

EECT/CE 7325 VLSI Design

7nm

128b Serial-to-Parallel

Converter

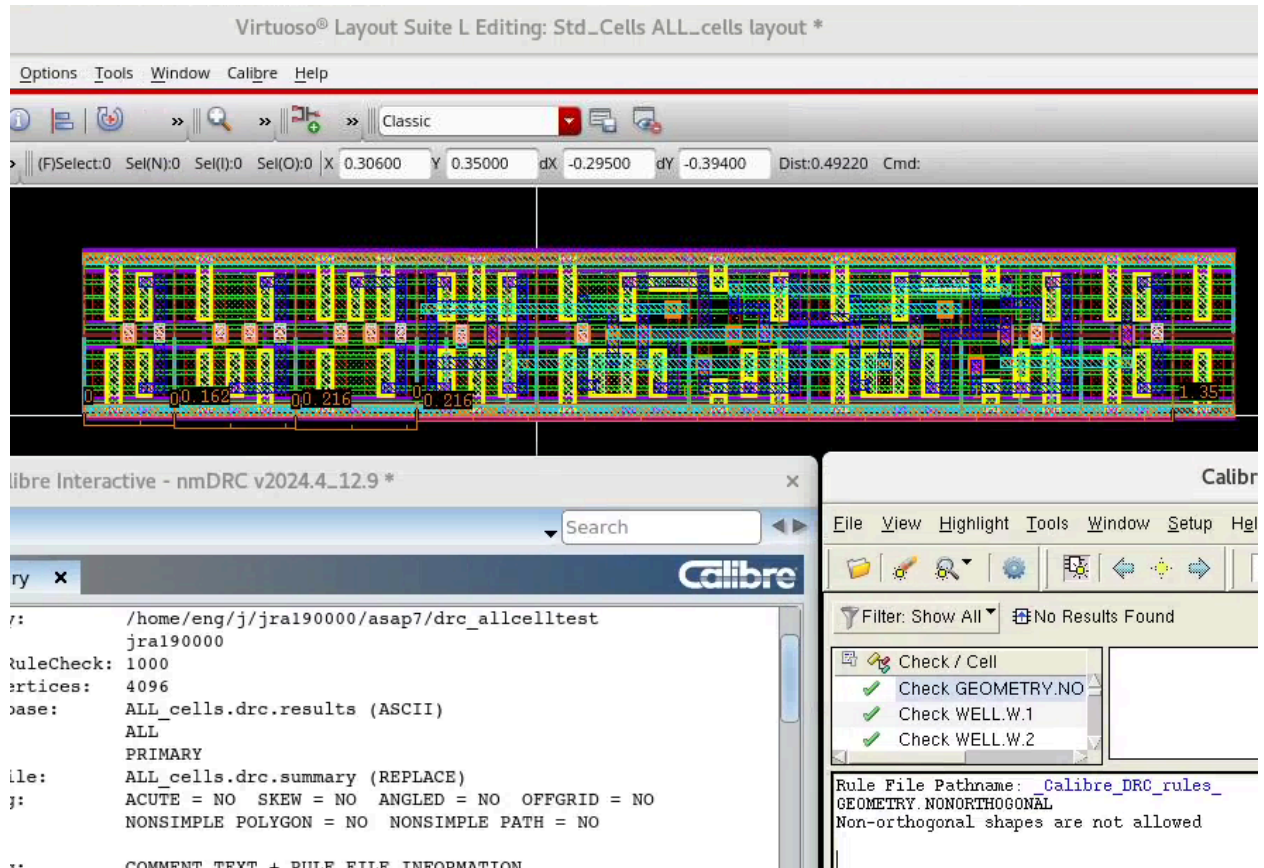
Project

Justin Alejandro

Reminder that this assignment was a feasibility project to test out 7nm ASAP7 and the PnR is done manually for the project

Report Results

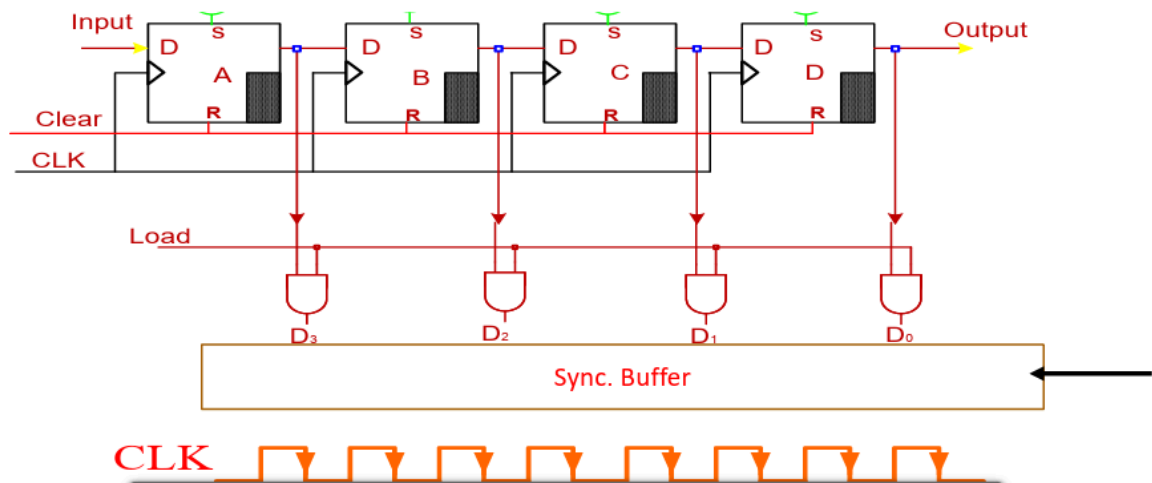
ALL CELLS ON DRC



The cells are listed as with dimensions,drc,lvs, and wave form, the .lib file is in zip file inside the synopsys folder named CELLS.lib. Waveforms simulations conducted using 15fG load capacitance, with an input slew rate of 12ps(0-100% and 100-0%). Pdiff and ndiff have 3 fin heights

Also had to update cell are, for some reason it was barely off, but DFF, NAND/NOR2, and inv is .388, , 0.062208, 0.046656 respectively

After getting my 4 main cells for my project its time for the design of the Serial to parallel converter,



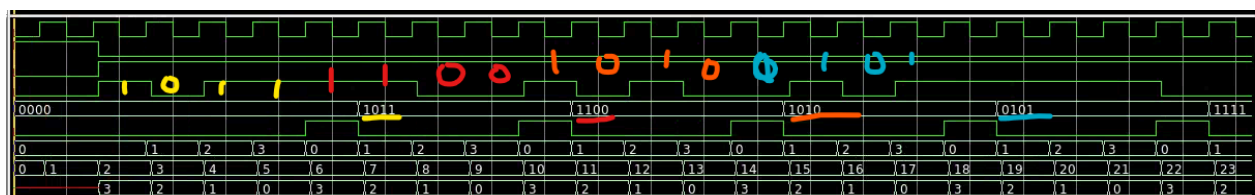
Essentially the Serial to parallel converter will take a serial input that is sequentially inserted every CLK cycle and transform it into a parallel format.

Because we will be manually placing and routing for this project, it would be beneficial to separate portions of the S2P that can be reproduced. For this project I've considered the different partitions to be the clk_counter used to count DFF transitions for load, the dff register chain that the serial bits propagate into, the enable gates that are made of ands (NAND+INV), and our output register that updates on sync.

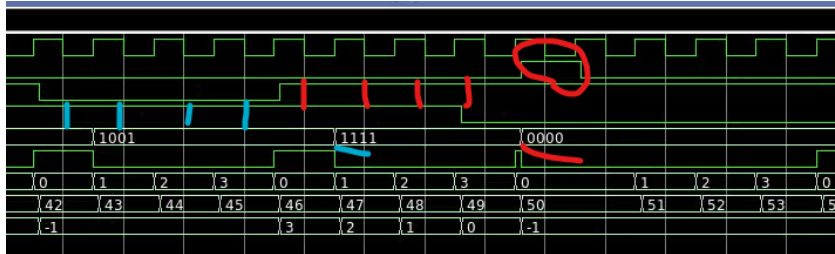
By creating these partitions, I can easily scale and verify the logic, additionally I can ensure what gates are being synthesized. This way I have more control and predictability on the gates that are being produced before the manual place and route stage.

First I made a 4bit S2P example just to analyze the functionality before scaling.

These sections are:



My model sim simulation is working properly. Yes there is a delay from a control signal, but for the sake of simplicity it will be allowed so we don't need a faster clk just for a control signal. Additionally as I scale, 1 clk delay from an input will be negligible, but is a consideration for future highspeed projects.



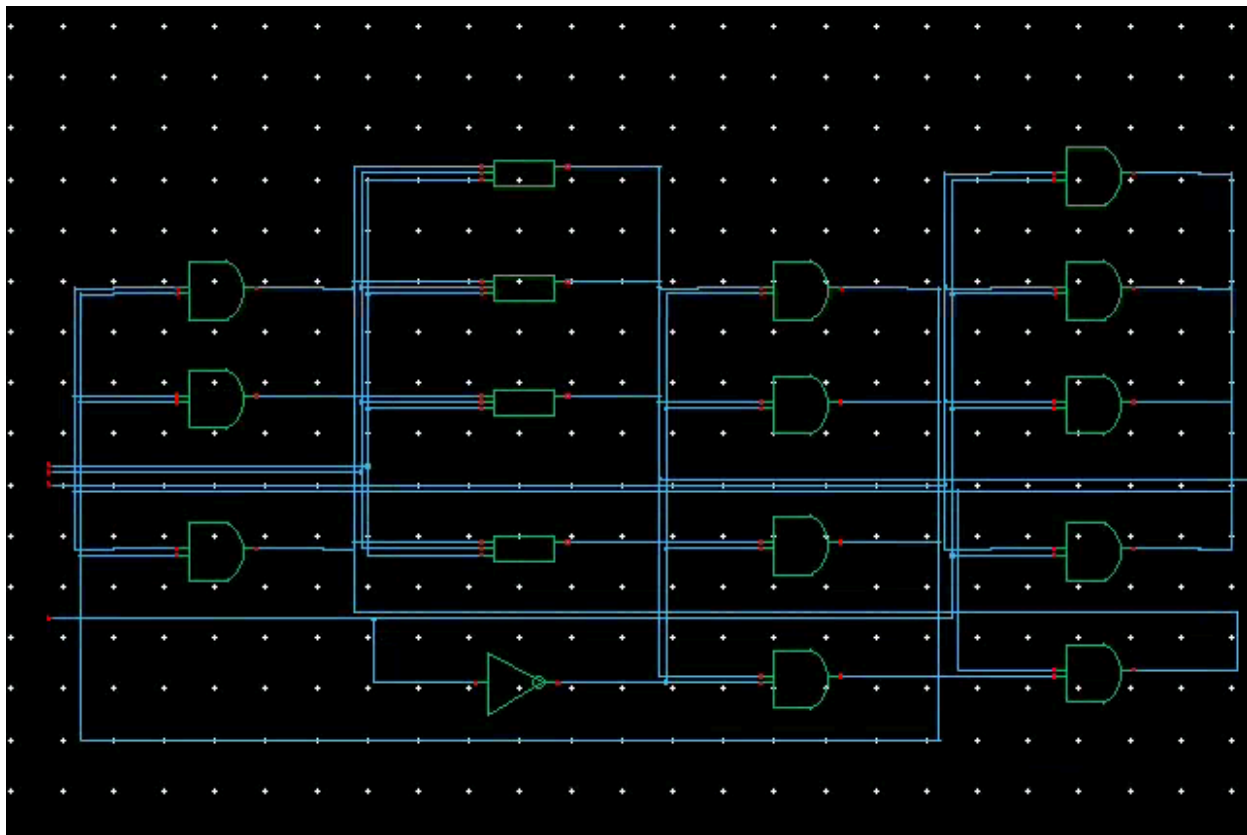
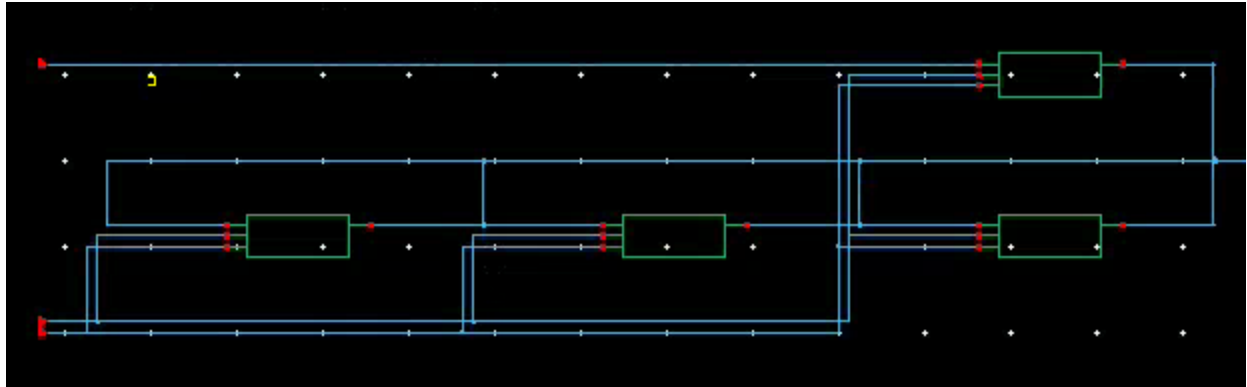
RESET LOGIC WORKING

u_out_reg/C21	nand2	library	1.000000
u_out_reg/C23	nand2	library	1.000000
u_out_reg/C25	nand2	library	1.000000
u_out_reg/C27	nand2	library	1.000000
u_out_reg/C29	nand2	library	1.000000
u_out_reg/C31	nand2	library	1.000000
u_out_reg/C33	nand2	library	1.000000
u_out_reg/C35	nand2	library	1.000000
u_out_reg/C37	nand2	library	1.000000
u_out_reg/C39	nand2	library	1.000000
u_out_reg/C41	nand2	library	1.000000
u_out_reg/C43	nand2	library	1.000000
u_out_reg/I_0	inv	library	1.000000
u_out_reg/u_out0/U3	inv	library	1.000000
u_out_reg/u_out0/q_reg	dff	library	7.000000 n
u_out_reg/u_out1/U3	inv	library	1.000000
u_out_reg/u_out1/q_reg	dff	library	7.000000 n
u_out_reg/u_out2/U3	inv	library	1.000000
u_out_reg/u_out2/q_reg	dff	library	7.000000 n
u_out_reg/u_out3/U3	inv	library	1.000000
u_out_reg/u_out3/q_reg	dff	library	7.000000 n
u_shift/u_dff0/U3	inv	library	1.000000
u_shift/u_dff0/q_reg	dff	library	7.000000 n
u_shift/u_dff1/U3	inv	library	1.000000
u_shift/u_dff1/q_reg	dff	library	7.000000 n
u_shift/u_dff2/U3	inv	library	1.000000
u_shift/u_dff2/q_reg	dff	library	7.000000 n
u_shift/u_dff3/U3	inv	library	1.000000
u_shift/u_dff3/q_reg	dff	library	7.000000 n
u_valid/C17	nand2	library	1.000000
u_valid/I_1	inv	library	1.000000
u_valid/U3	inv	library	1.000000
u_valid/valid_reg_reg	dff	library	7.000000 n
Total 43 cells			109.000000
design_vision>			

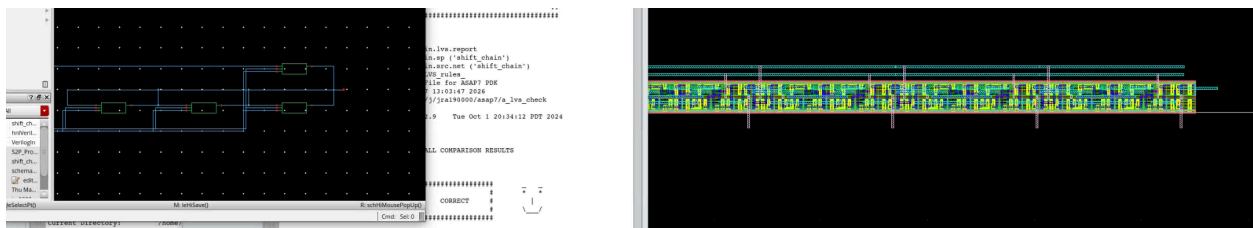
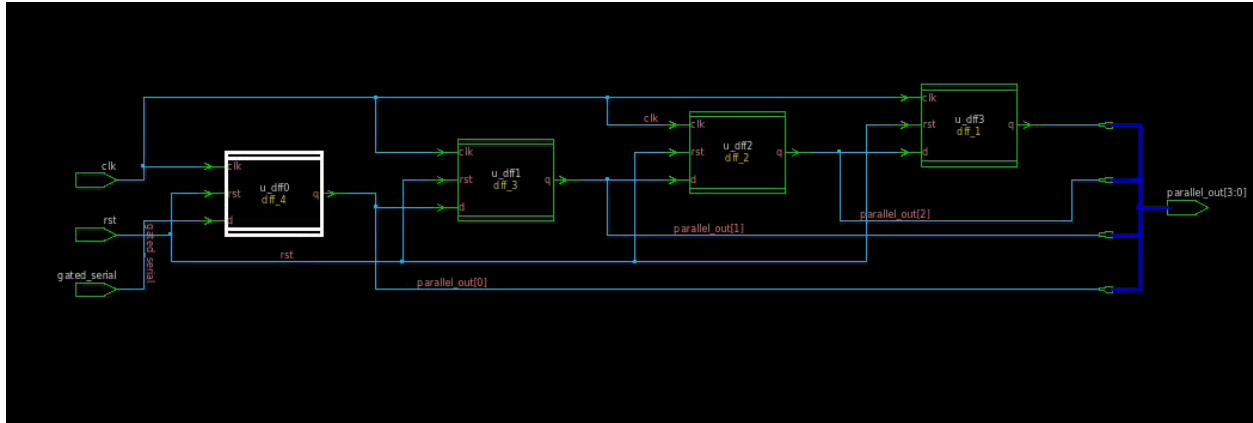
**i found out later i was using wrong

library.db

Did a design vision compile sanity check to see if it works and so I can do analysis for each of the modules. - Note I actually found it better to leave the module ungrouped to allow modular portions of the design to scale better as we got from 4bits to 16 bits

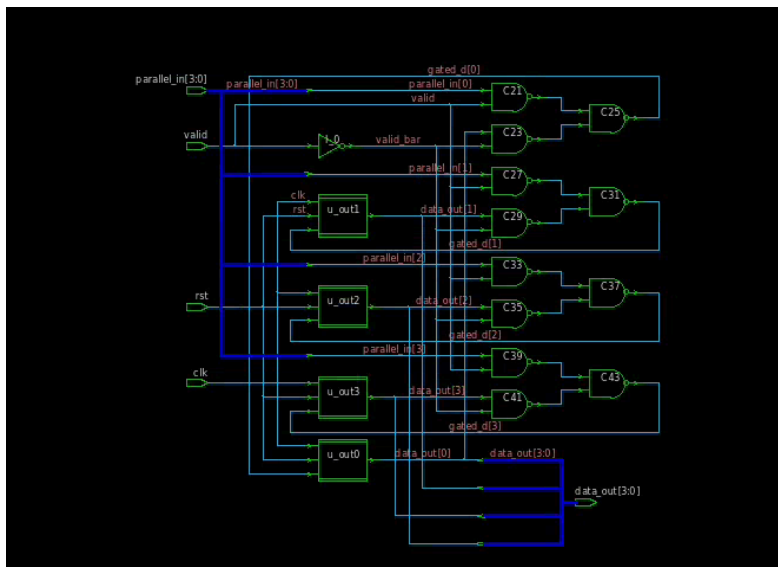


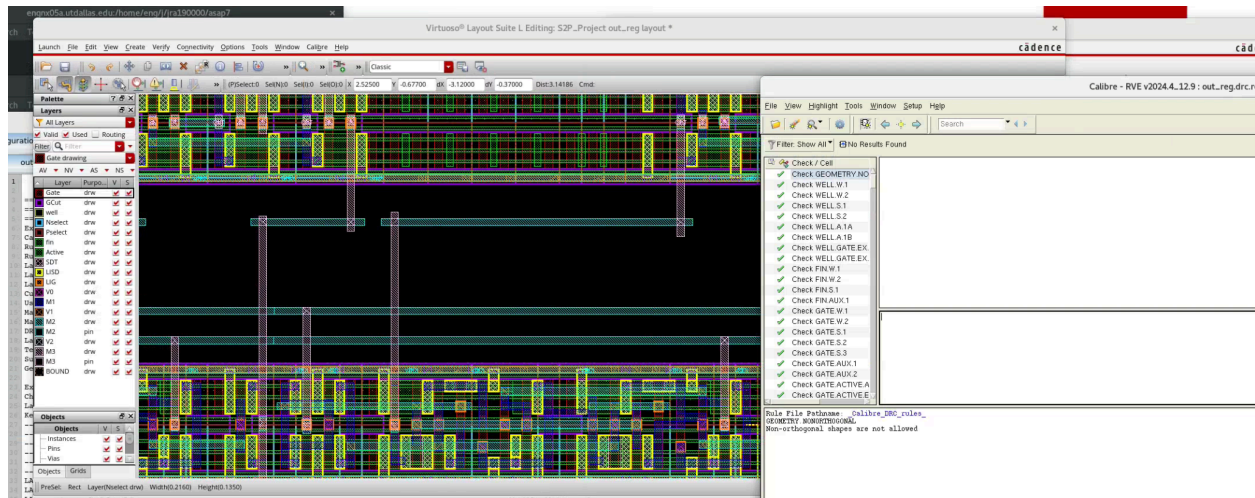
This is what the schematic of the gate netlists look like that i can do lvs test with layout for show integrity of layouts



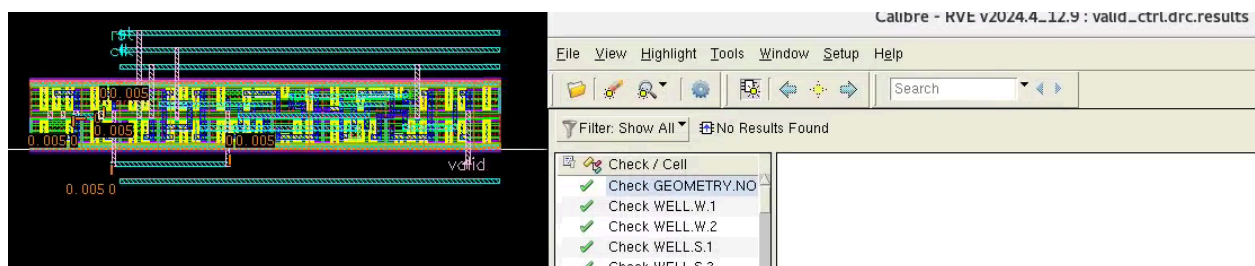
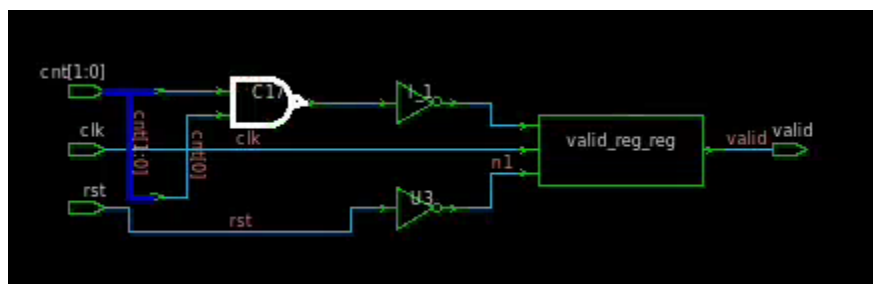
Then I made the DFF chain module which was quite easy because once I started routing two dffs I started to notice patterns and copied it. Then I rename the ports. Now I have the basis and setup to scale this to 16+bits. After passing LVS for this module I started doing the others. Once I knew I could do 4bits I knew I could do the others.

The next module was the out_Reg, which was essentially 4 registers with 3 nand gates for each bit/register





Boom then drc and lvs clean which is nice bc now i got two sections placed in a way that can scale well. Now I just will build the counter, and valid logic gates, which doesn't necessarily scale the same way as the register do in that linear fashion. The counters scale in a logarithmic fashion as the binary digits is need for the 2^n relationship.



```

LE NAME:      valid_ctrl.lvs.report
ME:          valid_ctrl.sp ('valid_ctrl')
ME:          valid_ctrl.src.net ('valid_ctrl')
:            _Calibre_LVS_rules
TITLE:       LVS Rule File for ASAP7 PDK
TIME:        Thu May 7 15:46:03 2026
IRECTORY:    /home/eng/j/jral90000/asap7/a_lvs_check
:            jral90000
ERSION:      v2024.4_12.9      Tue Oct 1 20:34:12 PDT

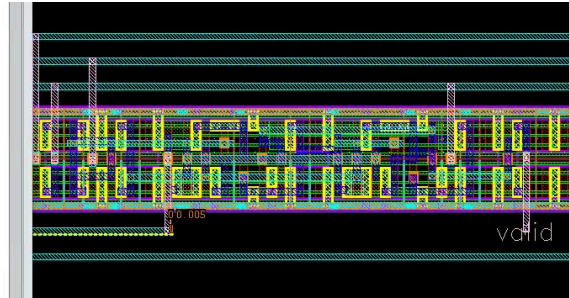
```

OVERALL COMPARISON RESULTS

```

#####
#          #
#          # CORRECT #
#          #
#####

```



Cell valid_ctrl Summary (Clean)

CELL COMPARISON RESULTS (TOP LEVEL)

```

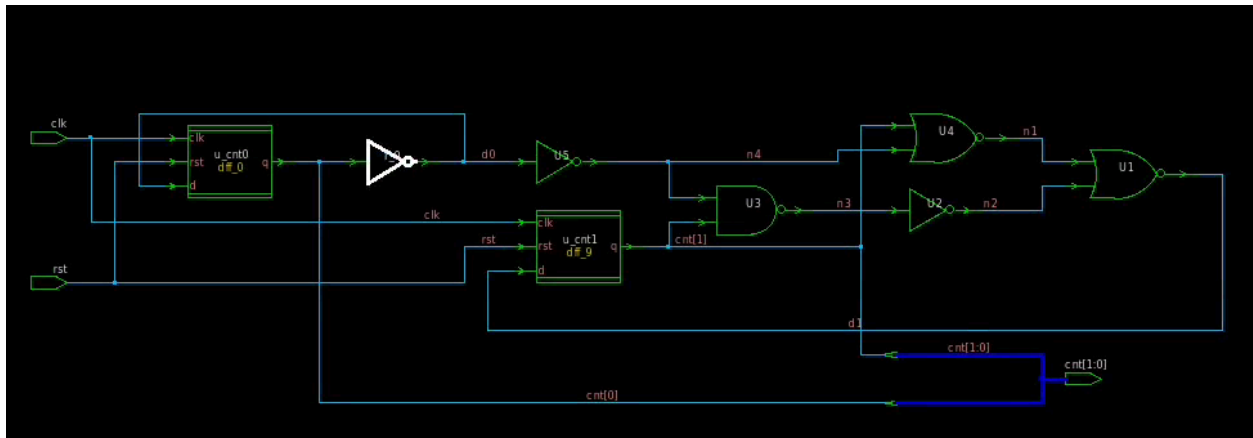
#####
#          #
#          # CORRECT #
#          #
#####

```

LAYOUT CELL NAME: valid_ctrl
SOURCE CELL NAME: valid_ctrl

INITIAL NUMBERS OF OBJECTS

Valid control logic (very small for 4 bits)



Module schematic for the counter

```

_Calibre_DRC_rules_

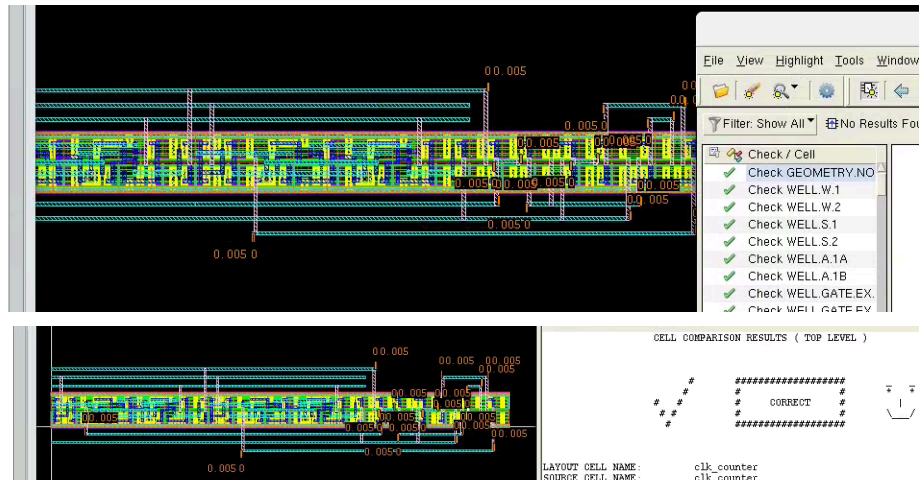
GDS
clk_counter.calibre.db
clk_counter
/home/eng/j/jral90000/asap7/drc_check
jral90000
: 1000
4096
clk_counter.drc.results (ASCII)
ALL
PRIMARY
clk_counter.drc.summary (REPLACE)
ACUTE = NO SKEW = NO ANGLED = NO

NONSIMPLE POLYGON = NO NONSIMPLE PATH =

COMMENT TEXT & DATE FILE INFORMATION

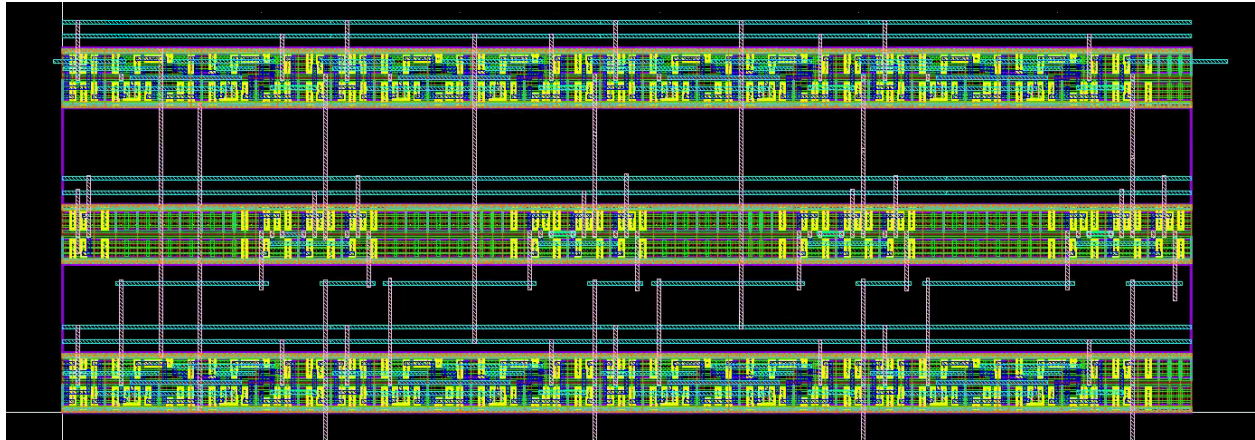
:      clk_counter.lvs.report
:      clk_counter.sp ('clk_counter')
:      clk_counter.src.net ('clk_counter')
:      _Calibre_LVS_rules
:      LVS Rule File for ASAP7 PDK
:      Thu May 7 17:18:01 2026
:      /home/eng/j/jral90000/asap7/a_lvs_check
:      jral90000
:      v2024.4_12.9      Tue Oct 1 20:34:12 PDT

```

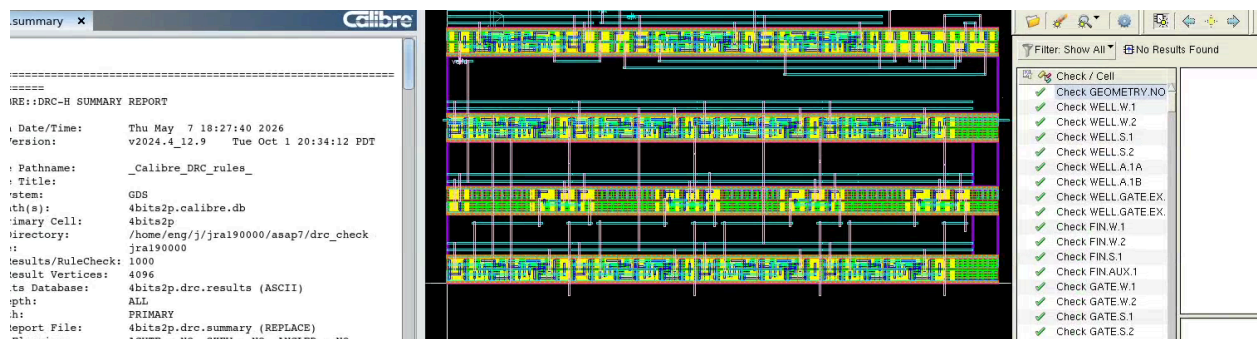


Drc and lvs clean for counter

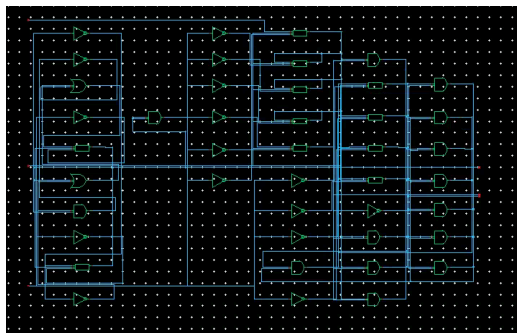
Now that I have each module for the project drc and lvs clean, I can easily combine them to result into a 4 bit s2p for 43 cells in total, but I want a better grade to we'll implement more cells



As we start to implement the different submodule together, you can see the intent of the designs as they mesh together well. Also reproducibility with the dff chain and output register works well together and will be the main area of scalability.



Drc pass 4bit with all module connected

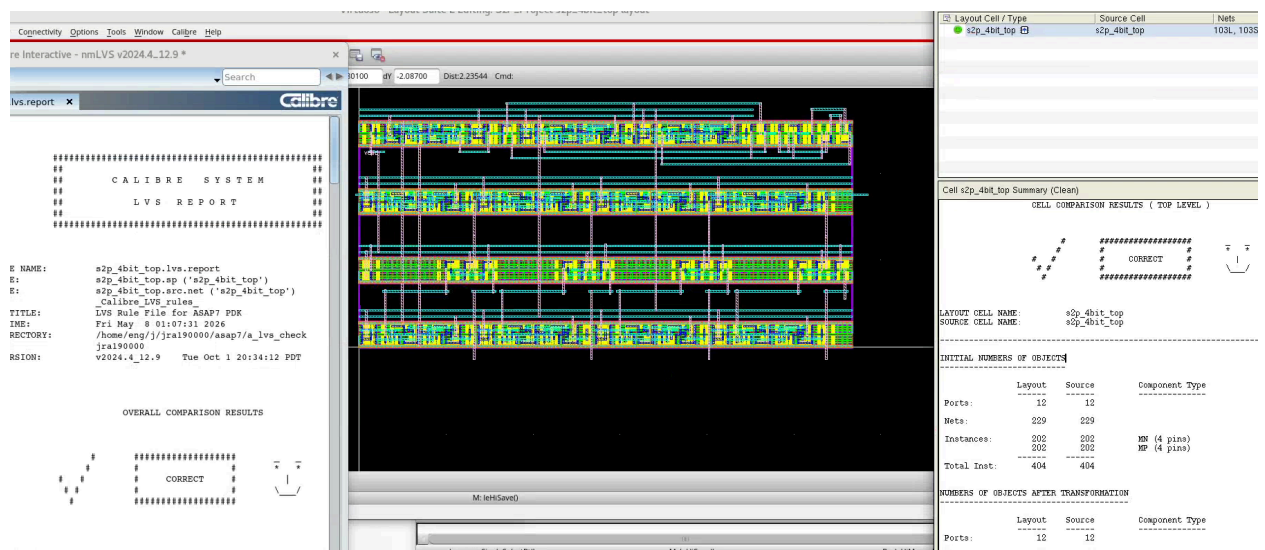


top schematic for LVS testing

***when doing top lvs i realized i had wrong [library.db](#) for synthesis and corrected it to [CELLS.db](#)

Yeah I actually ran into a lot of problems because the synthesis was using a pos edge DFF which change my logic for some of the modules because my dff is neg_edge... This sucked a lot...

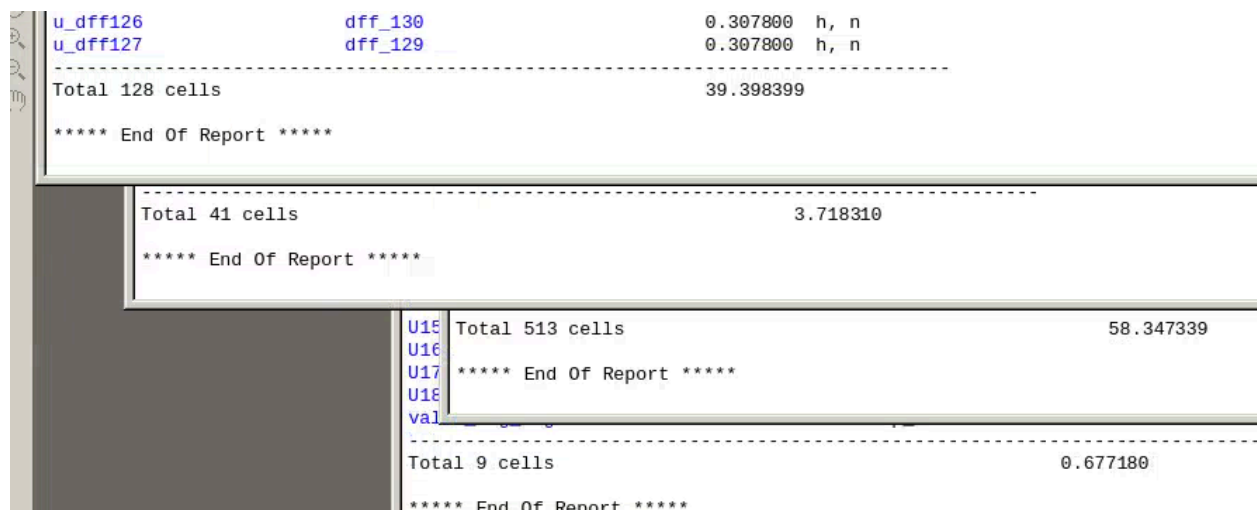
But i got it fixed an i passed DRC & LVS



The thing is that is only 38cells

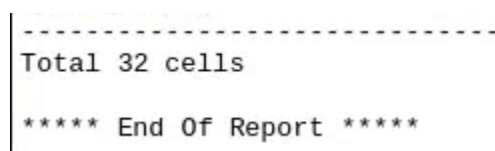
And frank i want points please so i'll make more a more complex design

In order to sustain scalability, I have to prevent fanout optimizations so I expect parasitics and timing to be bad.



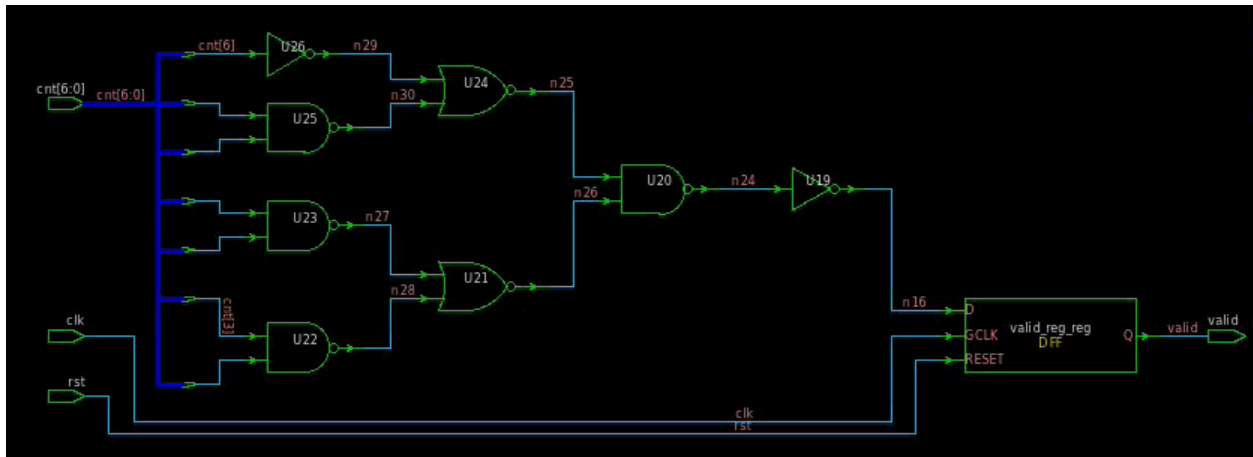
Total cells and area would be 101.59

And about **682** cells depending on routing optimization efforts



bottle neck for scaling

actually i optimize timer a little more because that is my



New version of valid logic

```

valid_ctrl.lvs.report
valid_ctrl.sp ('valid_ctrl')
valid_ctrl.src.net ('valid_ctrl')
_Calibre_LVS_rules_
LVS Rule File for Asap7 PDK
Fri May  8 06:19:28 2026
/home/eng/j3/jral90000/asap7/a_lvs_check
jral90000
V2024_4_12.9   Tue Oct 1 20:34:12 PDT
        
```

OVERALL COMPARISON RESULTS

```

# # #
# # # CORRECT # # #
# # #
        
```

Cell valid_ctrl Summary (Clean)

CELL COMPARISON RESULTS (TOP LEVEL)

```

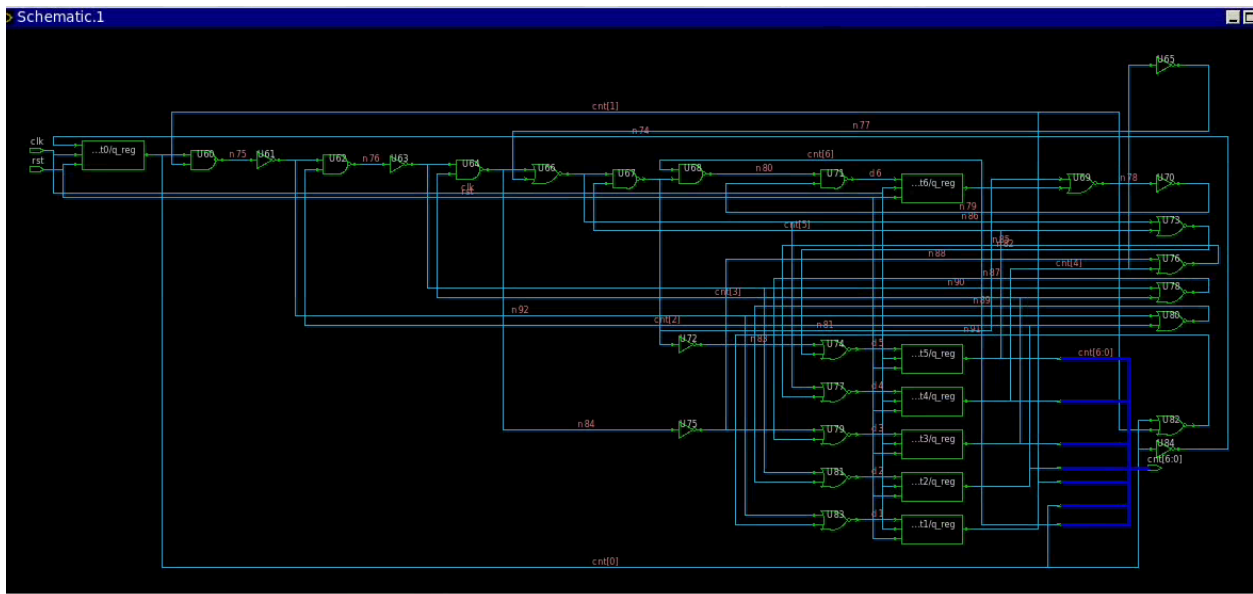
# # # CORRECT # # #
# # #
        
```

LAYOUT CELL NAME: valid_ctrl
SOURCE CELL NAME: valid_ctrl

INITIAL NUMBERS OF OBJECTS

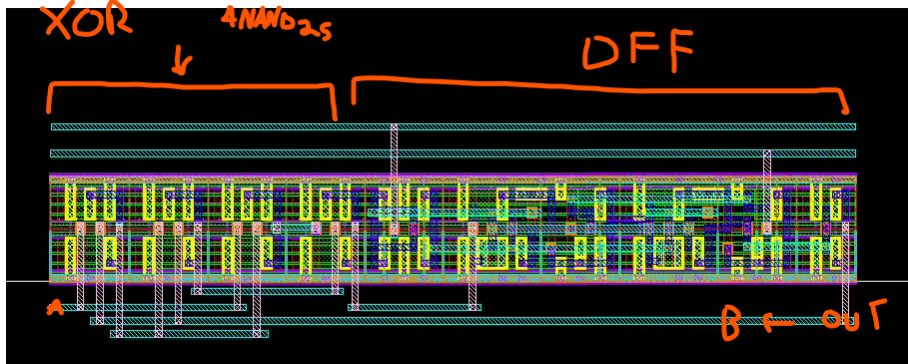
	Layout	Source	Component Type
Ports:	12	12	
Nets:	42	42	
Instances:	29	29	MW (4 ninal)

With lvs complete for



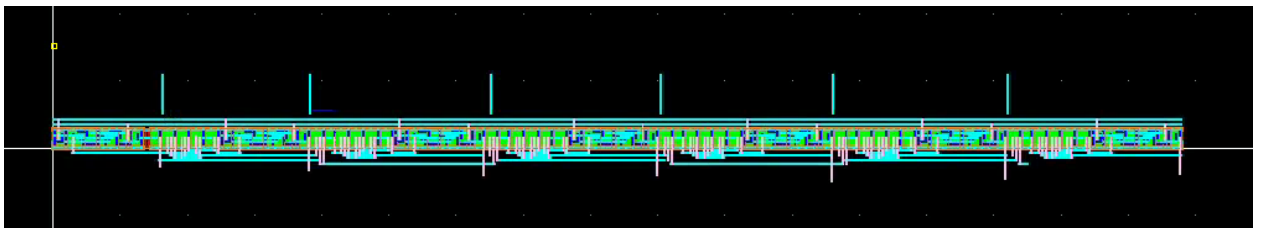
New version of counter gonna take a bit, but this is the most complex things then everything fall together. Man this counter itself could've been someone project.

- Alright making this is taking longer than i thought but essential i found a pattern that the binary counter utilizes and edge counter that can be used using xor, although i only have nand2 i can use 4 of them for xor logix, and the duplicate the pseudo xor, as each xor input will interact closely with each dff.

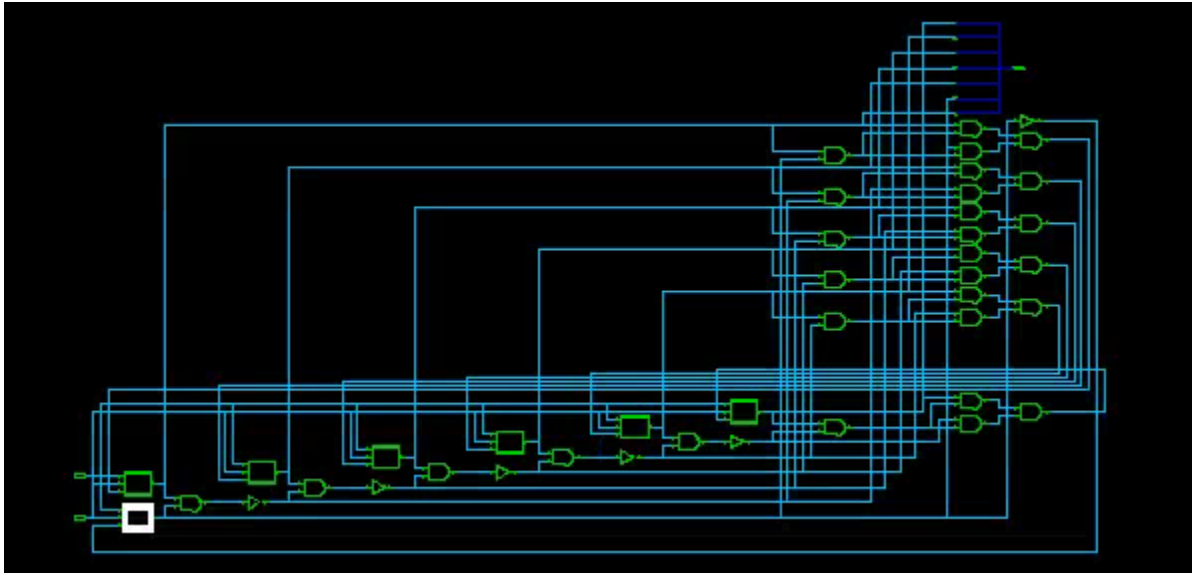


- And because we need 7 bits to count to 128 we weill have 7 xors and each dff/xor combo's output will be used into successive logic gates for ands as we increase in bitt significance.
- `assign d0 = ~cnt[0];`
- `assign d1 = cnt[1] ^ cnt[0];`
- `assign d2 = cnt[2] ^ (cnt[1] & cnt[0]);`
- `assign d3 = cnt[3] ^ (cnt[2] & cnt[1] & cnt[0]);`
- `assign d4 = cnt[4] ^ (cnt[3] & cnt[2] & cnt[1] & cnt[0]);`
- `assign d5 = cnt[5] ^ (cnt[4] & cnt[3] & cnt[2] & cnt[1] & cnt[0]);`
- `assign d6 = cnt[6] ^ (cnt[5] & cnt[4] & cnt[3] & cnt[2] & cnt[1] & cnt[0]);`

Just by using a nand2 and a inv for an and gate for the first i realized I can carry that and output further into another and2(nand2+inv) so i added an and2 to the current setup so then I can just copy it and connect it to finish the counter quickly.



After insertion structural logic to make sure synthesis follows i got the desired schematic



Each dff has an and gate ripple with the pseudo 'xor' with nand gates feeding into each dff.

It took a while to pass LVS

Run Control
Search
Transcript
Files

Run LVS
Show RVE

19 CREATION TIME: Fri May 8 22:23:24 2026
20 CURRENT DIRECTORY: /home/eng/j/jra190000/
21 USER NAME: jra190000
22 CALIBRE VERSION: v2024.4_12.9 Tue Oct 1
23 20:34:12 PDT 2024
24

RESULTS

OVERALL COMPARISON

Line	Symbol	Cell	Library	Match
1				
2				
3				
4				
5				
6				
7				
8				
9				
10				
11				
12				
13				
14				
15				
16				
17				
18				
19				
20				
21				
22				
23				
24				
25				
26				
27				
28				
29				
30				
31				
32				
33				
34				

OVERALL COMPARISON

#####

#

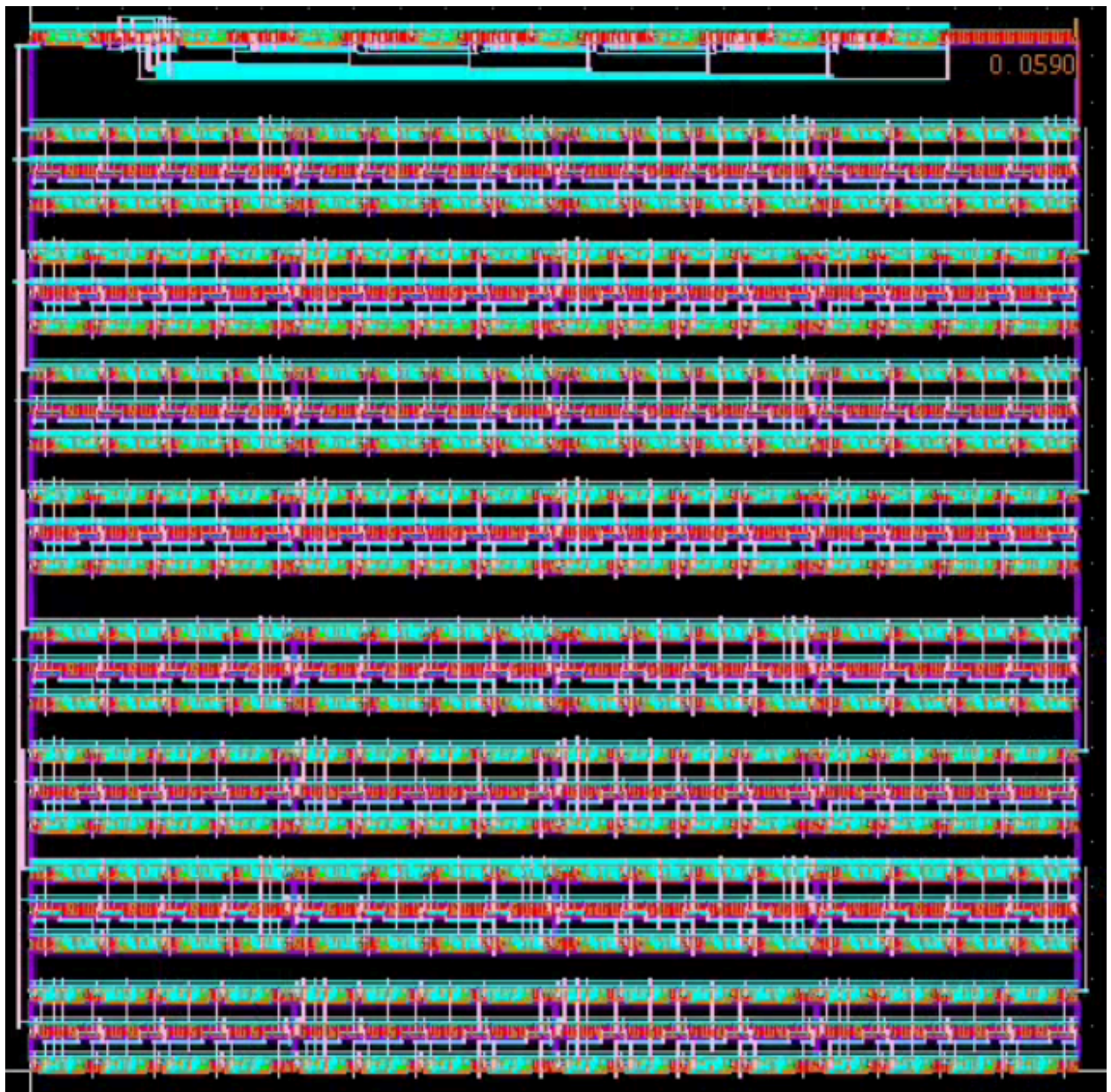
CORRECT

#

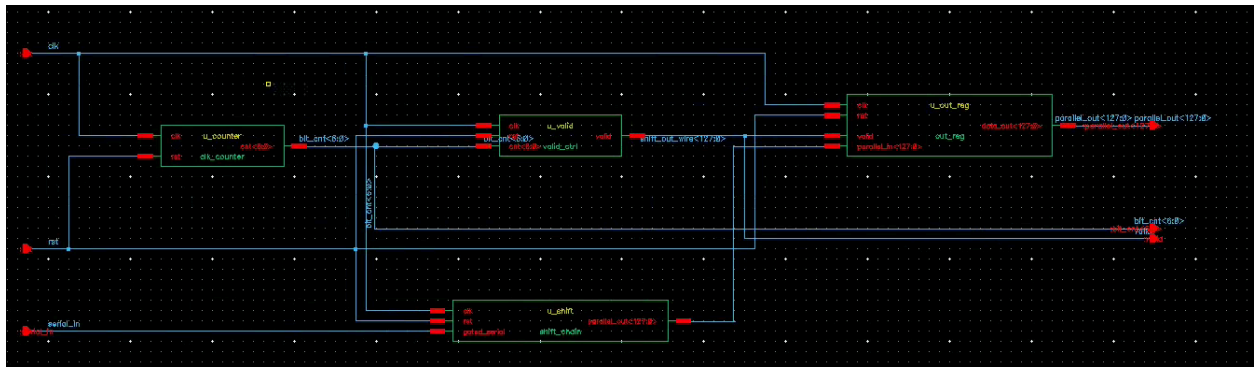
#####

Mainly because the 4 NAND2 pseudo xor setup was extremely case sensitive for lvs test.

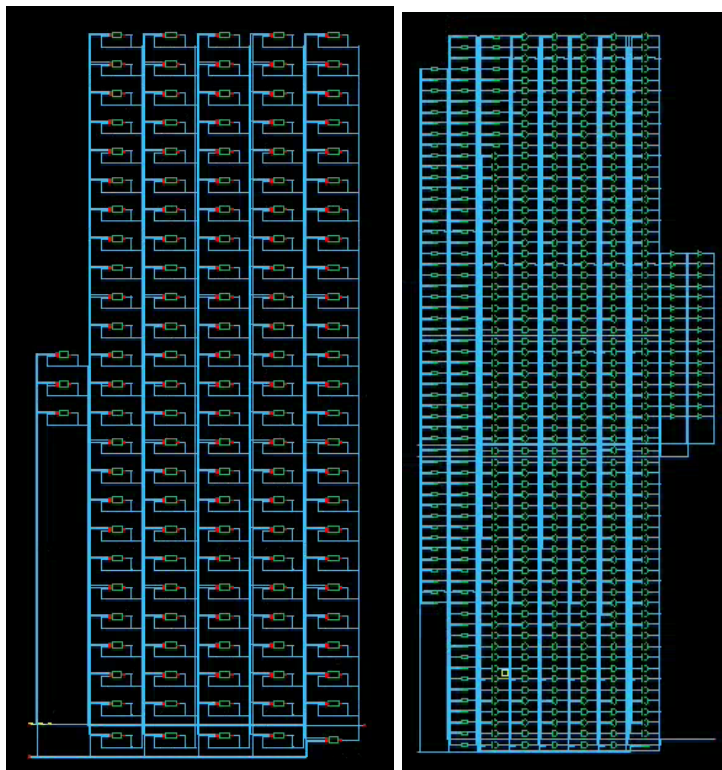
Then i implement it to the top



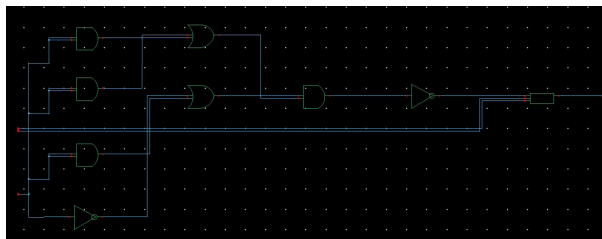
Schematic top is



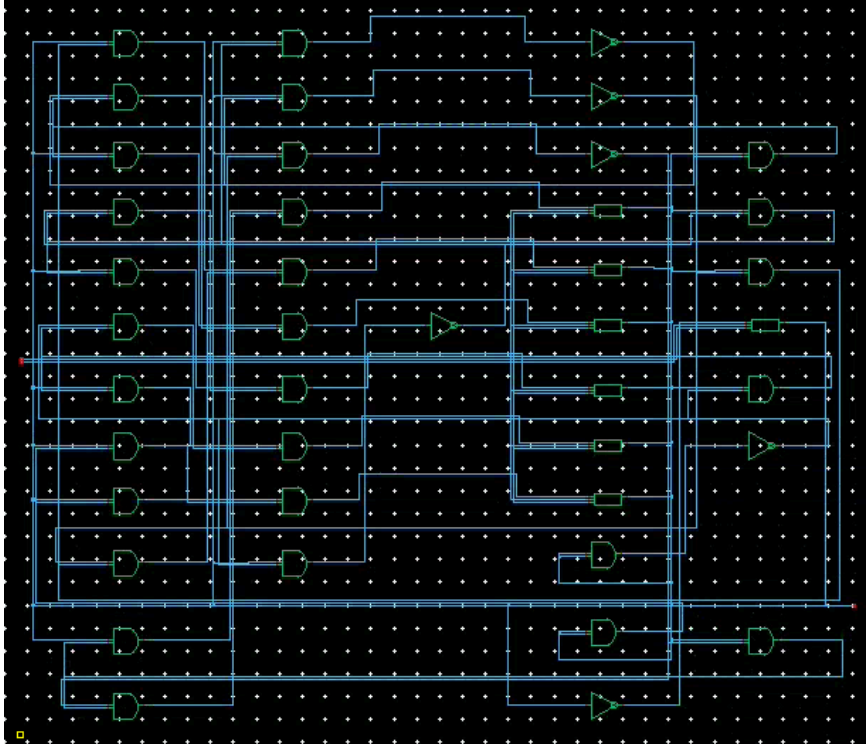
With readable schematics inside



Shift_chain and out_reg above



valid above



Counter above

DRC CLEAN

File View Highlight Tools Window Setup Help

Filter: Show All

No Results Found

Check / Cell

✓ Check GEOMETRY.NON

✓ Check WELL.W.1

✓ Check WELL.W.2

✓ Check WELL.S.1

✓ Check WELL.S.2

✓ Check WELL.A.1A

✓ Check WELL.A.1B

✓ Check WELL.GATE.EX

✓ Check WELL.GATE.EX

Calibre Run Completed Successfully -- Results are Valid

Run DRC

Show RVE

```
s2p_128bit_top.calibre.db
12 Layout Primary Cell: s2p_128bit_top
13 Current Directory: /home/eng/j/jral90000.
14 asap7/drc_check
15 User Name: jral90000
16 Maximum Results/RuleCheck: 1000
17 Maximum Result Vertices: 4096
18 DRC Results Database:
19 s2p_128bit_top.drc.results (ASCII)
20 Layout Depth: ALL
21 Text Depth: PRIMARY
22 Summary Report File:
23 s2p_128bit_top.drc.summary (REPLACE)
24 Geometry Flagging: ACUTE = NO SKEW = NO
25 ANGLED = NO OFFGRID = NO NONSIMPLE POLYGON = NO
26 NONSIMPLE PATH = NO
27 Excluded Cells:
28 CheckText Mapping: COMMENT TEXT + RULE
29 FILE INFORMATION
30 Layers: MEMORY-BASED
31 Keep Empty Checks: YES
```

CELL REPORTS

Report.3 - Cell

u_out115	DFF	op_cond	0.388000	n
u_out116	DFF	op_cond	0.388000	n
u_out117	DFF	op_cond	0.388000	n
u_out118	DFF	op_cond	0.388000	n
u_out119	DFF	op_cond	0.388000	n
u_out120	DFF	op_cond	0.388000	n
u_out121	DFF	op_cond	0.388000	n
u_out122	DFF	op_cond	0.388000	n
u_out123	DFF	op_cond	0.388000	n
u_out124	DFF	op_cond	0.388000	n
u_out125	DFF	op_cond	0.388000	n
u_out126	DFF	op_cond	0.388000	n
u_out127	DFF	op_cond	0.388000	n

Total 544 cells

75.043201

1

design_vision>

Log

History

design_vision>

Design Vision - TopLevel.1 (clk_counter)

File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Wire

Report.3 - Cell

g_x4_1	NAND2	op_cond	0.062200	
g_x4_2	NAND2	op_cond	0.062200	
g_x4_3	NAND2	op_cond	0.062200	
g_x4_4	NAND2	op_cond	0.062200	
g_x4_5	NAND2	op_cond	0.062200	
g_x4_6	NAND2	op_cond	0.062200	
u_dff0	DFF	op_cond	0.388000	n
u_dff1	DFF	op_cond	0.388000	n
u_dff2	DFF	op_cond	0.388000	n
u_dff3	DFF	op_cond	0.388000	n
u_dff4	DFF	op_cond	0.388000	n
u_dff5	DFF	op_cond	0.388000	n
u_dff6	DFF	op_cond	0.388000	n
Total 42 cells			4.800000	
1				
design_vision>				

Design Vision - TopLevel.1 (shift_chain)

File Edit View Select Highlight List Hierarchy Design Attributes Schematic Timing Test Power AnalyzeRTL Wire

shift_chain

u_dff116/q_reg	DFF	op_cond	0.388000	n
u_dff117/q_reg	DFF	op_cond	0.388000	n
u_dff118/q_reg	DFF	op_cond	0.388000	n
u_dff119/q_reg	DFF	op_cond	0.388000	n
u_dff120/q_reg	DFF	op_cond	0.388000	n
u_dff121/q_reg	DFF	op_cond	0.388000	n
u_dff122/q_reg	DFF	op_cond	0.388000	n
u_dff123/q_reg	DFF	op_cond	0.388000	n
u_dff124/q_reg	DFF	op_cond	0.388000	n
u_dff125/q_reg	DFF	op_cond	0.388000	n
u_dff126/q_reg	DFF	op_cond	0.388000	n
u_dff127/q_reg	DFF	op_cond	0.388000	n
Total 128 cells			49.664001	
1				

Design Vision - TopLevel.1 (valid_ctrl)

Report.3 - Cell

u - contains unmapped logic

Cell	Reference	Library	Area	Attr
U11	inverter	op_cond	0.046700	
U12	NAND2	op_cond	0.062200	
U13	NOR2	op_cond	0.062200	
U14	NAND2	op_cond	0.062200	
U15	NAND2	op_cond	0.062200	
U16	NOR2	op_cond	0.062200	
U17	NAND2	op_cond	0.062200	
U18	inverter	op_cond	0.046700	
valid_reg_reg	DFF	op_cond	0.388000	n
Total 9 cells			0.854600	
1				
design_vision>				

u - contains unmapped logic

Cell	Reference	Library	Area	Attributes
u_counter	clk_counter		4.800000	h, n
u_out_reg	out_reg		75.043201	h, n
u_shift	shift_chain		49.664001	h, n
u_valid	valid_ctrl		0.854600	h, n
Total 4 cells			130.361802	
1				
design_vision>				

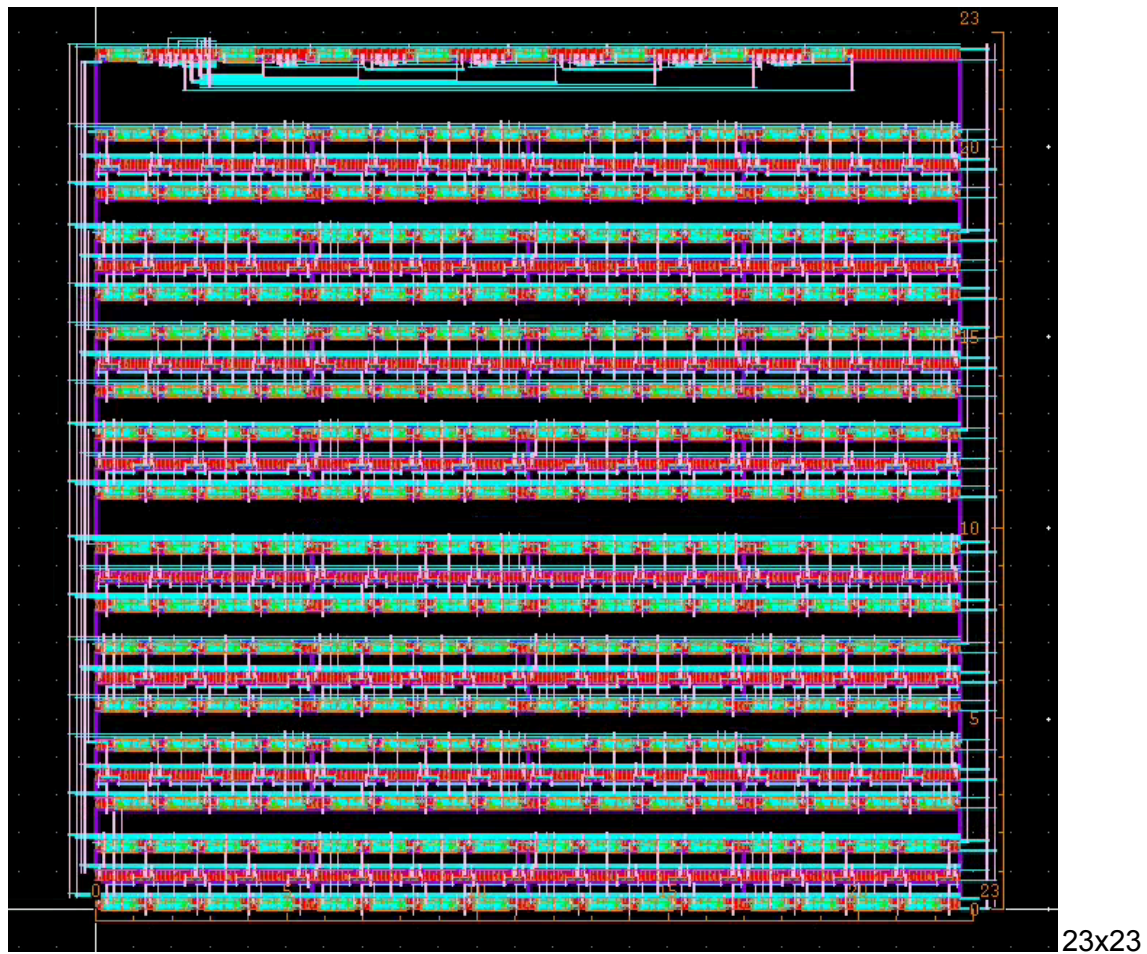
$$9 + 128 + 4 + 544 =$$

685

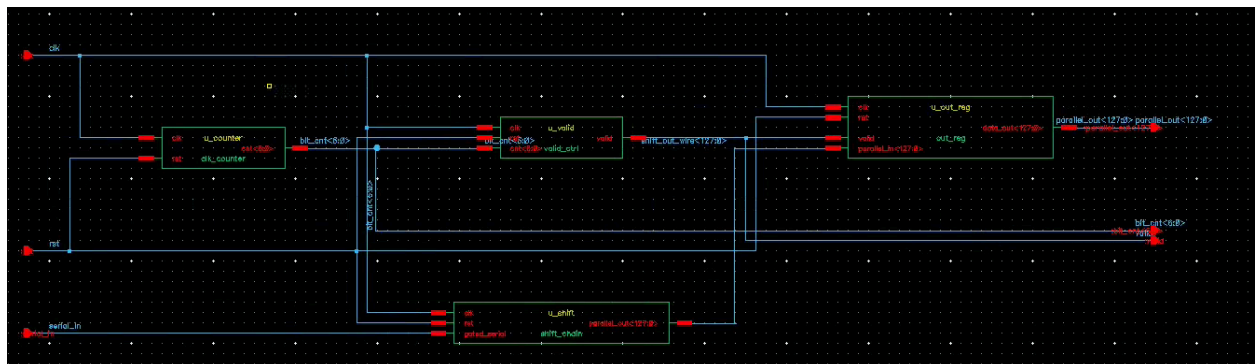
total cells

LVS CHECK

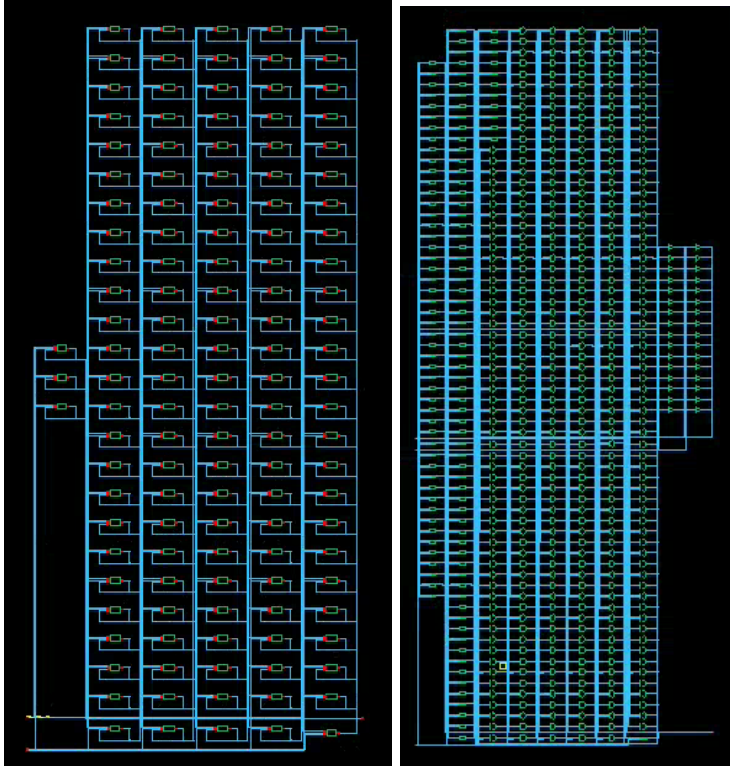
Fit size image. With Manual PnR, the objective was to pass LVS, and not to optimize timing, power, or



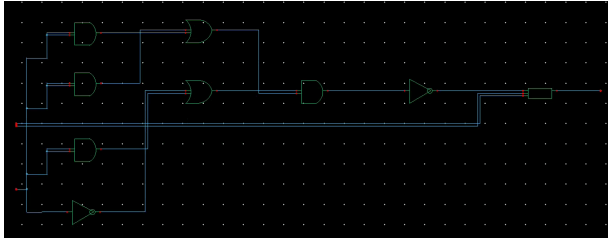
Schematic top is



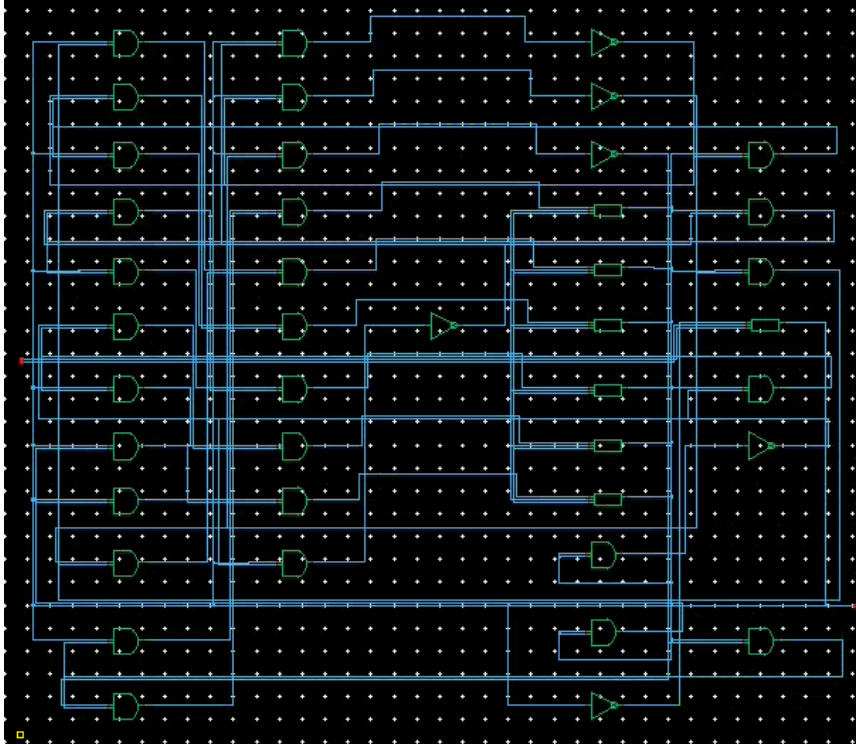
With readable schematics inside



Shift_chain and out_reg above



avlid above



Counter above

DRC CHECK

```
=====
SUMMARY REPORT

: Sat May 9 08:28:50
v2024.4_12.9 Tue
_Calibre_DRC_rules_
GDS

bre.db
: s2p_128bit_top
/home/eng/j/jra190000/
jra190000
eCheck: 1000
ices: 4096
e:
results (ASCII)
ALL
PRIMARY
.
```

LVS CHECK

```

s2p_128bit_top.lvs.report
s2p_128bit_top.sp ('s2p_128bit_top')
s2p_128bit_top.src.net ('s2p_128bit_top')
_Calibre_LVS_rules_
LVS Rule File for ASAP7 PDK
Sat May  9 08:29:51 2026
/home/eng/j/jral90000/asap7/a_lvs_check
jral90000
v2024.4_12.9   Tue Oct 1 20:34:12 PDT 2024

```

OVERALL COMPARISON RESULTS

```

#####
#          #          #          #          #          #          #          #
# CORRECT  #          #          #          #          #          #          #
#####

```

Cell s2p_128bit_top Summary (Clean)

CELL COMPARISON RESULTS (TOP LEVEL)

```

#####
#          #          #          #          #          #          #          #
# CORRECT  #          #          #          #          #          #          #
#####

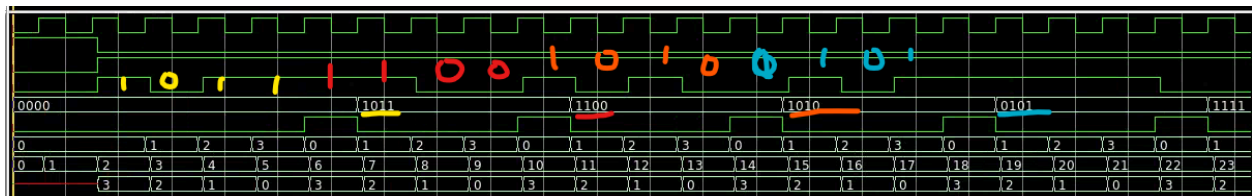
```

LAYOUT CELL NAME: s2p_128bit_top
SOURCE CELL NAME: s2p_128bit_top

INITIAL NUMBERS OF OBJECTS

	Layout	Source	Component Type
Ports:	141	141	
Nets:	5340	5340	
Instances:	4838	4838	MN (4 pins)

SIMULATIONS



Hdl simulation reduced to 4 bits for easier observation of a p2s, but 128b structural verilog include in zip

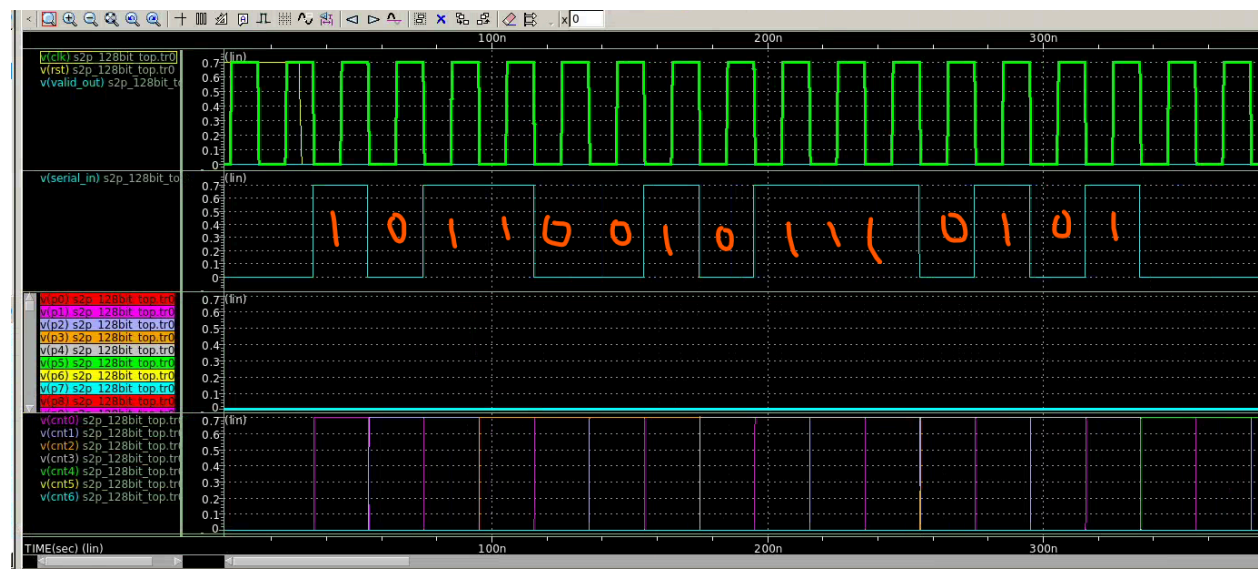
Waveform synopsys view

Showing serial 128b message with first 15b of useful info for demo

Expected 101100101110101

+ 0ns	0v			
+ 35ns	0.7v	34.99ns	0v	\$ bit 1: 1
+ 55ns	0v	54.99ns	0.7v	\$ bit 2: 0
+ 75ns	0.7v	74.99ns	0v	\$ bit 3: 1
+ 95ns	0.7v			\$ bit 4: 1
+ 115ns	0v	114.99ns	0.7v	\$ bit 5: 0
+ 135ns	0v			\$ bit 6: 0
+ 155ns	0.7v	154.99ns	0v	\$ bit 7: 1
+ 175ns	0v	174.99ns	0.7v	\$ bit 8: 0
+ 195ns	0.7v	194.99ns	0v	\$ bit 9: 1
+ 215ns	0.7v			\$ bit 10: 1
+ 235ns	0.7v			\$ bit 11: 1
+ 255ns	0v	254.99ns	0.7v	\$ bit 12: 0
+ 275ns	0.7v	274.99ns	0v	\$ bit 13: 1
+ 295ns	0v	294.99ns	0.7v	\$ bit 14: 0
+ 315ns	0.7v	314.99ns	0v	\$ bit 15: 1
+ 335ns	0v	334.99ns	0.7v	
+)				

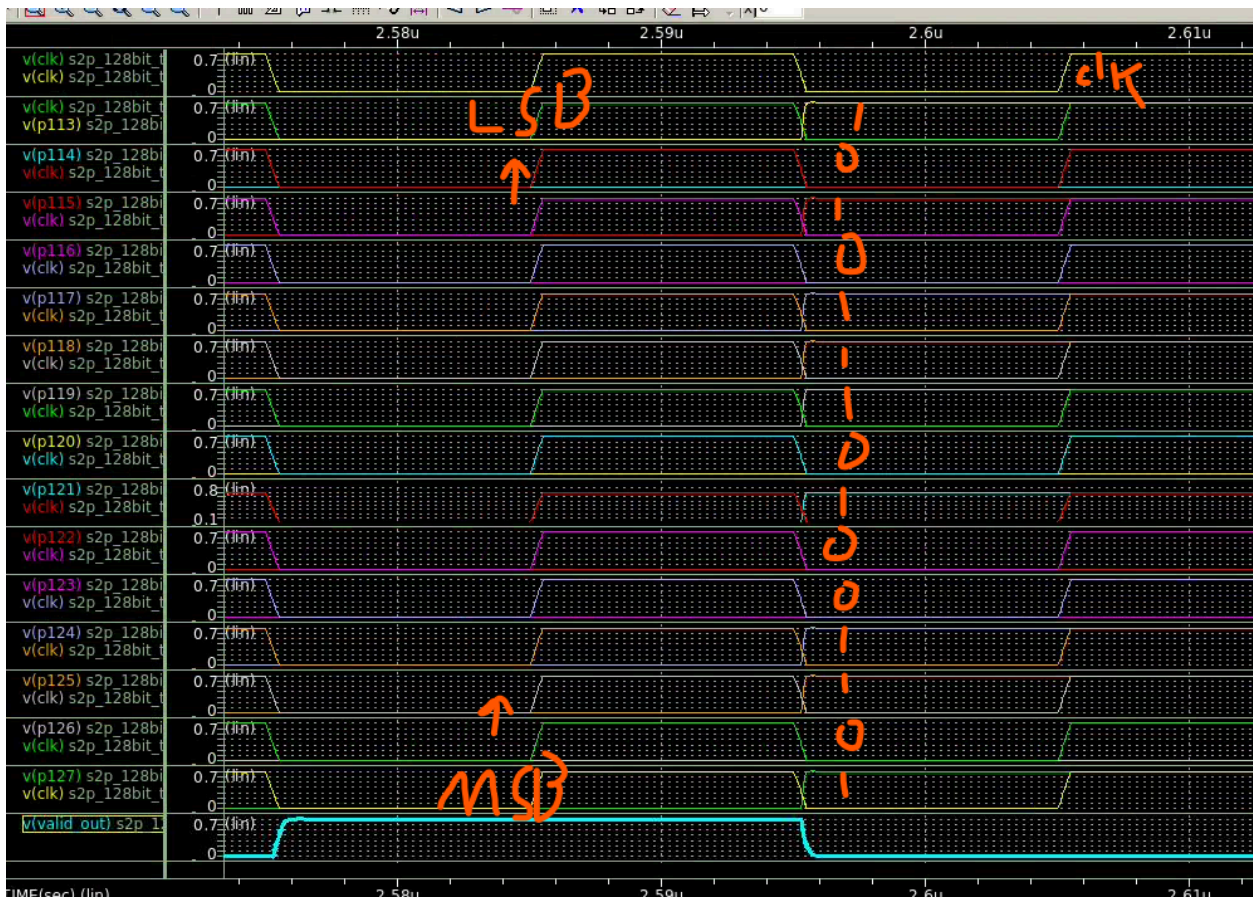
We can expect our serial data to be transferred



Now we can look at valid out w/ cnt interaction and its operation with parallel out (p1-127)



Now to take a closer look into those 15 msb of our 128b word
 And we can see the parallel out register displaying the right out!!
 Success!!



And then quick reset sanity check

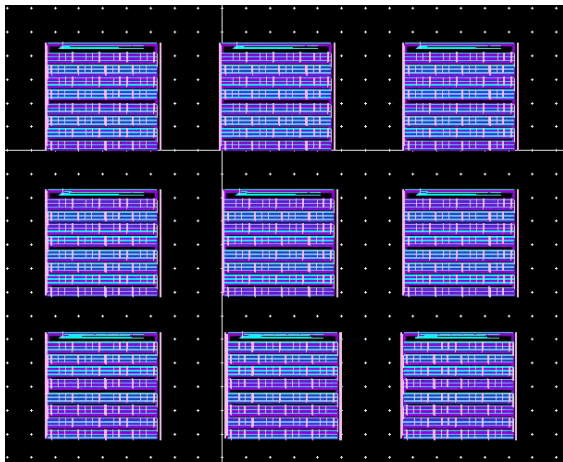


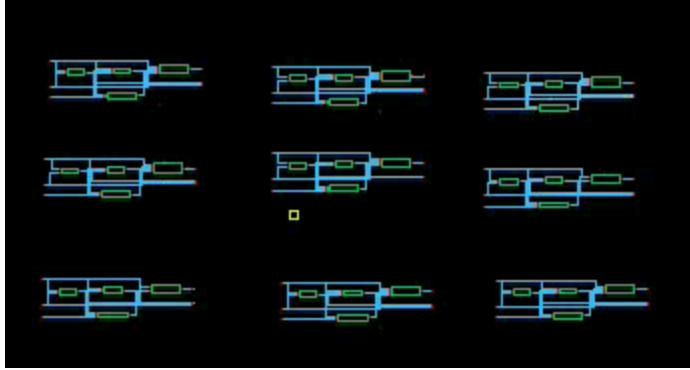
HDL code included in zip file along with layout, schematics, hspice script

BONUS: 9 channel 128b S2P for 6165cells

But I don't have enough space to simulate it in no machine

And manually renaming the parallelout<0-127> port would not be cool.





hierarchical schematic of a multi channel that could've been done (this project was graded competitively based on number of cells w/ manual PnR)