SAR ADC Logic Controller

Creating Data Flip Flop

GF 65nm process

Justin Alejandro

Setup Times	**(1)**	**(0)**
Tsu_DD	**25 ps**	**6 ps**
Tsu_opt	**41 ps**	**4 ps**
Thold	**6 ps**	**25 ps**
TCLK->Q(optimal)	**329 ps**	**331ps**
tD(optimal)	**370 ps**	**335 ps**
Height	**3.38 um**	**-**
Width	**11.7 um**	**-**
Area	**39.546 um^2**	**-**

*Cell design is optimized for height(instead of diffusion breaks) to minimize unused height for other standard cells.

Report Results

No vertical M2 was used. The only M2 present is for pins

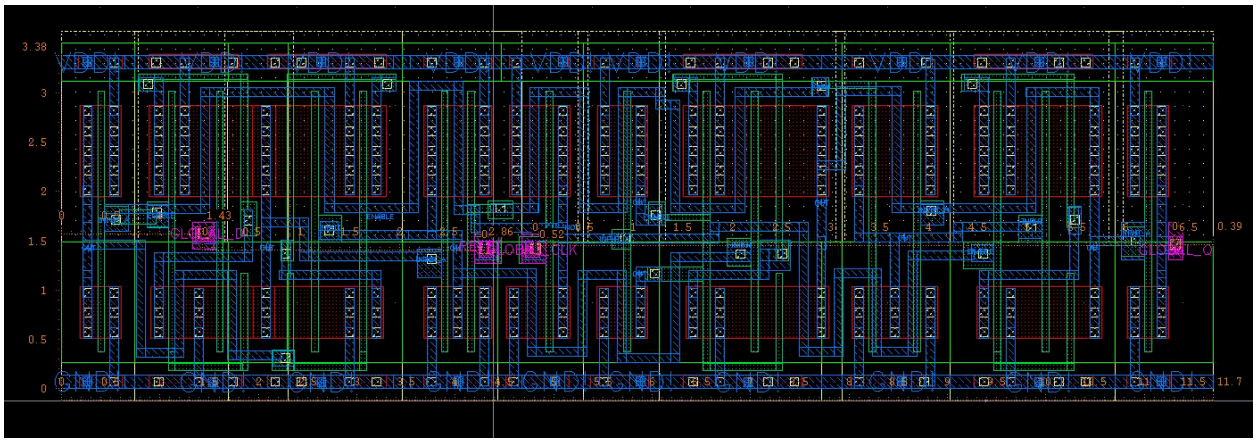

Figure 1. *Overall layout*

My pins are all separated by an increment of 0.26um with JX and JZ borders being 0n* .26um+.13um my height is 3.38um which is an odd multiple of 0.26um. My overall width is 11.7um which is multiple of 0.26um pin pitch

end>1.43>D>2.86>Reset>.52>Globalclk>6.5>Q>.39>end

Pitch rules:

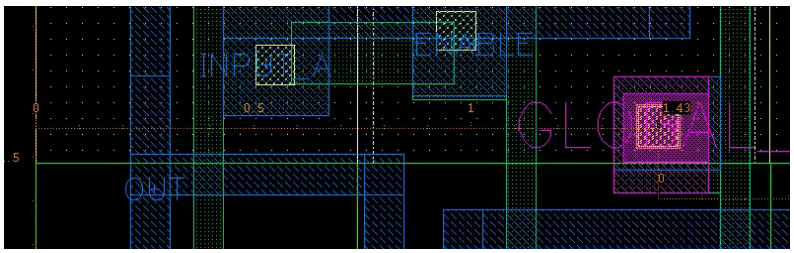

Figure 2. *Outside to Global_Q margin of 1.43*

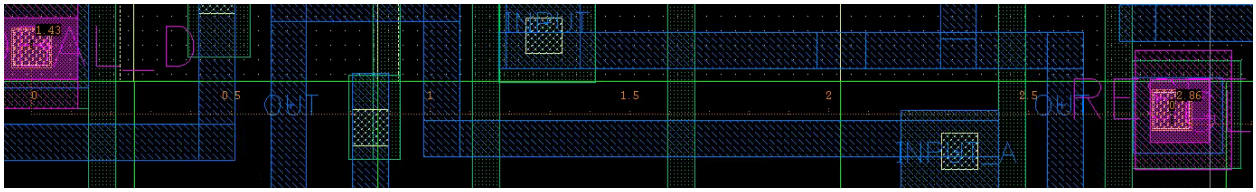

Figure 3. *Global_D to RESET margin of 2.86*

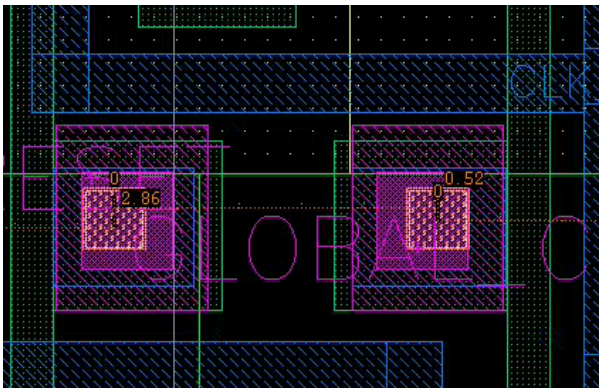

Figure 4. *RESET to Global_CLK margin of 0.52*

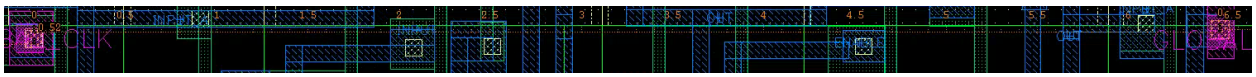

Figure 5. *Global_CLK to Global_CLK margin of 6.5*

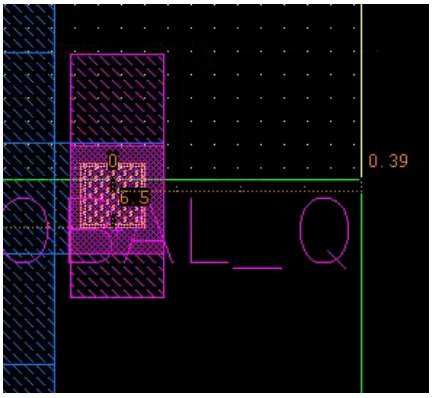

Figure 6. *Global_Q to Edge margin of 0.39*

DRC Test

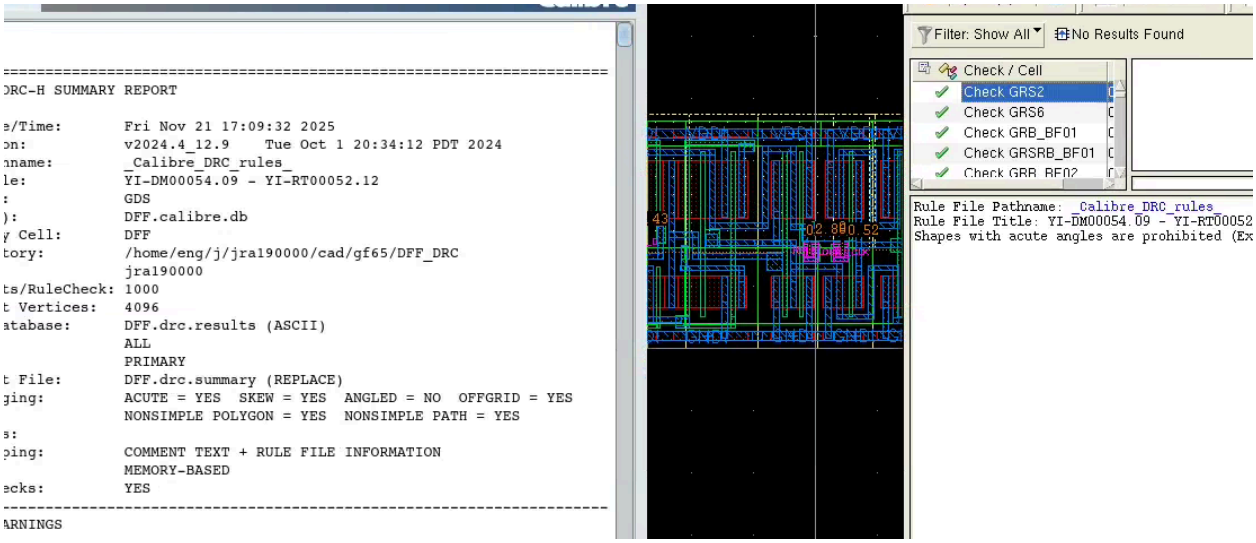

Figure 7. *DRC test*

LVS

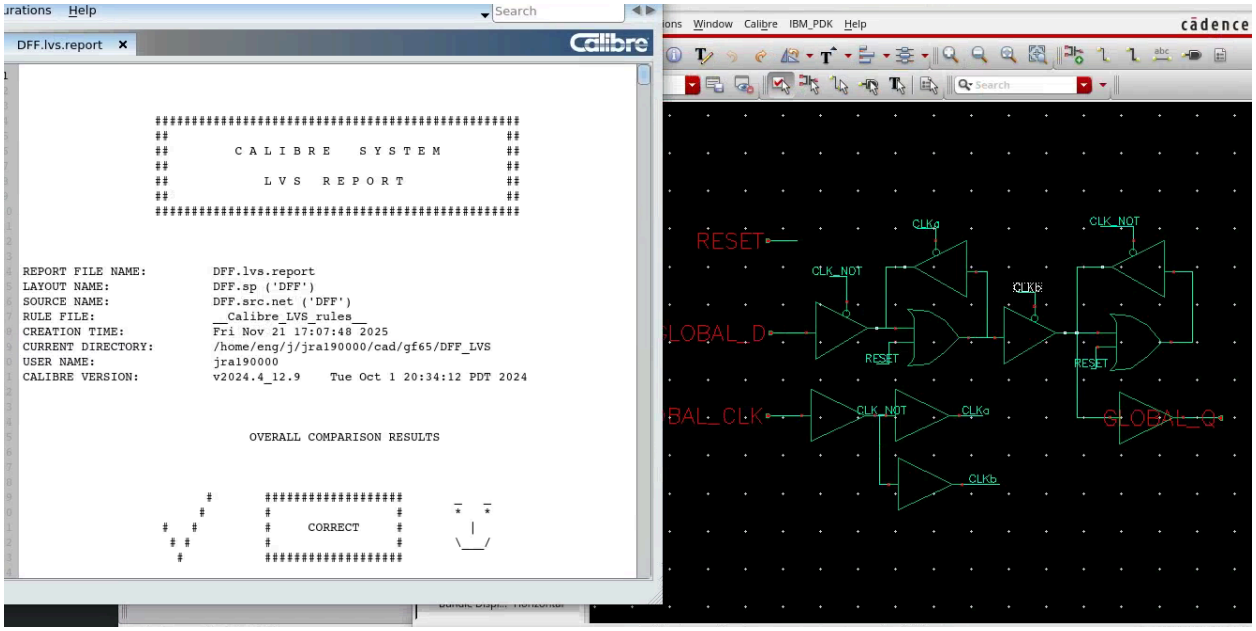

Figure 8. *LVS test*

HOW I CAME UP WITH THE LAYOUT- (I explain journey through a mistake)

The original design goals were to reduce my cell area by minimizing diffusion breaks and cell width. Additionally, I would need to try to meet the cell height of my cells in project 4, which I successfully optimized the DFF to be 3.38um tall. This was already very short considering the design considerations. We are required to minimize the amount of vertical metal 2 used and not allowed to use any horizontal metal 2, which I used neither. I rarely used horizontal poly, unless it was used to optimize a gate itself as it was shown in class. I didn't use any horizontal poly to be an extension for my metal 1 routing, unless it was near the gate input on a very rare occasion.

The layout plan was originally laid out in a schematic horizontal pattern to visually see how the lined up gates would connect.

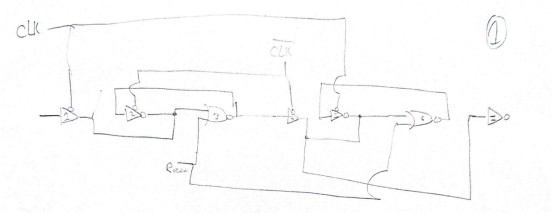

Figure 9. *Original DFF schematic*

By treating them as nodes, I could see how the amount of connections needed to complete the DFF could be reduced

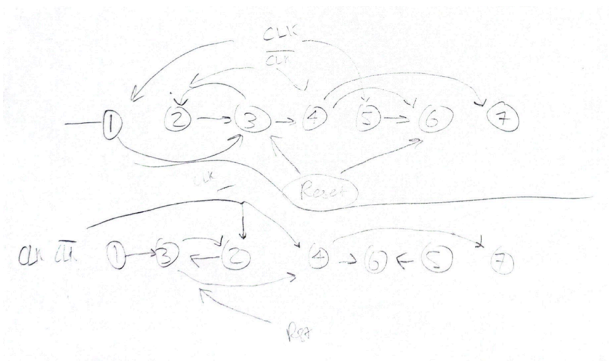

Figure 10. *Complex network analysis*

I made considerations like reducing diffusion breaks or implementing more folding, but those considerations came at the cost of more metal 1 wiring bypassing gates which ultimately, made the gate taller. At all costs, my biggest priority was to make my design to achieve the smallest height possible and maintain the cell height of project 4 cells. To do this I found maintaining a similar pattern to the traditional DFF schematic where inputs and outputs of the gates are next to

each other is the most optimal (roughly). I had analyzed tradeoffs of different permutations of gates that could be interchanged, but found the most optimization came from reducing the clock and reset wiring. For example, moving the nor2 gate closer to reduce the reset wiring, with the tradeoff of missing the opportunity of fusing the diffusions of the tri-state inverter.

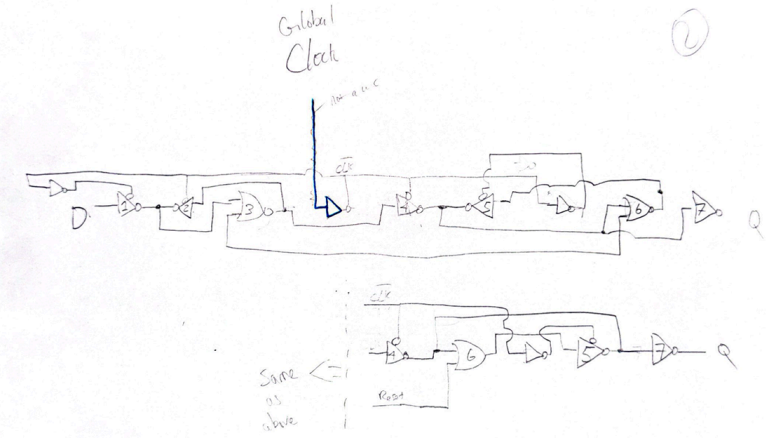

Figure 11. *System optimization*

The charm of my first architecture is from the utilization of a second stage parallel driven buffer layout, where I could put inverters on the outer sides of the layout to produce the final clk'(pos) to enable the outer tristate inverter (found to be wrong logic later & later fixed).

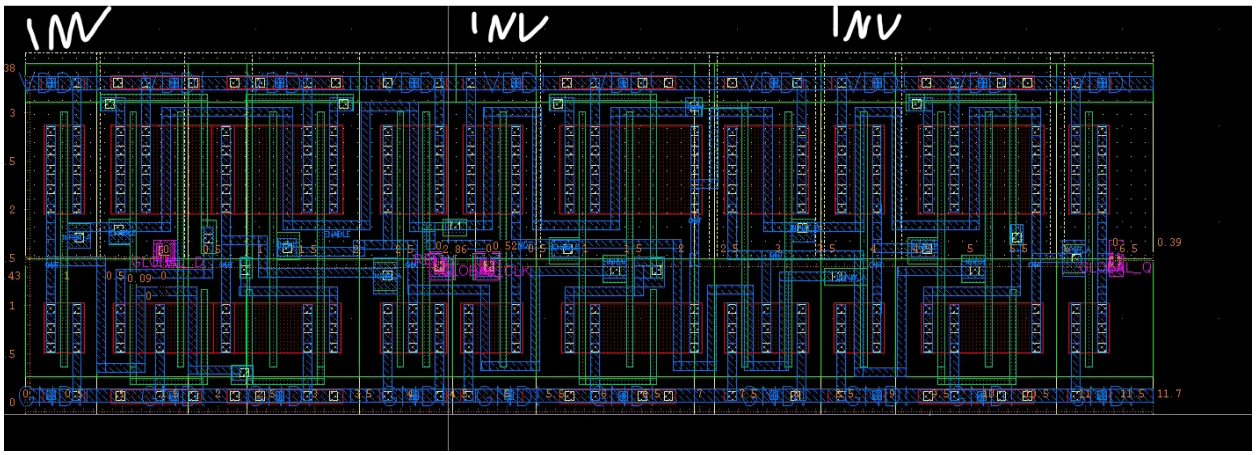

Figure 12. *Inverter layout locations*

This layout choice would allow me to reduce the clk signals metal wiring going through the layout by 1. Meaning I no longer need a clk' to spaghetti through the layout. The reason why this was important is because the maximum amount of metals 1 layer I can channel through a gate was two excluding input and output connected to the gate itself which wasn't as much of a limiting factor.

An issue arose when I originally designed the DFF. I made a mistake in the logic of the tri-state inverters which were incorrectly connected to the wrong clock signal. Ultimately, that made my DFF work on a falling edge. To remedy this issue to make it work correctly on a rising edge I had to reroute the DFF. Luckily, my pre-existing architecture allowed me to make minor routing changes to fix the logic of ¾ of the tristate inverters. Unfortunately, there was still an issue with one of them. It needed a clk' that was positive which required an inverter to invert the clk_not signal. Essentially, I needed to reposition an inverter to a new position.

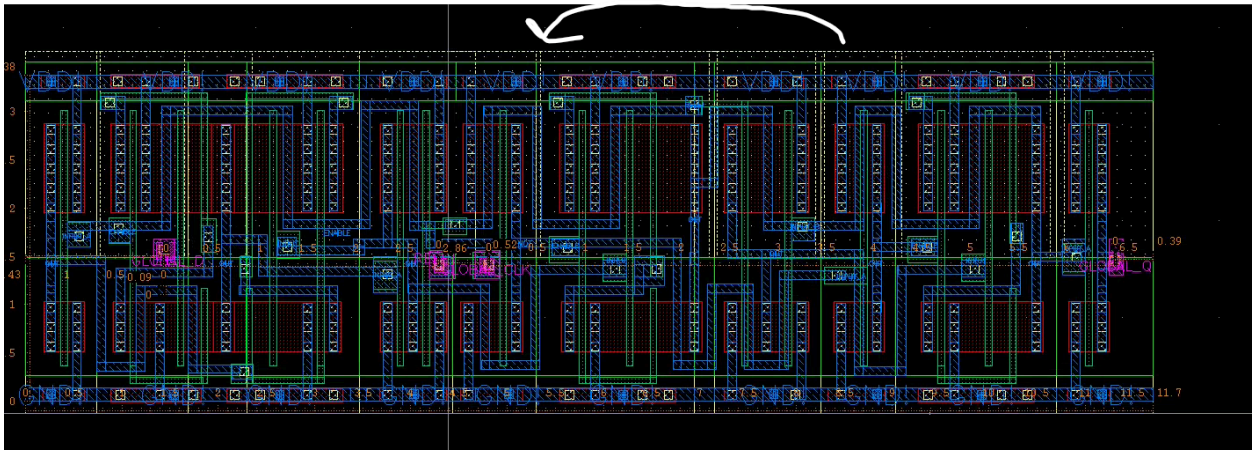

Figure 13. *Layout inverter relocation*

Unfortunately, in my new setup I would need 3 metal_1 layers to pass the inverter. Originally this wasn't an issue because the inverters were on the outer edges of the DFF. Additionally, I couldn't keep the inverter in the original position because I would have to backtrack the metal 1 clk' output backwards to the tri-state inverter and cause more metal that can't fit (as seen in figure). In order to remedy this issue, I decided to explore the possibility of changing the inverter to make this possible.

VLSI theory inverter output split drive- testing normal double inverter vs double split.

So the reason why I introduced a split inverter is because the normal traditional inverter layout has limitations for the context of my DFF and the project as a whole. If I expand the width of the inverter gate I can manage 2 metal_1 wires to pass through with this current height. This would not work for the INV in the new location because 3 wires need to bypass the gate. By creating a split inverter layout (contingent it works) it can allow more flexibility with allowing gates to go through.

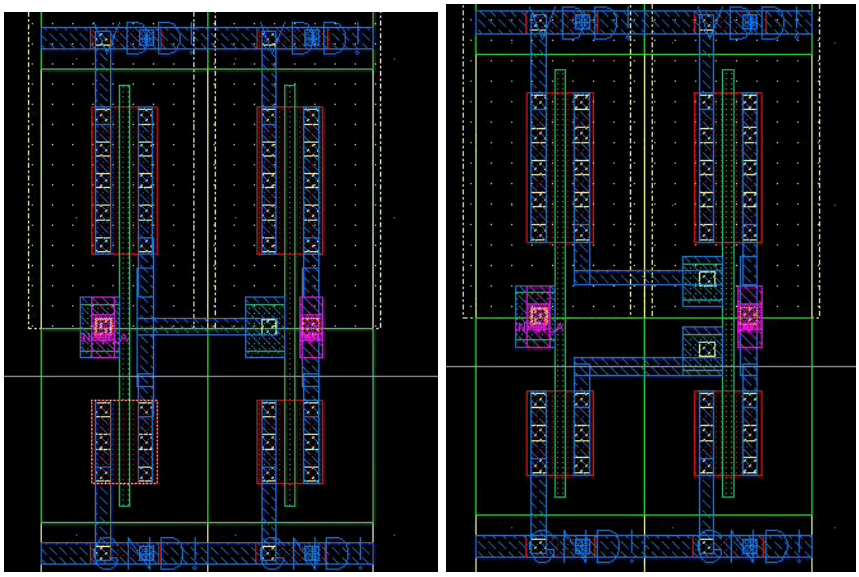

Figure 14. *a) original double INV b) double split INV*

To ensure it works I performed DRC,LVS, and logic tests to ensure it works like a normal inverter.

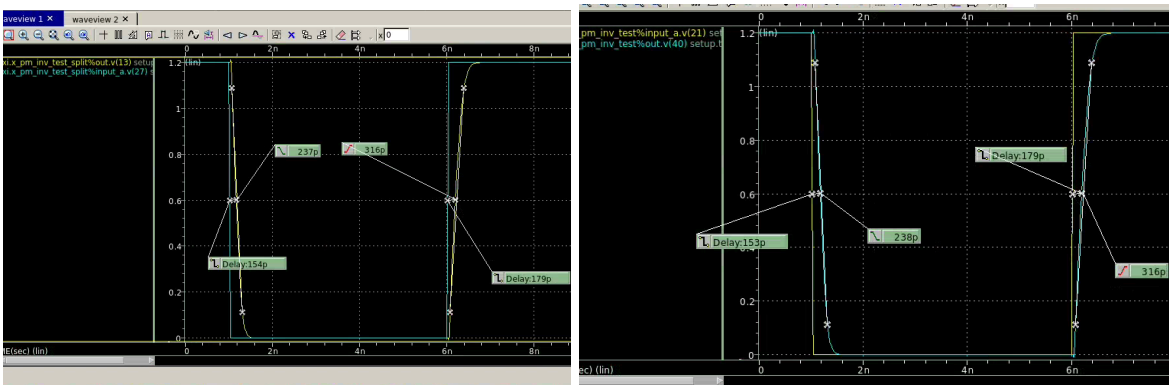

Figure 15. *Hspice test comparison match*

Luckily it does and I can implement the new working rising edge DFF while resembling my original layout architecture. Based on constraints I'm convinced this is the smallest you can make besides squeezing a few um between RX drawings.

The new DFF now works on a clock rising edge rather than a falling edge.

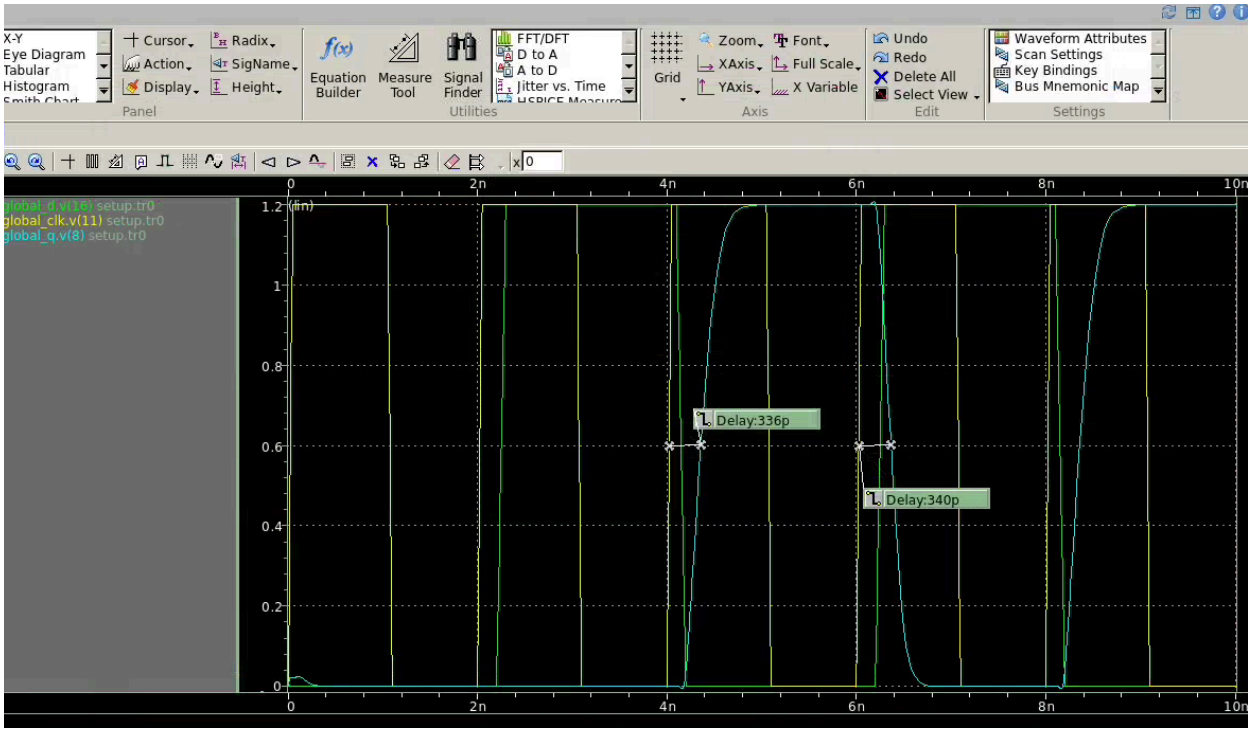

Figure 16. *Correct DFF logic*

How I found setup times

Conditions: input slew rate (10-90) is 35ps. A load capacitance of 45fF.

Tsu_dd

To find Tsu_dd, I had the global clock signal start at 1ns and performed a sweep to test various setup times by measuring picoseconds before the clock signal. Additionally, I would measure the output, Q signal that corresponded with the signal. This way I was able to measure the drop dead signal by looking at the latest the d signal needed to start for output stability. This was done by checking in wave view, and also then analyzing the setup from .measure in the setup file.

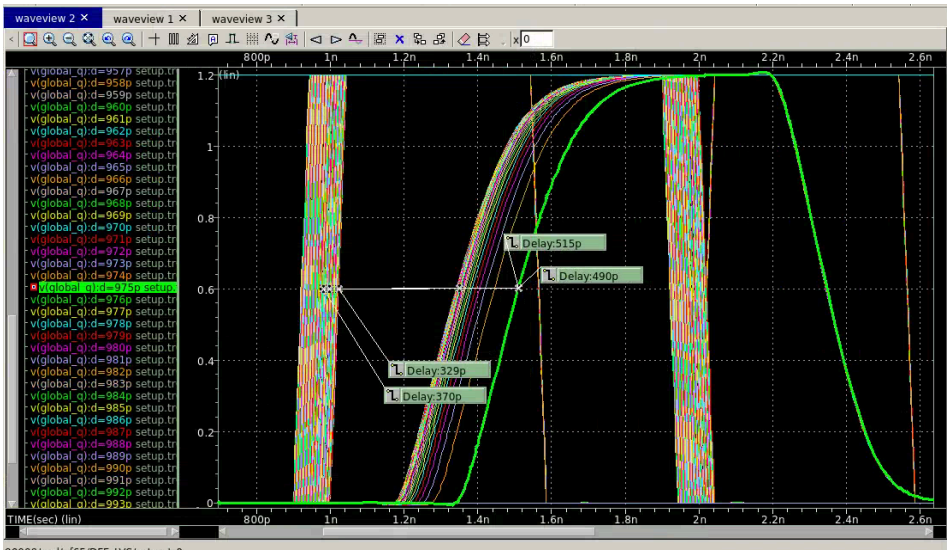

Figure 17. *DFF **off to on** dead drop setup*

The d input occurs at 975ps. Therefore the dead drop is 25ps

Tsu_opt

To find the optimal setup time we found the .measure signal delay(50-50) from input D and the output. Then we plotted the measurements and found the optimal minimum time. I found it to be at d=959ps, therefore the setuptime is 41ps seconds before clock time.

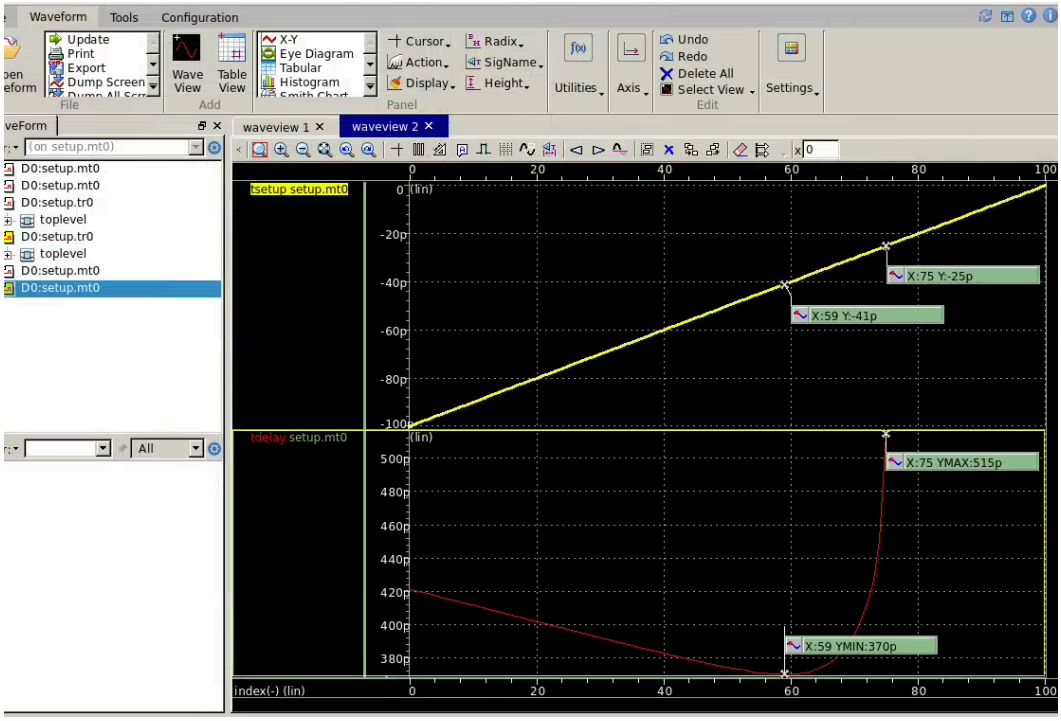

Figure 18. *DFF **off to on** setup optimization*

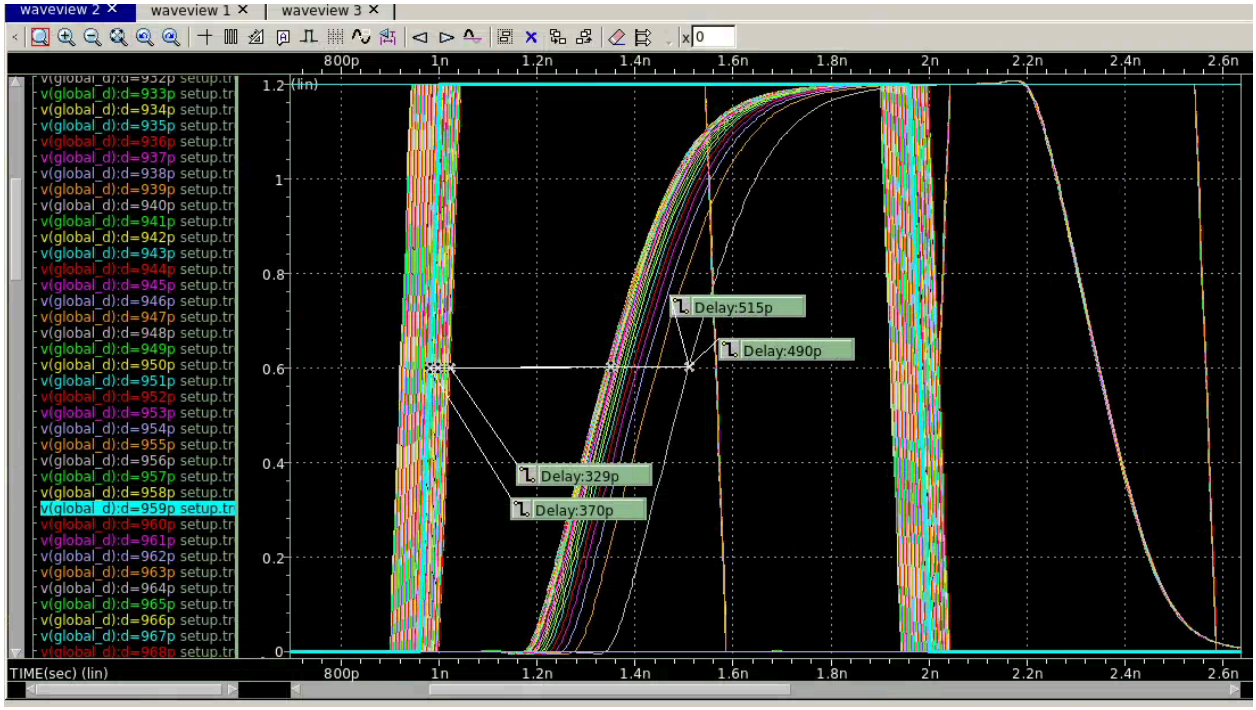

Figure 19. *DFF **off to on** waveview opt test*

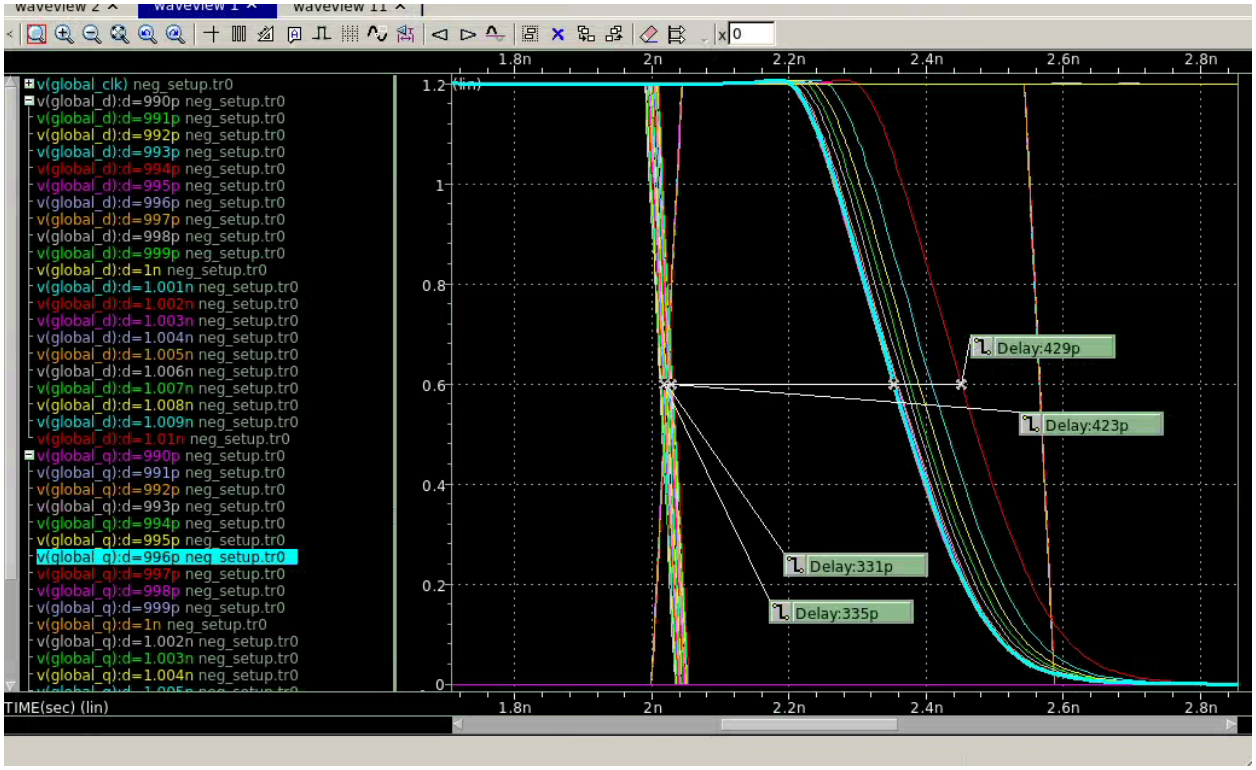

Figure 20. *DFF* ***on to off*** *setup waveview*

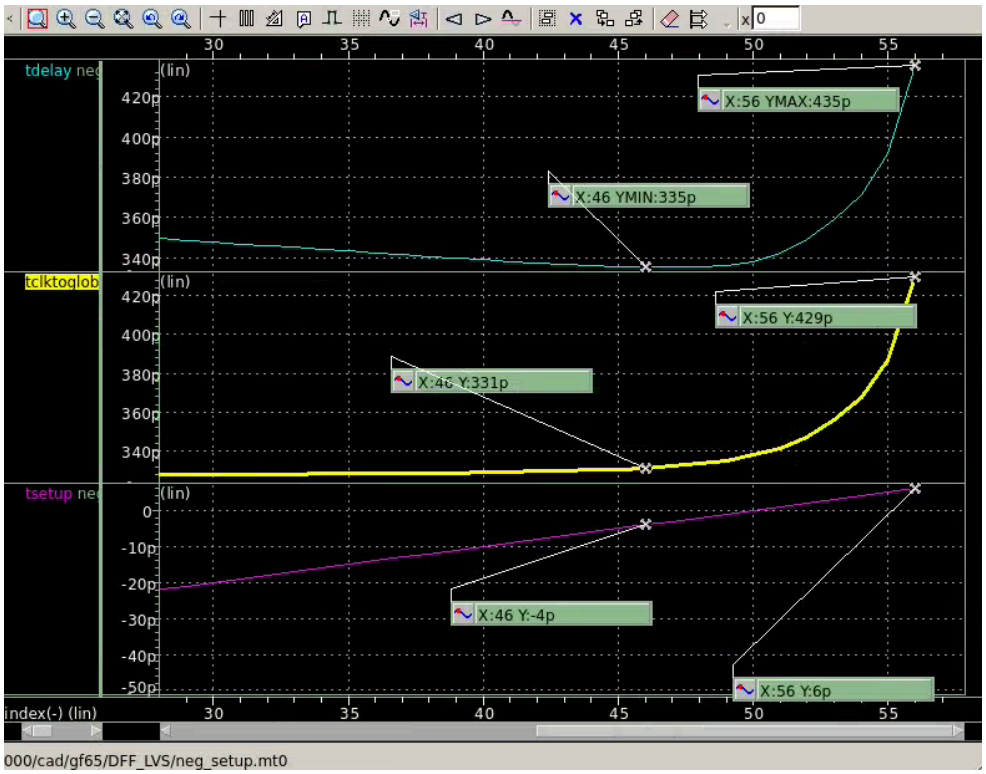

Figure 21. *DFF* ***on to off*** *setup optimization*

Thold- using Tsu_dd(1) = |Thold(0)| & |Tsu_dd(0)| = |Thold (1)| , we can find the Tsu_dd(0) and hold the Thold(0)|.

Tsu_dd(1) = |Thold(0)| => 25ps = |Thold(0)|

Tsu_dd(0) = |Thold(1)| => Tsu_dd(0) = 6ps

Tclk->Q is measured by taking delay(50-50) from clock to q output

Tclk->Q(1)opt = 339ps

Tclk->Q(1)dd = 490ps

Tclk->Q(0)opt = 331ps

Tclk->Q(0)dd =429ps

tD - delay is found by adding setup and clk->q time and verified by finding delay from D to Q

tD(1)opt = 370ps = 25+329

tD(1)dd = 515ps = 41+490

tD(0)opt = 315ps = 4+331

tD(0)dd =429ps = (-6) + 435

Just proving reset logic below. Even though d signal is high reset brings it back to 0 seen in Figure 21 below

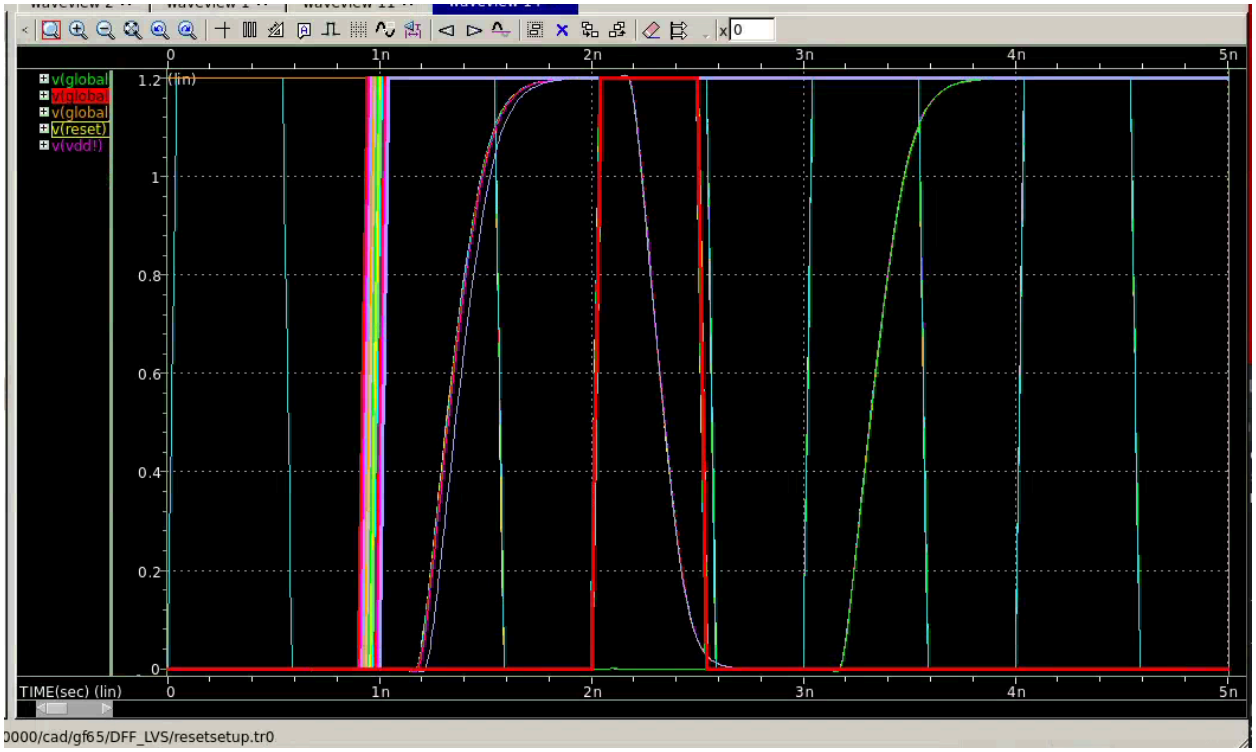

Figure 21. *DFF reset logic proof*

Conclusion-

It is found that the second stage parallel inverter setup saves the height and size overall, but may not have the most optimal delay times for the DFF. For the context of my SAR, it might've been a good consideration. Otherwise, my design only uses a few DFF while the rest is combinatorial logic, and additionally we aren't graded on design speed.

Spice testing setup files

Code positive

```
$transistor model for positive edge
.option post runlvl=5
.include
"/proj/cad/library/mosis/GF65_LPe/cmos10lpe_CDS_oa_dl064_11_20160415/models/YI-SM00
030/Hspice/models/design.inc"
```

```
.include "DFF.pex.sp"
xi GND! GLOBAL_Q VDD! GLOBAL_D RESET GLOBAL_CLK DFF
vdd VDD! GND! 1.2v
vclk GLOBAL_CLK GND! pulse(0 1.2 0ps 44ps 44ps 500ps 1000ps)
vr RESET GND! pwl(0ns 0v)
.param d=950ps
vd GLOBAL_D GND! pwl(0ns 0v d 0v 'd+44ps' 1.2v '1ns+d' 1.2v '1ns+d+44ps' 0v)
cq  GLOBAL_Q GND! 45f
.tran 5ps 5ns sweep d 900ps 1000ps 1ps
.measure tsetup trig v(GLOBAL_CLK) val=0.6 rise=2 targ v(GLOBAL_D) val=0.6 rise=1
.measure tclktoGLOBAL_Q_rise trig v(GLOBAL_CLK) val=0.6 rise=2  targ v(GLOBAL_Q)
val=0.6 rise=1
.measure tdelay param='(abs(tsetup) + tclktoGLOBAL_Q_rise)'
.end
```

CODE NEGATIVE setup.sp

```
$transistor model for negativeedge
.option post runlvl=5
.include
"/proj/cad/library/mosis/GF65_LPe/cmos10lpe_CDS_oa_dl064_11_20160415/models/YI-SM00
030/Hspice/models/design.inc"
.include "DFF.pex.sp"
xi GND! GLOBAL_Q VDD! GLOBAL_D RESET GLOBAL_CLK DFF
vdd VDD! GND! 1.2v
vclk GLOBAL_CLK GND! pulse(0 1.2 0ps 44ps 44ps 500ps 1000ps)
vr RESET GND! pwl(0ns 0v)
.param d=950ps
vd GLOBAL_D GND! pwl(0ns 1.2v d 1.2v 'd+44ps' 1.2v '1ns+d' 1.2v '1ns+d+44ps' 0v 3ns 0v
3.044ns 1.2v 5ns 1.2v)
cq  GLOBAL_Q GND! 45f
.tran 5ps 5ns sweep d 950ps 1006ps 1ps
.measure tsetup trig v(GLOBAL_CLK) val=0.6 rise=3 targ v(GLOBAL_D) val=0.6 fall=1
.measure tclktoGLOBAL_Q_fall trig v(GLOBAL_CLK) val=0.6 rise=3  targ v(GLOBAL_Q)
val=0.6 fall=1
.measure tdelay param='(abs(tsetup) + tclktoGLOBAL_Q_fall)'
.end
```

CODE RESET setup

```
$transistor model for rest proof
.option post runlvl=5
.include
"/proj/cad/library/mosis/GF65_LPe/cmos10lpe_CDS_oa_dl064_11_20160415/models/YI-SM00
030/Hspice/models/design.inc"
.include "DFF.pex.sp"
xi GND! GLOBAL_Q VDD! GLOBAL_D RESET GLOBAL_CLK DFF
vdd VDD! GND! 1.2v
vclk GLOBAL_CLK GND! pulse(0 1.2 0ps 44ps 44ps 500ps 1000ps)
vr RESET GND! pwl(0ns 0v 2ns 0v 2.044ns 1.2v 2.5ns 1.2v 2.544ns 0v 5ns 0v)
.param d=950ps
vd GLOBAL_D GND! pwl(0ns 0v d 0v 'd+44ps' 1.2v '1ns+d' 1.2v '1ns+d+44ps' 1.2v)
cq  GLOBAL_Q GND! 45f
.tran 5ps 5ns sweep d 950ps 975ps 5ps
.measure tsetup trig v(GLOBAL_CLK) val=0.6 rise=2 targ v(GLOBAL_D) val=0.6 rise=1
.measure tclktoGLOBAL_Q_rise trig v(GLOBAL_CLK) val=0.6 rise=2  targ v(GLOBAL_Q)
val=0.6 rise=1
.measure tdelay param='(abs(tsetup) + tclktoGLOBAL_Q_rise)'
.end
```