximum performance frequency valid.

VI. Power Calculations (Step 5)

Before we jump into the session, here are some useful knowledge and information regarding the power consumption modes of a circuit. As we have learned, dynamic power can be represented as:

$$P_{dynamic} = P_{switching} + P_{short} \quad (1)$$

Whereas, static power is defined as the power leakage when the circuit is not in operation and can be represented as:

$$P_{static} = (I_{sub} + I_{gate} + I_{junc} + I_{contention}) * VDD \quad (2)$$

A. Dynamic Power Simulation

To perform this, I am assuming that $P_{dynamic} >> P_{static}$. Which means that $P_{total} \sim= P_{dynamic}$. ***Therefore, the approximation of the circuit's dynamic power consumption (through the entire input sequence) can be represented as the average power consumption***. Since the average power can give us a good approximation of the average dynamic power (i.e: approximate power of all the switching within the circuit). I will be taking measurements of power with the command below in HSPICE:

.measure tran avgpower AVG power from=1ps to= period (3)

Where the from and to will be our entire sequence operational time, as adding more than necessary will interfere with the average power calculation (since hspice assumes that the last bit will drag on until we put a stop to it). I will be making my power measurements from 300ps down to 40ps with several measurements. Following the measurements yield the following table: (stops at 40ps since 30ps is the maximum operational freq)

Note: I have 10 inputs, so the total period for hspice simulation is equal to: period = 10 * (one period).

Table 8. *Frequency vs AVG power*

One period (ps)	Frequency (GHz)	$P_{average}$ (uW)
300	3.33	10.39
270	3.7	11.19
240	4.167	11.7
210	4.76	13.2
180	5.55	15.66
150	6.67	17.2
120	8.33	21.98
90	11.11	30.45
60	16.67	48.11
50	20	57.9
45	22.22	64.1
40	25	68.8
35	28.57	72.0
33	30.3	74.4

In the end, I wanted to make the graph as accurate as possible, so I made a few more measurements. Below is the graph generated from this measurement set:

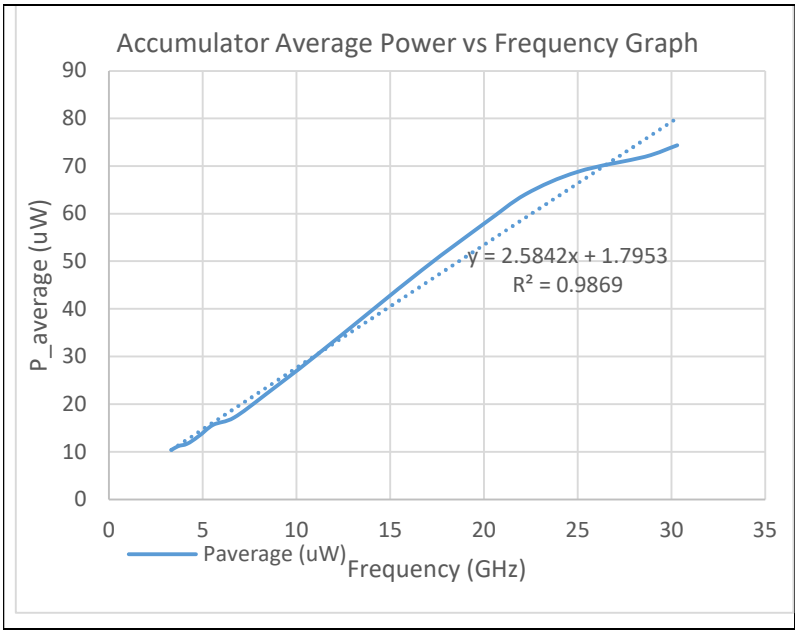


Figure 17. *Accumulator Average Power vs Frequency Graph*

From this graph, we can see that power consumption in terms of uW/GHz is approximately the slope of the line, which is shown through the trend line in the graph. The reason being that the slope represents the possible increment of uW per GHz. For example, at 11.11 GHz: $P_{average\sim}$ = 2.5842*11.11 = 28.71uW, which is almost equal to what we have in the table. However, this result yields an interesting observation: the intercept variable of the trendline.

Ultimately, the power in terms of uW/GHz is the slope, which is 2.5842 uW/GHz. Which is the equivalent of:

2.5842 uW/(MHz*10^-3) = 2.5842mW/MHz

The graph seems to stay consistent towards the end around for the given range from 3.3GHz to 30.3GHz, this can be seen through the consistency of the trendline of the given graph (shown as the dotted line). ***However, the graph exhibits a sudden dip around 30.3GHz, this means that we are approaching the maximum operational mode and is driving the FETs to be unsteady close to the maximum frequency. This explains why the FETs are kind of unsteady towards the maximum frequency.***

B. Extrapolate to predict

We predict the maximum frequency f = 33.33GHz or period ~=0. With this, our extrapolation of the maximum operating frequency of 33.33GHz is approximate:

$P_{average}$ = 2.5842(uW/GHz)*33.33(GHz) + 1.79uW = 87.9uW

Compares this with the hspice simulation value:

```
$DATA1 SOURCE='HSPICE' VERSION='E-2010.12-SP2 32-BIT'
.TITLE '** generated for: hspiced'
 wn                avgpower          maxpower          temper
                   alter#
    0.              7.909e-05         4.935e-04          25.0000
                   1
```

So the error % is: (79.09-87.9)/79.09 = 11.15%

While this isn't the closest approximation, however, it is close enough to +- 10% error margin. ***As stated in part A, as we get closer to the maximum frequency, the circuit becomes more and more unstable, which makes the power extrapolation to be slightly off***.

To calculate the static power, one interesting observation from this diagram is the y-intercept, in this case, is 1.79uW. Since static power is defined as the power leakage of FETs and devices when the circuit is not in operation or steady states, which means that this consumption is being consumed when transistors are consistent. Which means that, this consumption could be estimate to be consistent throughout the process – regardless of frequency. Which means that, ideally, the static power should be the y-intercept value. While I noticed that adding more points to the graph will cause a grow in inconsistency of the circuit trend line. Meaning that the best option should be making the trendline as fit as possible to the uW line, that way, we can get the most accurate readings of the static power, or y-intercept. The reason behind this is that, static power is extremely small when compared to dynamic power (In normal assumptions and ideals). However, as transistors keep on shrinking and can operate at a faster frequency, the contribution of static power to the total power consumption increases. Ideally, in a circuit architecture, we would only want dynamic power consumption – as static power is the lost of power without achieving any activities within the circuit [1]. Thus, to exclude the cases where the circuit behaves in an inconsistent fashion when close to the maximum frequency, I reduce the range to about 25GHz, and we have the graph below:

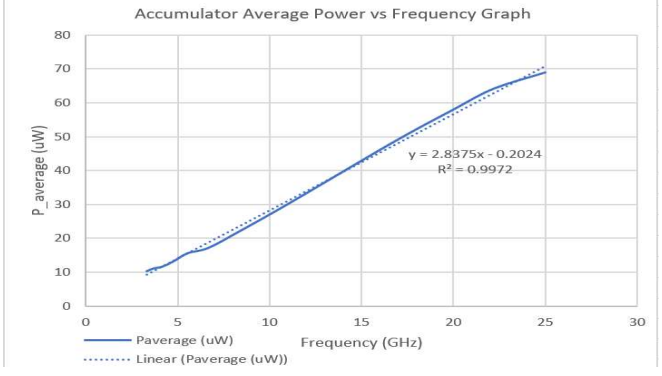


Figure 18. *Accumulator Power vs Frequency best fit line*

This measurement supports the idea that the closer we are to the maximum operational frequency, the more the circuit behaves inconsistently and thus, we should be measuring static power when the circuit is at its most stable state. ***From the graph, we can make an approximation that, ideally, static power will be approximately 0.2024uW***. It makes sense, as the static power should – ideally – be very small when compared to switching power.

The frequency when static power and dynamic power be equal should be:

F_{match} = 0.2024(uW)/2.8375(uW/GHz) = 0.071GHz

In other words, the matching frequency for dynamic and static power is: 0.071GHz = 71.33MHz.

VII. Effort Based Sizing (Step 7):

Load capacitance according to Student's ID (30553600):

$$C_{load} = 20 + (0 + 0) = 20fF$$

In this section, we will assume that:

i. No branching effort from SOUT to HA input.

ii. Our accumulator has $W_p = 2W_n$

iii. Our path starts from input C_{in} and ends at C_{load}

iv. Since electrical effort "h" is defined as C_{out}/C_{in}, it means that the total electrical effort of a path is equivalent to C_{load}/C_{first}. Where C_{first} denotes the capacitance of the first gate of the path. This is because every other capacitance cancels each other out: $C_2/C_1 * C_3/C_2 *\ldots *C_N/C_{N-1} = C_N/C_1$.

v. Assuming results from lab 1, we have 16 inverters equal to 5fF. Which means that each inverter has C = 0.31fF. That means the load is 20/0.31 = 64 times the load of a minimum sized inverter.

A. Effort and Sizing Relevant Equations

We have the diagram below for the accumulator's path and the efforts for each stage noted below:

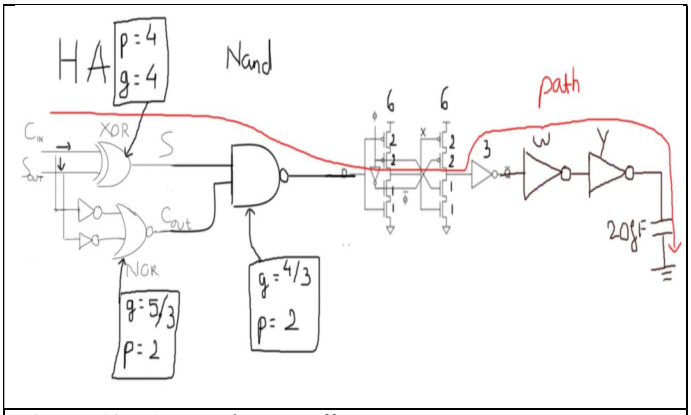


Figure 19. *Accumulator's effort mapping*

The XOR gate has a total of 8 MOSFETs, with 4 of each type. Since we opted to ignore the SOUT that is looping back to the system, we can affectively ignore the branching of SOUT which means that CIN directly controls 4 inputs of the 4 FETs (from

Fig. 3), and 2 of those 4 inputs are CIN_noninverted. Which means that the total input capacitance is equal to:

$$CIN \sim= Cin\_noninverting\_inputs + C\_inverter$$

$$CIN \sim= 3 + 6 \text{ (since inverter also drives another 2 FET)}$$

Where Cin_noninverting_inputs refer to the 2 gates that are directly connected to CIN (refer to Fig. 2, where CIN is A and A_bar). In addition, I decided that the flipflops would have a delay of 2 since it is a stack of 2 inverters.

Below is the diagram of stage by stage effort:

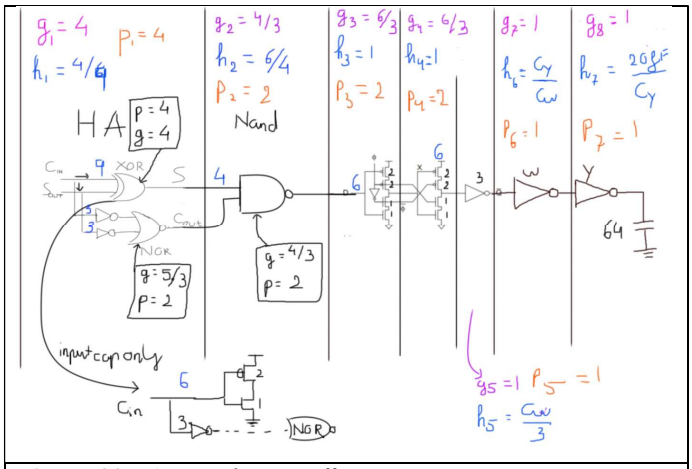


Figure 20. *Accumulator's effort mapping*

Therefore, we have the following parameters:

$G = 4*(4/3)*(6/3)*(6/3)*1*1*1 = 4*4/3*2*2 = 64/3$

$H = 64/9$

$P = 4+2+2+2+1+1+1 = 13$

$F = GH = (64/9)*(64/3) = 151.7$

$F\hat{} = F^{1/7} = 2.05$

Delay (D) = 7*F^ + P = 27.3

Now, we can work backwards for size:

Y = C_{load} * G_Y / F^ = (64/9 * 1)/2.05 = 3.47 ~= 4

W = C_Y * G_w / F^ = 3.47/2.05 = 1.7 ~= 2

The result does upsize as the driver gets closer to the load capacitance, and each stage is upsized by a factor of F^. The numbers above are expressed in term of parasitic inverter delay, as delays are related to dimensions of transistors.

B. Checking with SPICE

In order to approach this problem, we will use similar procedure to lab 1. There will be an initial measurement of the whole system with 2 stage drivers and a 20fF load capacitance. After that, we will only attach the accumulator with some load capacitance and extrapolate the capacitance of the 2 upsized inverters, while not needed for this lab, here is the formula:

$$C_{extra} = (1/C_{load} + 1/C_{inverter} + 1/C_{inverter})^{-1}$$

This is because the capacitors are being lined up in series, and for capacitors in series, we add them up parallelly.

As stated by the lab procedure, we first measure delay of an unloaded inverter.

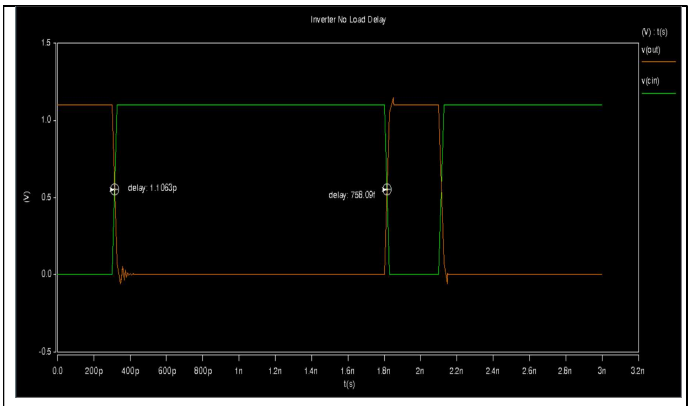


Figure 21. *Inverter Delay No Load @ 3.33GHz*

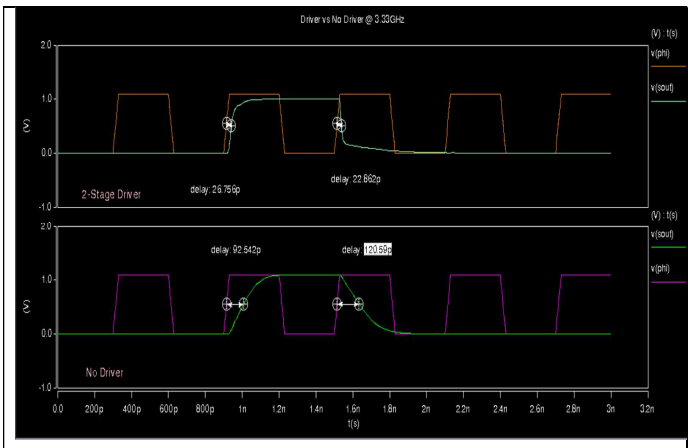


Figure 22. *Driver vs No Driver at f = 3.33 GHz*

We see that with upsizing, the delay reduced with effort-based sizing, from the graph, we have the following information about delay:

Table 9. *Delay comparison with Load*

Type	Delay Rising Edge (ps)	Delay Falling Edge (ps)
With Driver	26.756	22.862
No Driver	92.542	120.59
Comparison	28.85%	18.66%

This means that, with this resizing factor, we effectively minimized the delay between SOUT and PHI (or CIN), while maintaining the voltage level of SOUT. Further upsizing will reduce the max output of SOUT and distort the circuit's behaviors.

Since every gate is default to be compared with inverter, this is because the inverter has uniform electrical and logical effort. Most of all, it contains only 1 of each type of FET, making it ideal to compare with others. Follow our graphs, we have the following observation:

Table 10. *Table 9 Compare with Inverter*

Type	Delay Rising Edge (ps)	Delay Falling Edge (ps)
With Driver	26.756	22.862
No Driver	92.542	120.59
Inverter no Load	1.1	0.758

Since we are given the formula: $d = d_{abs}/T$. This is because we define delay to be "multiple of unloaded inverter delay (T)". Where d_{abs} is the t_{pd} and t_{pd} is given by:

$$T_{pd} = (t_{pdr} + t_{pdf})/2 \qquad \textit{(average of delay)}$$

Thus, between the two edges, we have the average delay of:

With Driver: (26.756 + 22.862)/2 = 24.809ps

No Driver: (92.542 + 120.59)/2 = 106.566ps

Inverter Unloaded: (1.1 + 0.758)/2 = 0.929ps

Thus, according to the definition of delay (lecture 4 slide 9), delay of the path from CIN to SOUT is given by the following formula:

$$D = D_{path}/T_{inverter} \qquad (1)$$

Table 11. *Path Delay by CSCOPE of 2-Stage*

Type	Calculation (ps/ps)	Delay (D) (parasitic of unloaded inverter)
With Driver	24.809/0.929	26.7
No Driver	106.566/0.929	114.7
Inverter no Load	0.929/0.929	1

Table 11 shows us that, the delay for the path designed in part A should have been 26.7, whereas we got 27.3. The difference is about:

$$\textit{(26.7 – 27.3)/26.7 = 2.23\%}$$

While the error is noticeable, this is to be expected, since our calculations are made based upon 3 assumptions that could skewed the results:

i. Input capacitance of HA is 9, that is taking both A and A_bar into account. This is because the prompt mentioned to ignore looping SOUT and its branches.

ii. The minimum sized inverter has the capacitance of 0.31fF through lab 1 extrapolation method.

iii. Rounding when computing y and w in part A. For example, Y = 3.47 ~= 4.

However, we see that the result is very close to each other, the assumption is compensated by over widening of the gate. Since, as stated in the prompt, we could potentially ignore SOUT and still yield the correct result. Ultimately, the result is not dead-on due to the assumptions listed above that took out some variables during the calculation of path delay.

C. What to do when we have 4-stage driver?

We want to find the C_{load} that will result in a 4-stage optimal delay is lower than that of 2-stage driver. I made the following measurements in CSCOPE to compare delay at SOUT. For the graphs below, ***my upsizing are (in order): 2, 4, 6, 8 (This is for testing purposes ONLY)***.

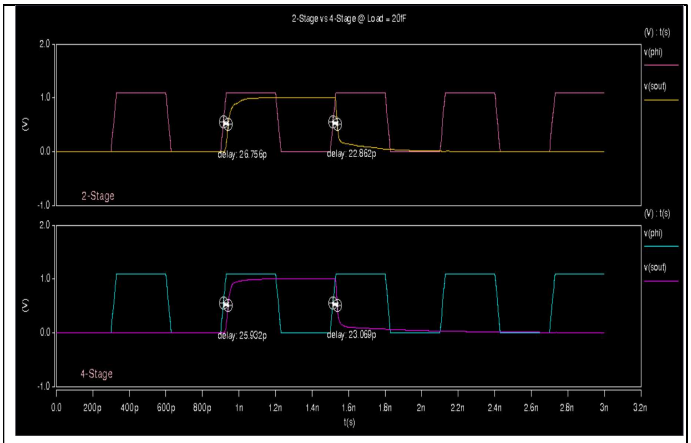


Figure 23. *2-Stage vs 4-Stage @ Load = 20fF*

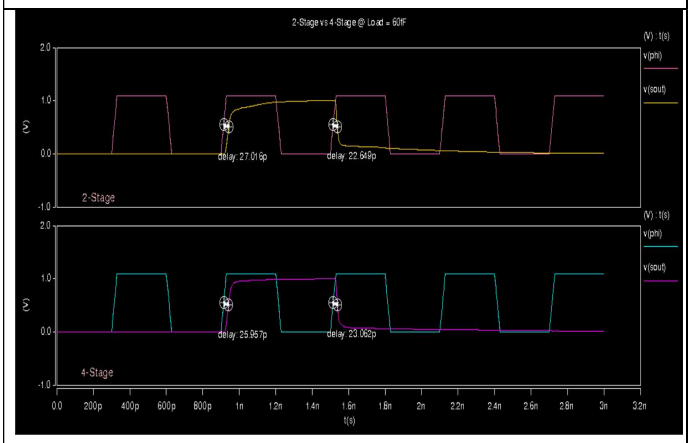


Figure 24. *2-Stage vs 4-Stage @ Load = 60fF*

We observe that as load capacitance increases, the rising edge of SOUT in 2-Stage decreases.

This means that, 4-Stage can always have less delay than 2-Stage (provided that we are upsizing correctly), however, the reason why we pick 2 instead of 4 or even larger number of stages is because that 2 perform the job just as well as larger number of stages. If this is the case, our 4-Stage parameters will be:

By solving for the Delay formula, setting it like this:

$$\textit{Delay 4-Stage} < \textit{Delay 2-Stage}$$

Then we have: (GH) is the same for both stages

$9*F\hat{} + 15 < 7*F\hat{} + 13 \Rightarrow 9*(GH)^{1/9} + 15 < 7*(GH)^{1/7} + 13$

Which yields: GH > 24672.78.

Therefore: $C_{loadfactor}/9 = 24672.78/(64/3) \Rightarrow C_{loadfactor} = 10408$

This tells us that $C_{load} = 10408*0.31 = 3226.635$ fF

---------- 4-Stage ----------

G = 4*(4/3)*(6/3)*(6/3)*1*1*1*1*1 = 4*4/3*2*2 = 64/3

H = 10408/9

P = 4+2+2+2+1+1+1+1+1 = 15

F = GH = (10408/9)*(64/3)= 24670.8

F^ = $F^{1/9}$ = 3.07

Delay (D) = 9*F^ + P = 42.63

Now, we can work backwards for size:

Z = C_{load} * G_z / F^ = (10408 * 1)/3.07 = 375.9 ~= 376

X = C_z * G_x / F^ = 375.9/3.07 = 122.2 ~= 122

Y = C_x * G_Y / F^ = (122.2 * 1)/3.07 = 39.8 ~= 40

W = C_Y * G_w / F^ = 39.8/3.07 = 12.9 ~= 13

---------- 2-Stage ----------

G = 4*(4/3)*(6/3)*(6/3)*1*1*1 = 4*4/3*2*2 = 64/3

H = 10408/9

P = 4+2+2+2+1+1+1 = 13

F = GH = (10408/9)*(64/3)= 24670.8

F^ = $F^{1/7}$ = 4.24

Delay (D) = 7*F^ + P = 42.68

Now, we can work backwards for size:

Y = C_x * G_Y / F^ = (10408/9 * 1)/4.24 ~= 273

W = C_Y * G_w / F^ = 273/4.24 ~= 65

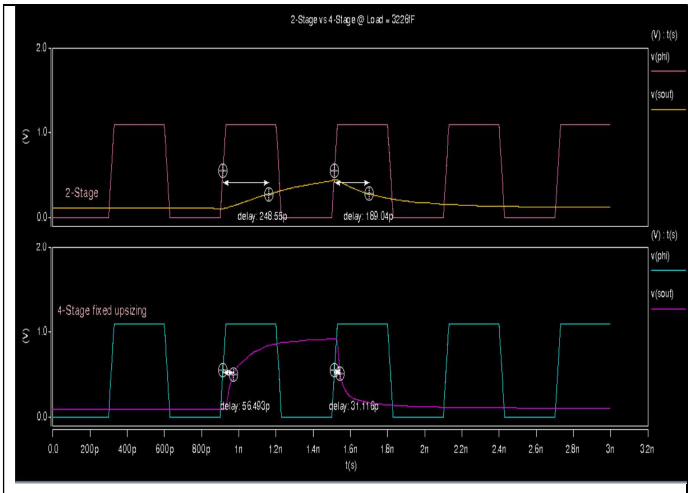


Figure 25. *2-Stage vs 4-Stage @ Load = 3226fF*

From Fig. 26, we have the following table of delay:

Table 12. *Path Delay by of 2-Stage vs 4-Stage*

Stages	T_{pdr} (ps)	T_{pdf} (ps)	T_{pd} (ps) (avg)
2-Stage	248.55	189.04	218
4-Stage	56.493	31.116	43.5

From Table 12, we see that the delay time of SPICE for 2-Stage does not match up with what we had expected it to be, this is because with only 2 stages, upsizing by a large amount will create a leap in each of the transistor. ***This big gap in upsizing cause the output to deteriorate. For example, we upsized Z by 376, which is way larger than 273 in Y, but we still have a good graph in the case of 4-Stage***. This is evident that upsizing needs to have stages and decreasing in order, if a big leap was made, the result will not conform with what we expected, due to the large difference between the two stages. ***Therefore, we want more stages for larger load capacitance, as we can upsizing each stage without leaving a big gap in between***.

Checking SPICE for 4-Stage with hand calculated give us:

Delay % error = (43.5 – 42.63)/43.5 = 2%

This means that our approximation using the above method is correct, and we see that 4-Stage delay is a lot slower than that of 2-Stage by a large margin (i.e: 43.5ps vs 218ps). Ultimately, I can conclude that C_{load} = 3226fF (or 10408*min inverter cap), theoretically, we should be seeing Delay (D) from 4-Stage is less than that of 2-Stage. While it might not have been the correct minimum capacitor needed, however, SPICE also proves that this value works (albeit 4-Stage is better than 2-Stage by a large margin, an unexpected occurrence if looking from the theoretical point of view).

VIII. SUMMARY

In conclusion, this lab teaches us how to better use SPICE tools and combining more gates than that of lab 1. In addition, we also got to see the importance of sizing and their effects on the circuit's delay and behaviors.

REFERENCES

[1] J. Butts and G. Sohi, "A static power model for architects," Proceedings 33rd Annual IEEE/ACM International Symposium on Microarchitecture. MICRO-33 2000, Dec. 2000.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

AppenDix A

Fig. 3

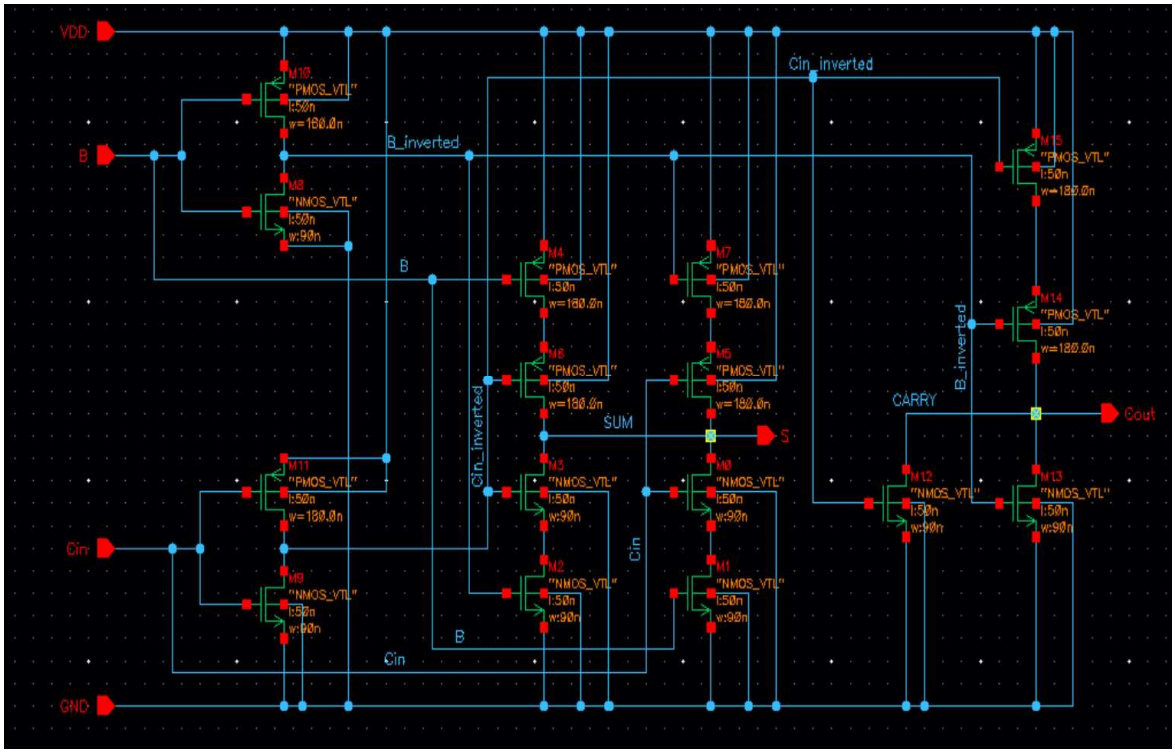


Fig. 6

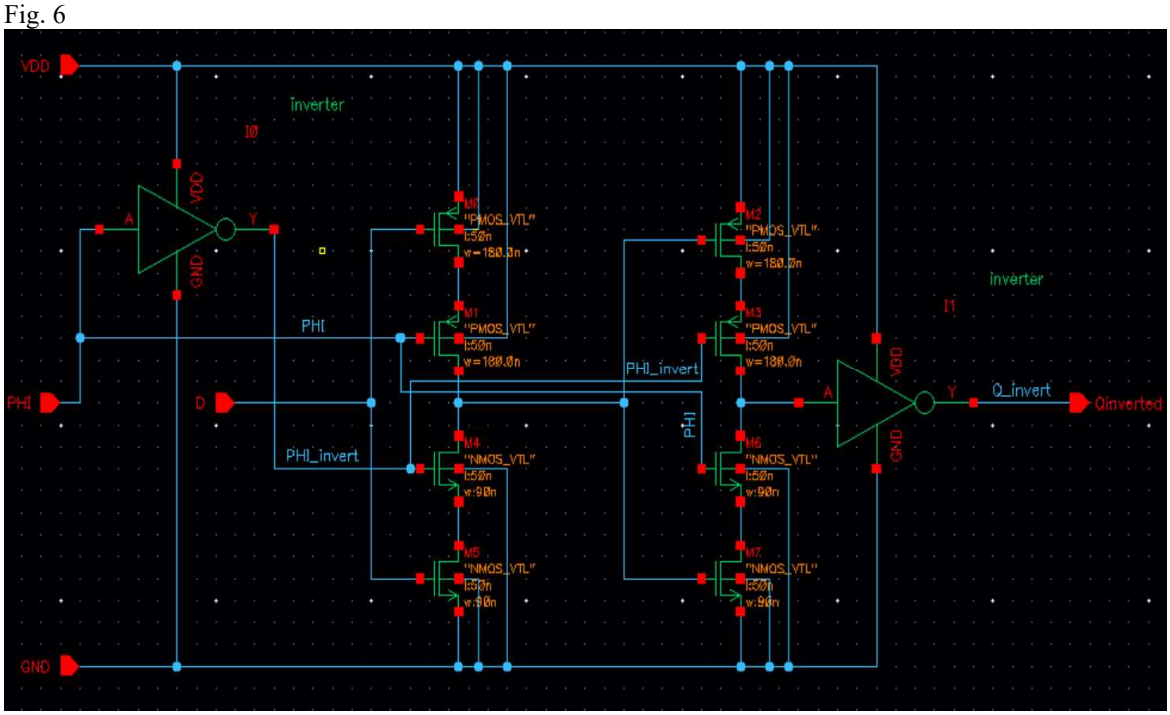
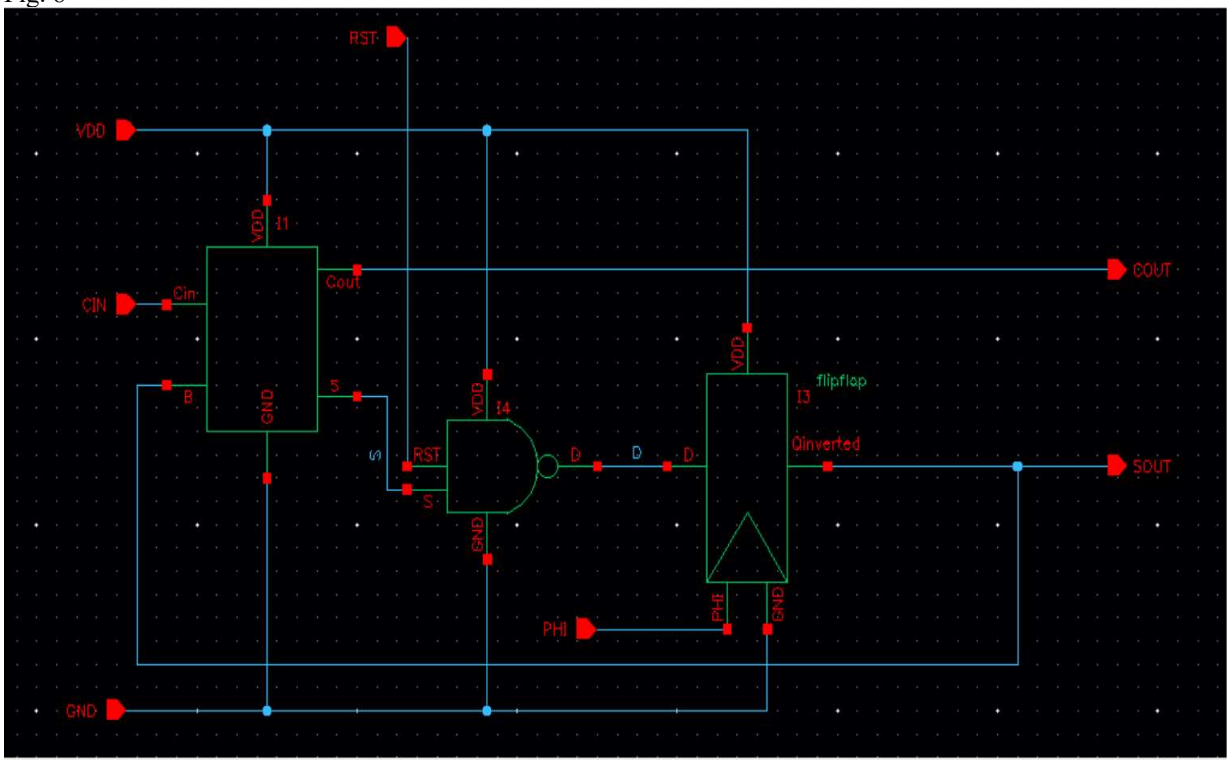


Fig. 8


RST
VDD
VDD
I1
CIN
Cin
Cout
COUT
B
GND
S
VDD
I4
RST
S
GND
D
D
D
VDD
I3
flipflop
Qinverted
SOUT
PHI
PHI
GND
GND