Abstract

Carry lookahead adder is an adder used to perform fast addition. We have learned about it in CPEN 311 and most of the information about it can be found in the book [1]. In contrast to the ripple adder which transfers the carry bits sequentially, the carry lookahead adder generates the carry bits in parallel for all of the full adder bits.

The circuit itself expands the outputs of the full adder to have two additional inputs -

1. Propagate: P = A XOR B
2. Generate: G = A AND B

Where A and B are single bit inputs into the full adder. In addition to the adders we create an additional combinational logic unit which takes the P and G from each full adder and calculates the carries of the Xbits adder. The calculation is done recursively for each carry bit which makes this circuit substantially larger than a normal ripple adder.

The carries are generated in the following way -

- $C1 = G0 + (P0 \cdot C0)$
- $C2 = G1 + (P1 \cdot C1) = G1 + (P1 \cdot G0 + (P1 \cdot P0 \cdot C0))$
- $Ci = Gi-1 + (Pi-1 \cdot Ci-1)$

Where i corresponds to the ith bit, starting at 0 index bit.

Results: The final block of the 8-Bit Full Carry Lookahead Adder has a maximum delay of about 0.205 nanoseconds. To see full results see the Simulation section.

Circuit Implementation

To implement the full carry lookahead circuit I built the building blocks for the circuit one by one. In this section I will present all the building blocks, their functionality, and how they were tested, later sections will present the testbench results and the design considerations for each block.

All functionalities were tested individually for every single block verifying its logical operation using stimuli. After each individual block was tested it was used to build a higher level block. All higher level blocks were also tested for logical functionality. For example - an inverter was built and tested and then used to build a xor gate that was also individually tested.

Summary of building blocks

1. Inverter gate - 1 input 1 output
2. And gate - 2 input 1 output
3. Or gate - 2 input 1 output
4. Xor gate - 2 input 1 output
5. Full adder - 3 inputs (a,b, carryin) 4 outputs (sum, generate, propagate, carryout)

6. 4 bit carry lookahead combinational logic - 9 inputs (4 generate, 4 propagate, 1 carryin) 4 outputs (carryout 1-4)
7. 4 bit carry lookahead adder - 9 inputs (a 1-4, b 1-4, carryin) 5 outputs (sum 1-4, carryout)

Inverter Gate

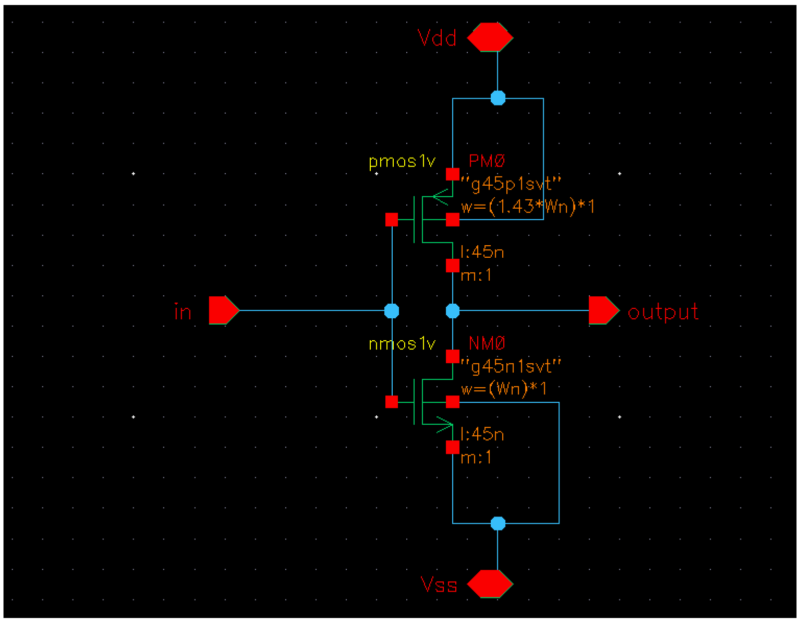


Figure 1 - Inverter Gate Schematics

And Gate

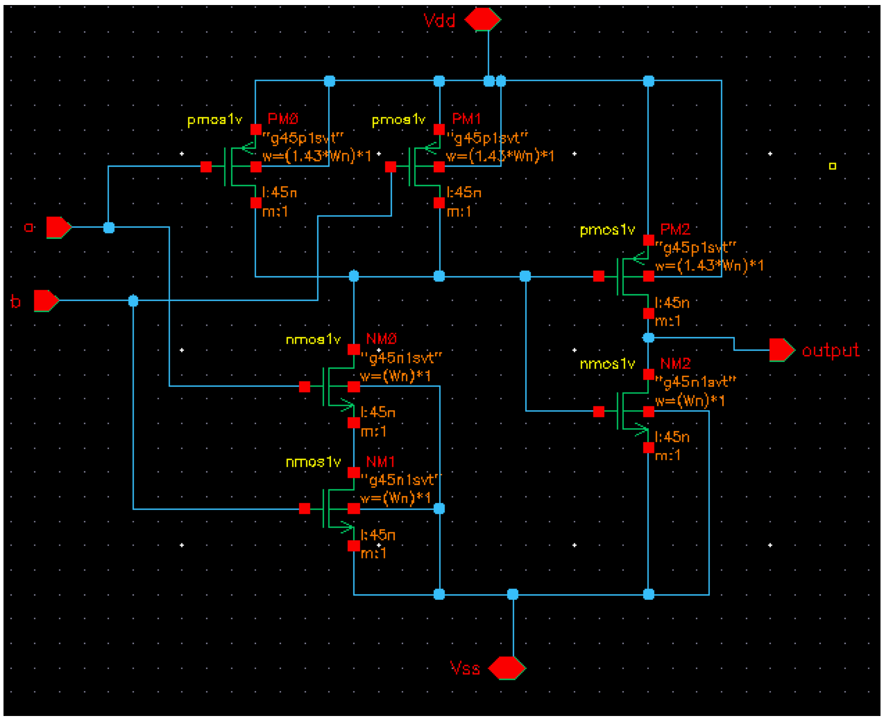
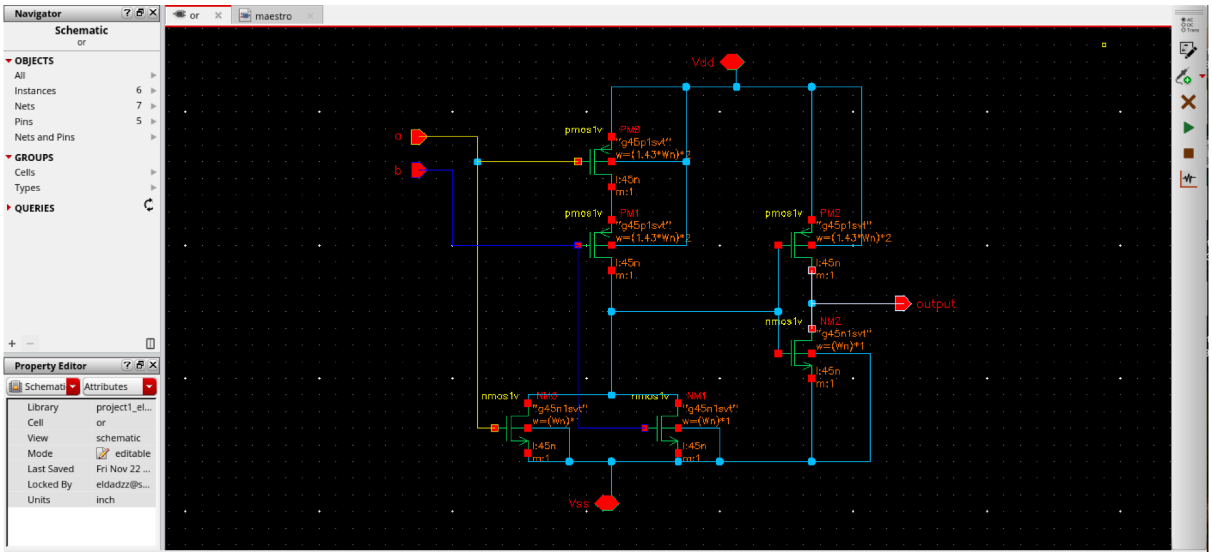


Figure 2 - And Gate Schematics

Or Gate



Library	project1_el...
Cell	or
View	schematic
Mode	editable
Last Saved	Fri Nov 22 ...
Locked By	eldadzz@s...
Units	inch

Figure 3 - Or Gate Schematics

Xor Gate

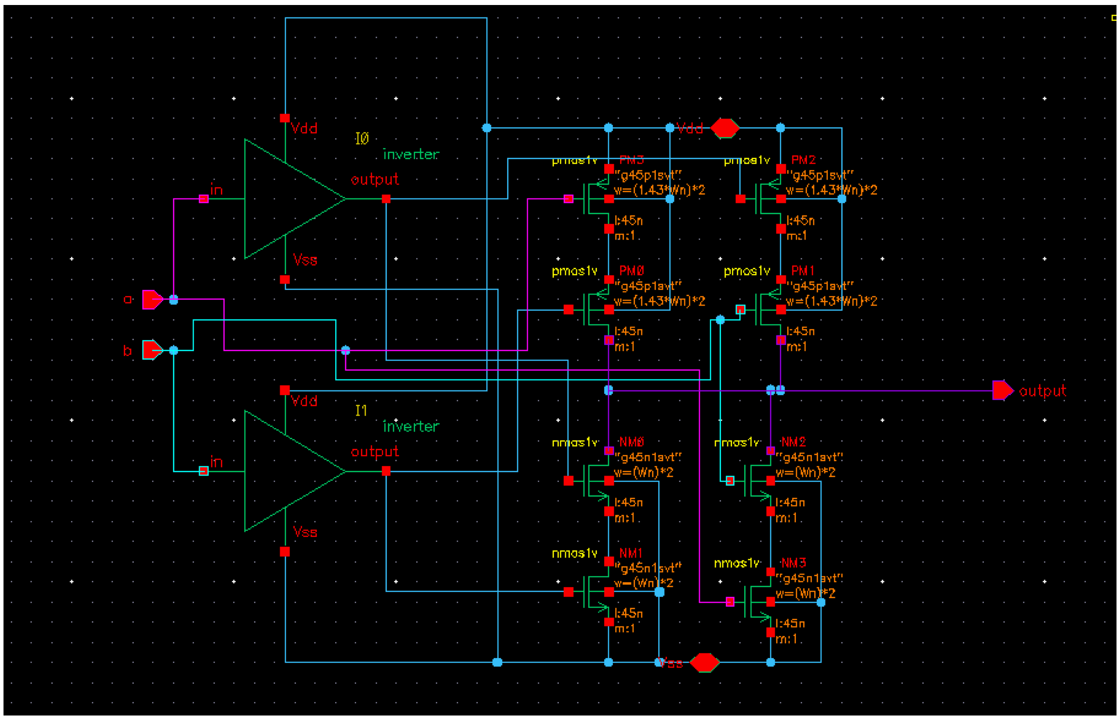


Figure 4 - Xor Gate Schematics

Full Adder with Carry and Propagate

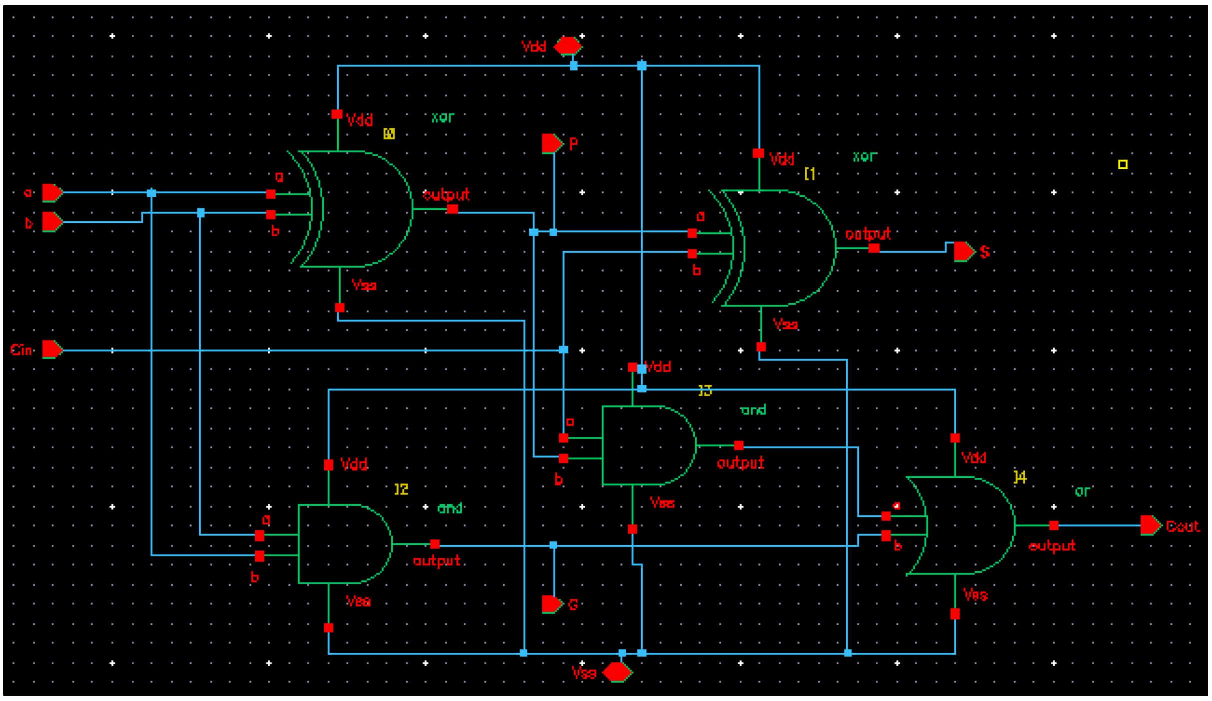


Figure 5 - Full Adder Gate Schematics

Four Bit Carry Lookahead Combinational Unit

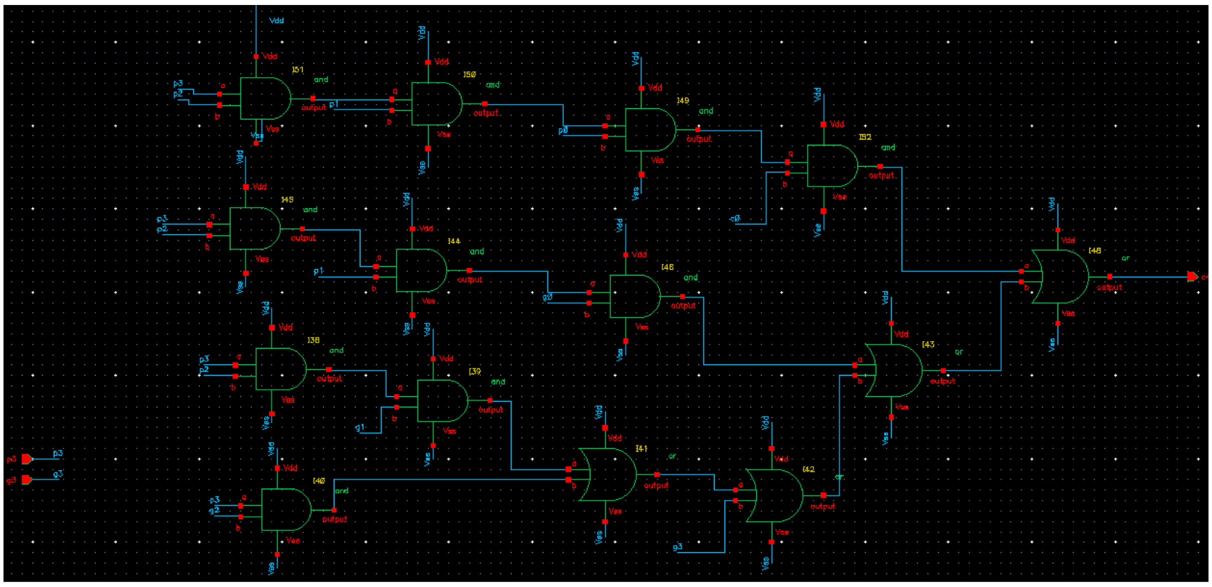


Figure 6 - Four Bit Carry Lookahead Combinational Unit Gate Schematics - Top

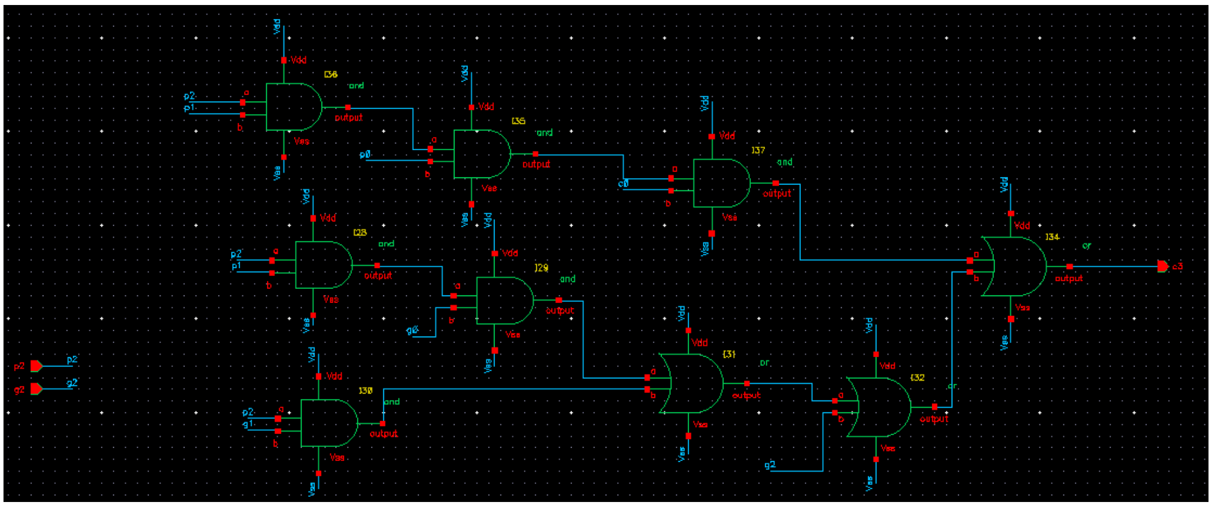


Figure 7 - Four Bit Carry Lookahead Combinational Unit Gate Schematics - Middle

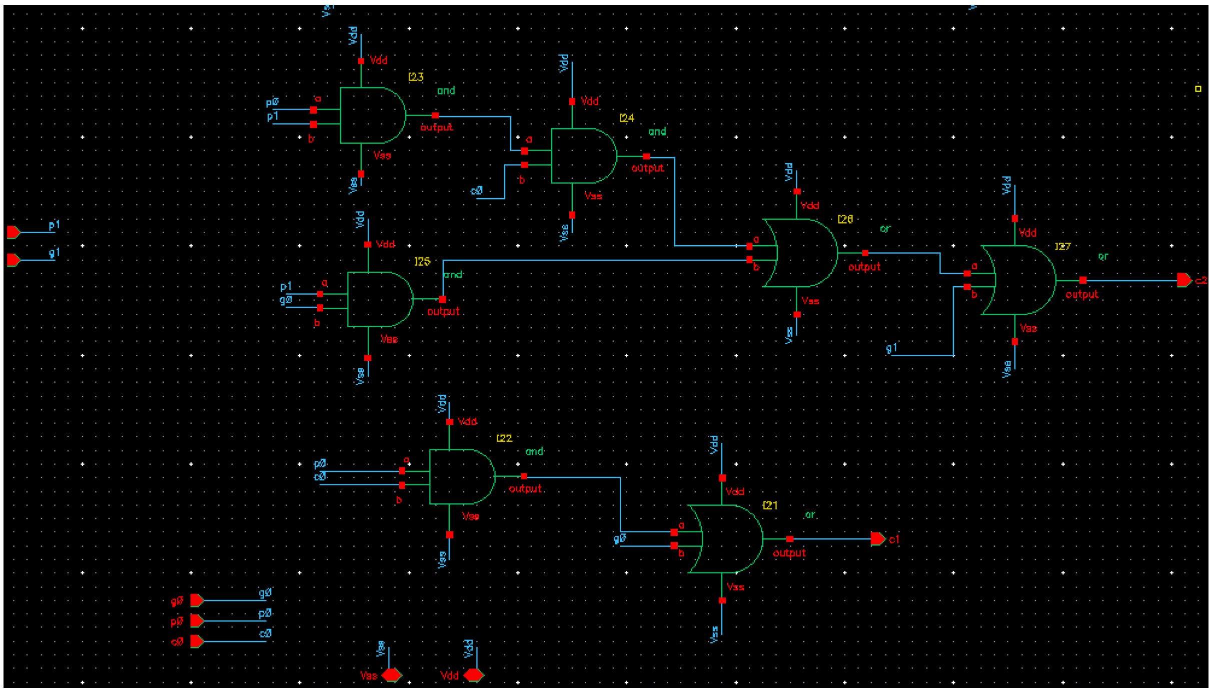


Figure 8 - Four Bit Carry Lookahead Combinational Unit Gate Schematics - Bottom

Four Bit Carry Lookahead Adder

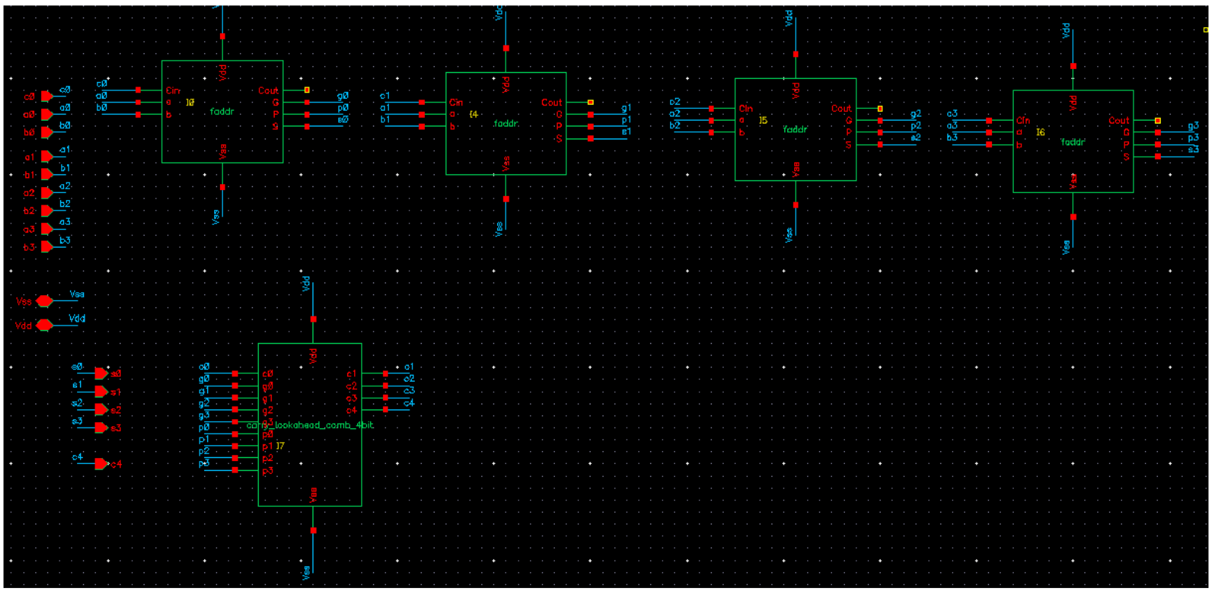


Figure 9 - Four Bit Carry Lookahead Adder Schematics

Eight Bit Carry Lookahead Adder

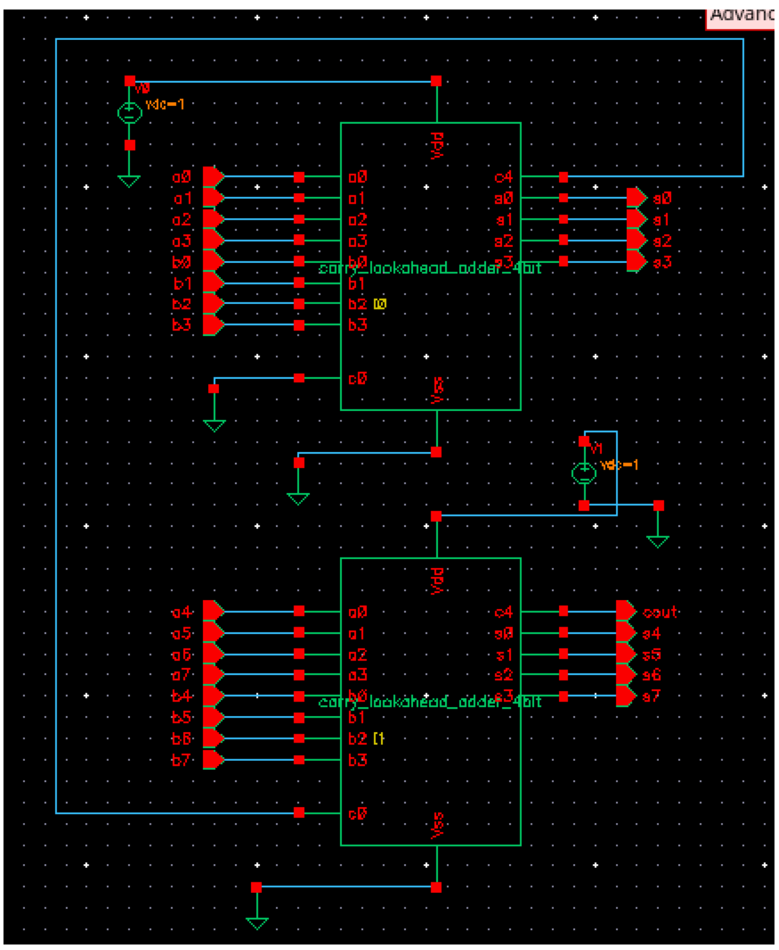


Figure 10 - Eight Bit Carry Lookahead Adder Schematics

Design Consideration and Techniques

Multiple Input Gate vs Cascaded Two Input Gate

In the combinational unit of the carry lookahead adder we need to decide how we would like to implement boolean functions of multiple bits. For example: (P1•P0•C0) for the calculation of C1. This applied to gates up to the largest gate we have which would have been a 5 input and.

The reason I decided to chain 2 input gates together instead of creating multiple input gates is because multiple input gates cause the pull down/ pull up network to have more capacitance/resistance (depending on the gate) which causes the rise time and fall time of the gate to increase. This consideration outweighs the fact that we need more MOS devices in order to implement our blocks as we are maximizing the speed of our adder.

As seen in some of the simulation graphs (in waveform), this design contains glitches. While glitches are to be expected in every design by the nature of the fact that we cannot have all lines be the exact same length, they can be minimized. For example, the design of the XOR gate contains two inverters that go only to some of the inputs, granting that some inputs are delayed, issues like these should be the first to be addressed for a higher quality of design.

Simulation Results

Simulation results are divided into two categories:

1. Verified using pattern matching of signal plotted in python, all annotated
    a. Inverter
    b. And
    c. Or
    d. Xor
    e. Full adder with propagate and generate
2. Verified using assert statements in python
    a. Carry lookahead combination unit
    b. 4-bit carry lookahead adder unit
    c. 8-bit carry lookahead adder

Verification Using Graphing

Methodology:
For each gate block I created a stimuli profile with as many pulses as there are inputs. I assigned a delay and a different pulse width to all of them in order to have any possible input combination in them.

Inverter Gate

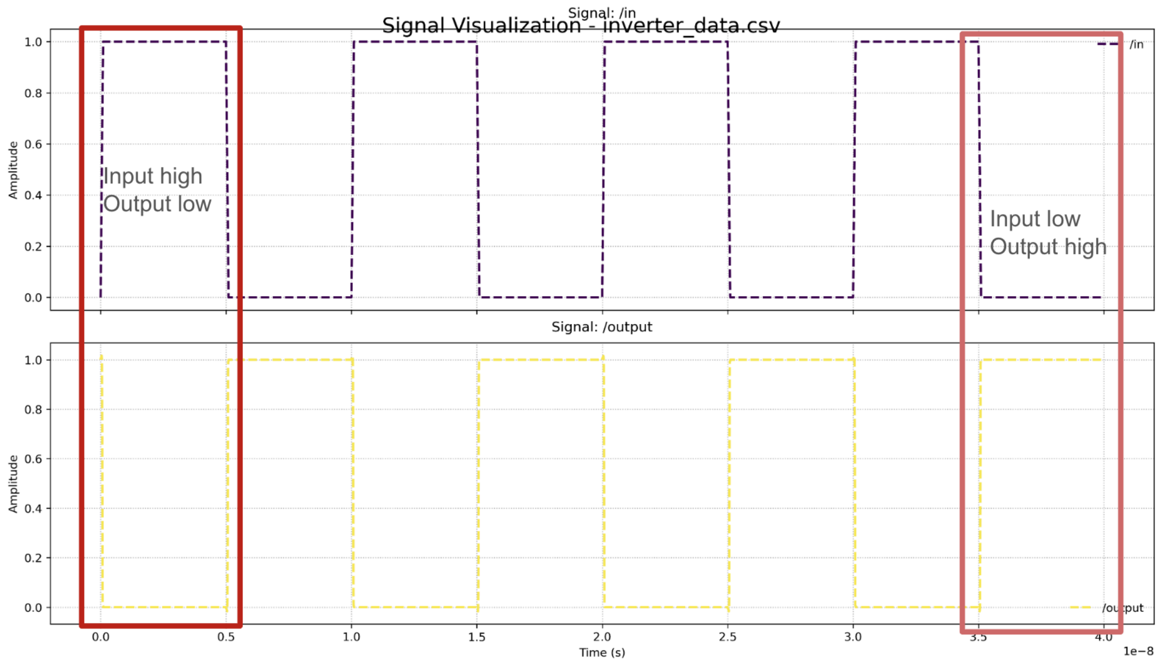


Figure 11 - Inverter Gate Verification

And Gate

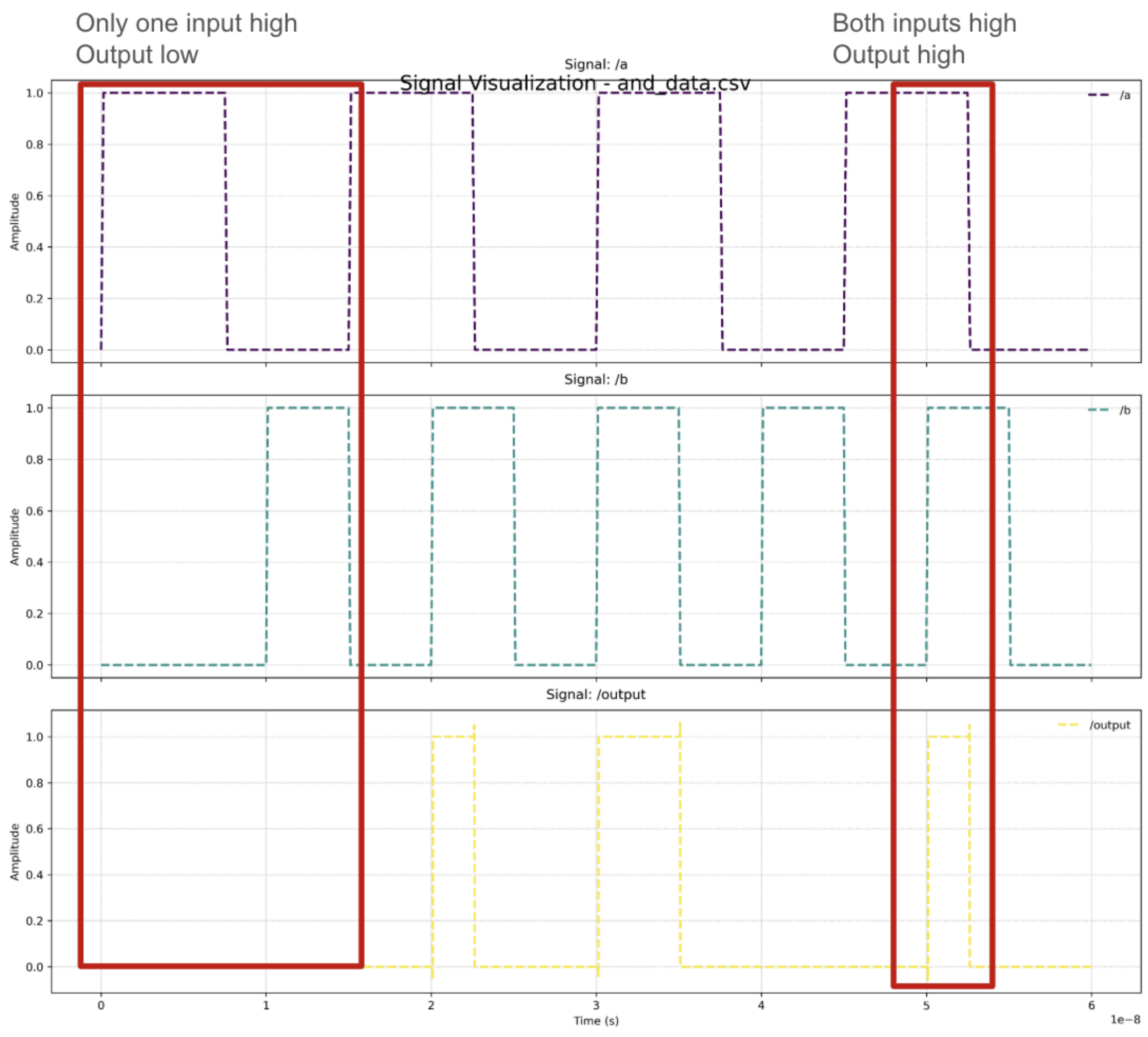


Figure 12 - And Gate Verification

Or Gate

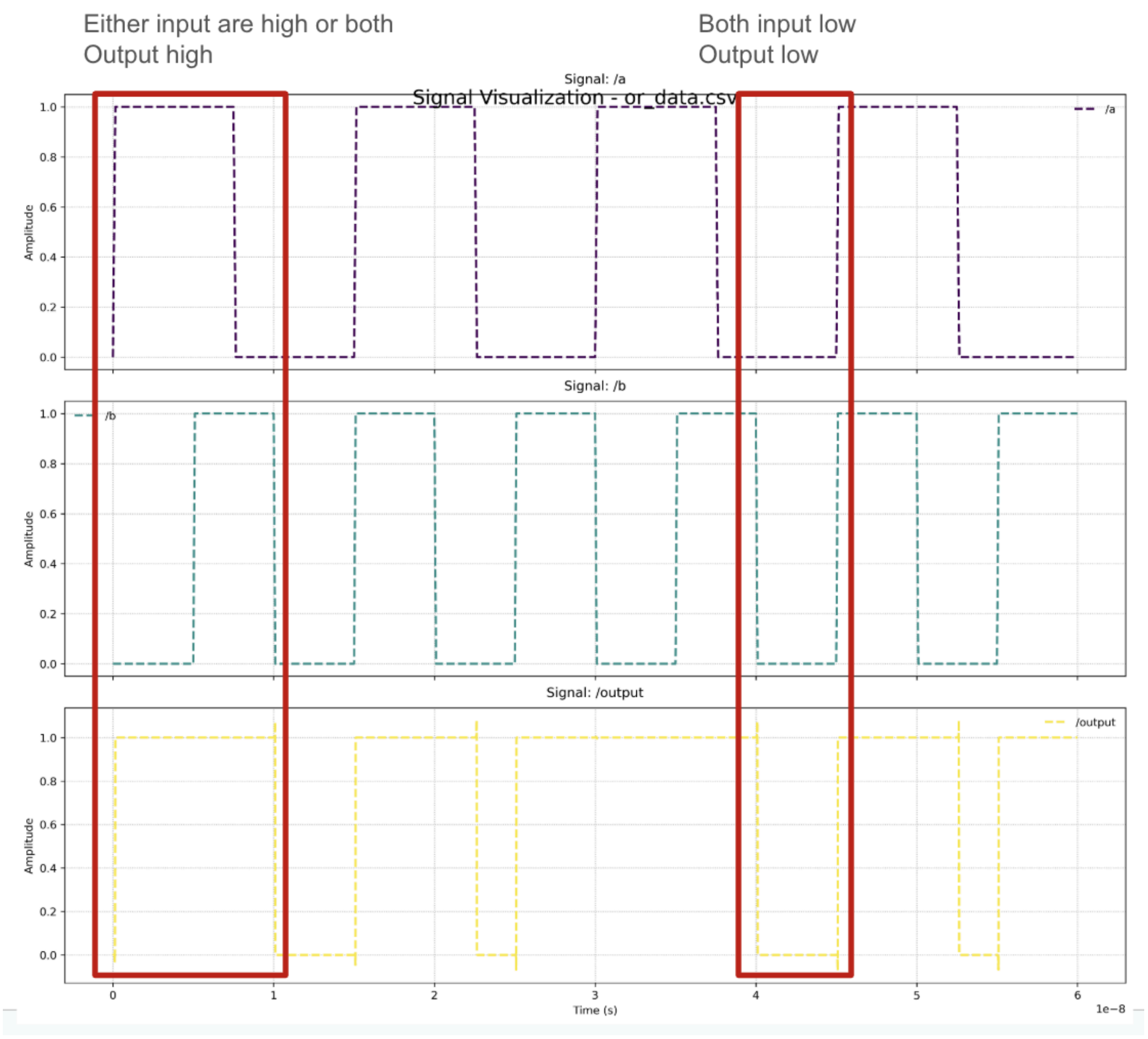


Figure 13 - Or Gate Verification

Xor Gate

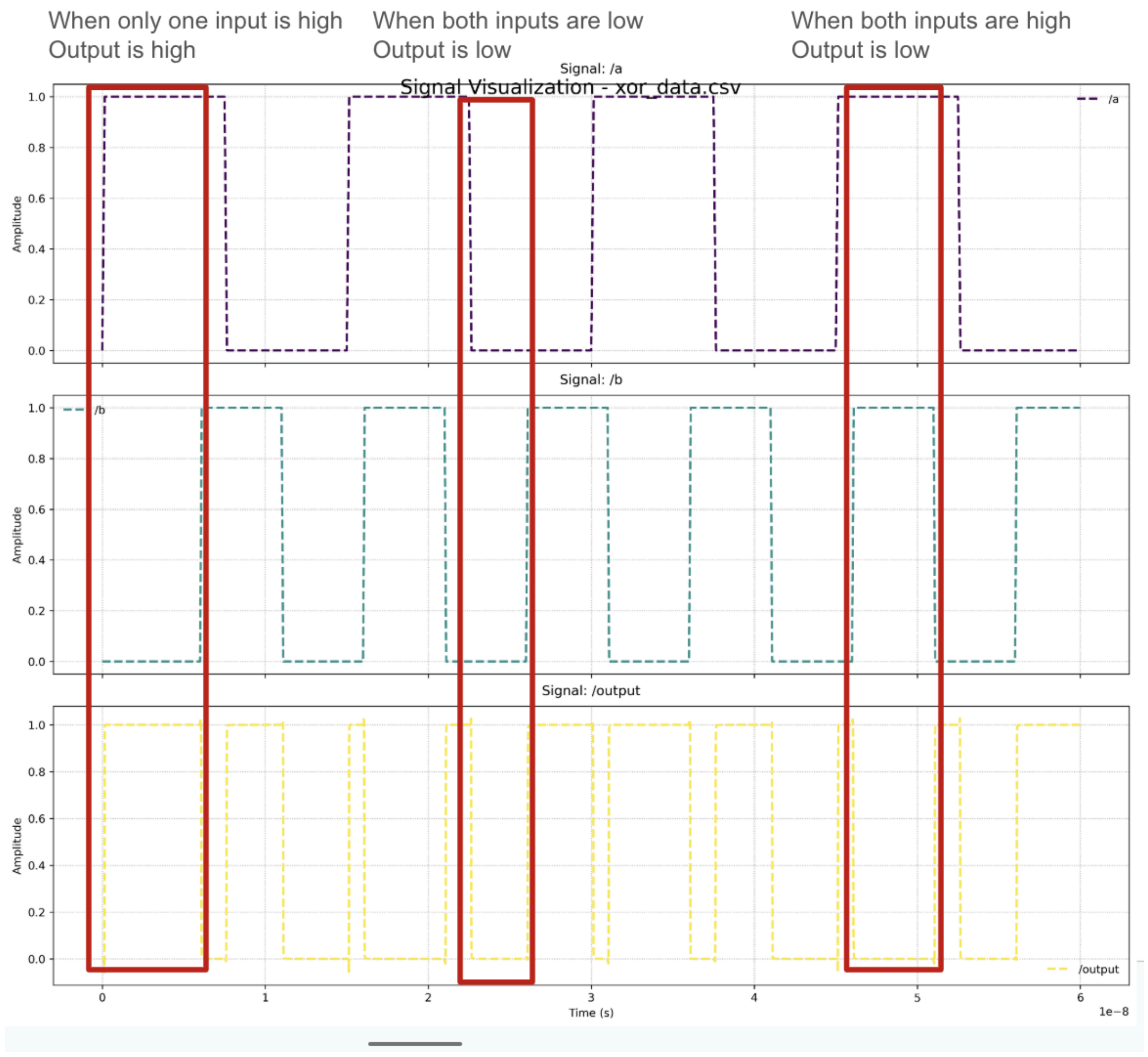


Figure 14 - Xor Gate Verification

Full Adder with Carry and Propagate

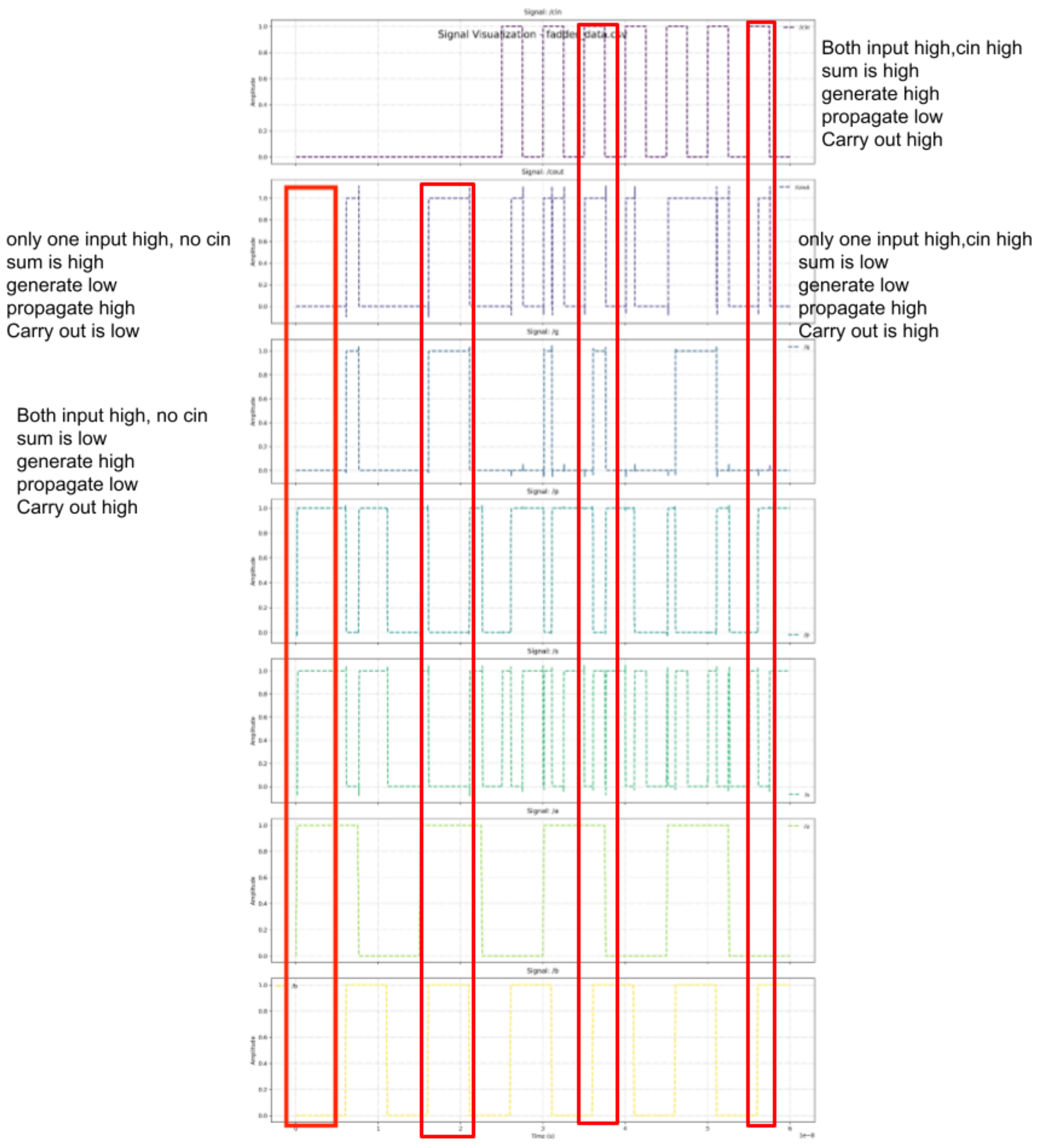


Figure 15 - Full Adder Gate Verification

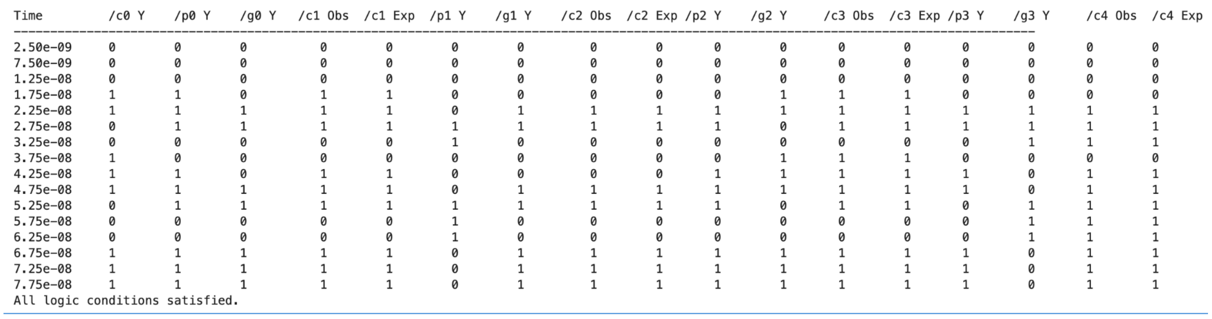
Verification Using Python

Methodology:
Create 9 different random signals in a pwlf file that is used to drive the inputs of the block.

Make a simulation with all the inputs and outputs in Cadence. Export the simulation into CSV files.

Write python code to validate that the actual outputs match expected outputs given the random inputs. To see all python jupyter notebook used for these output go to sim/ (description in Appendix)

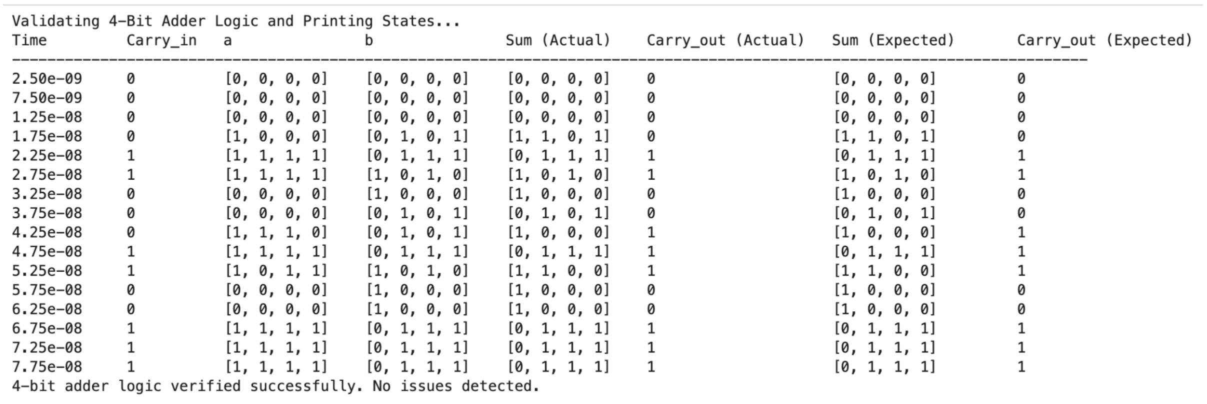
Four Bit Carry Lookahead Combinational Unit



```
Time        /c0 Y    /p0 Y    /g0 Y    /c1 Obs  /c1 Exp  /p1 Y    /g1 Y    /c2 Obs  /c2 Exp /p2 Y    /g2 Y    /c3 Obs  /c3 Exp /p3 Y    /g3 Y    /c4 Obs  /c4 Exp
--------------------------------------------------------------------------------------------------------------------------------------------
2.50e-09    0        0        0          0        0        0        0          0        0        0        0        0        0        0        0        0        0
7.50e-09    0        0        0          0        0        0        0          0        0        0        0        0        0        0        0        0        0
1.25e-08    0        0        0          0        0        0        0          0        0        0        0        0        0        0        0        0        0
1.75e-08    1        1        0          1        1        0        0          0        0        0        1        1        1        0        0        0        0
2.25e-08    1        1        1          1        1        0        1          1        1        1        1        1        1        1        1        1        1
2.75e-08    0        1        1          1        1        1        1          1        1        1        0        1        1        1        1        1        1
3.25e-08    0        0        0          0        0        1        0          0        0        0        0        0        0        0        1        1        1
3.75e-08    1        0        0          0        0        0        0          0        0        0        1        1        1        0        0        0        0
4.25e-08    1        1        0          1        1        0        0          0        0        1        1        1        1        1        0        1        1
4.75e-08    1        1        1          1        1        0        1          1        1        1        1        1        1        1        0        1        1
5.25e-08    0        1        1          1        1        1        1          1        1        1        0        1        1        0        1        1        1
5.75e-08    0        0        0          0        0        1        0          0        0        0        0        0        0        0        1        1        1
6.25e-08    0        0        0          0        0        1        0          0        0        0        0        0        0        0        1        1        1
6.75e-08    1        1        1          1        1        0        1          1        1        1        1        1        1        1        0        1        1
7.25e-08    1        1        1          1        1        0        1          1        1        1        1        1        1        1        0        1        1
7.75e-08    1        1        1          1        1        0        1          1        1        1        1        1        1        1        0        1        1
All logic conditions satisfied.
```

Figure 16 - Four Bit Carry Lookahead Combination Unit Verification

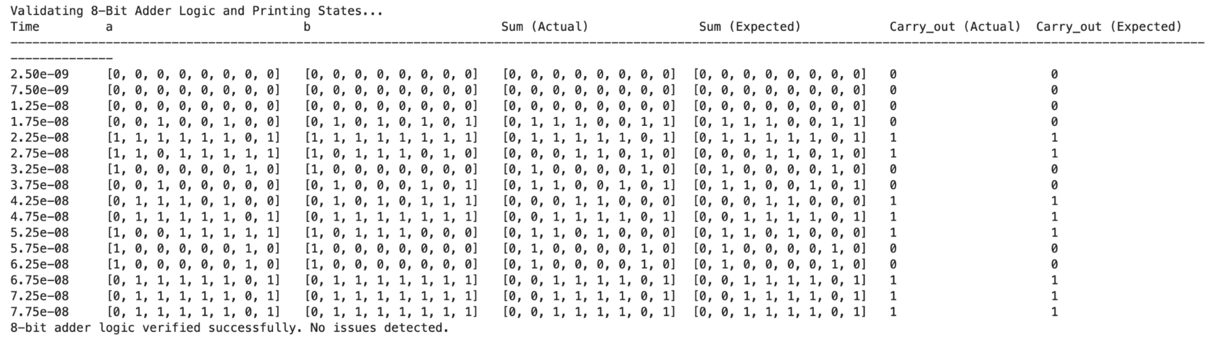
Four Bit Carry Lookahead Adder



```
Validating 4-Bit Adder Logic and Printing States...
Time         Carry_in   a               b               Sum (Actual)    Carry_out (Actual)   Sum (Expected)       Carry_out (Expected)
-------------------------------------------------------------------------------------------------------------------
2.50e-09     0          [0, 0, 0, 0]    [0, 0, 0, 0]    [0, 0, 0, 0]    0                    [0, 0, 0, 0]         0
7.50e-09     0          [0, 0, 0, 0]    [0, 0, 0, 0]    [0, 0, 0, 0]    0                    [0, 0, 0, 0]         0
1.25e-08     0          [0, 0, 0, 0]    [0, 0, 0, 0]    [0, 0, 0, 0]    0                    [0, 0, 0, 0]         0
1.75e-08     0          [1, 0, 0, 0]    [0, 1, 0, 1]    [1, 1, 0, 1]    0                    [1, 1, 0, 1]         0
2.25e-08     1          [1, 1, 1, 1]    [0, 1, 1, 1]    [0, 1, 1, 1]    1                    [0, 1, 1, 1]         1
2.75e-08     1          [1, 1, 1, 1]    [1, 0, 1, 0]    [1, 0, 1, 0]    1                    [1, 0, 1, 0]         1
3.25e-08     0          [0, 0, 0, 0]    [1, 0, 0, 0]    [1, 0, 0, 0]    0                    [1, 0, 0, 0]         0
3.75e-08     0          [0, 0, 0, 0]    [0, 1, 0, 1]    [0, 1, 0, 1]    0                    [0, 1, 0, 1]         0
4.25e-08     0          [1, 1, 1, 0]    [0, 1, 0, 1]    [1, 0, 0, 0]    1                    [1, 0, 0, 0]         1
4.75e-08     1          [1, 1, 1, 1]    [0, 1, 1, 1]    [0, 1, 1, 1]    1                    [0, 1, 1, 1]         1
5.25e-08     1          [1, 0, 1, 1]    [1, 0, 1, 0]    [1, 1, 0, 0]    1                    [1, 1, 0, 0]         1
5.75e-08     0          [0, 0, 0, 0]    [1, 0, 0, 0]    [1, 0, 0, 0]    0                    [1, 0, 0, 0]         0
6.25e-08     0          [0, 0, 0, 0]    [1, 0, 0, 0]    [1, 0, 0, 0]    0                    [1, 0, 0, 0]         0
6.75e-08     1          [1, 1, 1, 1]    [0, 1, 1, 1]    [0, 1, 1, 1]    1                    [0, 1, 1, 1]         1
7.25e-08     1          [1, 1, 1, 1]    [0, 1, 1, 1]    [0, 1, 1, 1]    1                    [0, 1, 1, 1]         1
7.75e-08     1          [1, 1, 1, 1]    [0, 1, 1, 1]    [0, 1, 1, 1]    1                    [0, 1, 1, 1]         1
4-bit adder logic verified successfully. No issues detected.
```

Figure 17 - Four Bit Carry Lookahead Adder Verification Results

Eight Bit Carry Lookahead Adder



```
Validating 8-Bit Adder Logic and Printing States...
Time         a                           b                           Sum (Actual)                Sum (Expected)              Carry_out (Actual)  Carry_out (Expected)
--------------------------------------------------------------------------------------------------------------------------------------------------------------
--------------
2.50e-09     [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   0                   0
7.50e-09     [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   0                   0
1.25e-08     [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   [0, 0, 0, 0, 0, 0, 0, 0]   0                   0
1.75e-08     [0, 0, 1, 0, 0, 1, 0, 0]   [0, 1, 0, 1, 0, 1, 0, 1]   [0, 1, 1, 1, 0, 0, 1, 1]   [0, 1, 1, 1, 0, 0, 1, 1]   0                   0
2.25e-08     [1, 1, 1, 1, 1, 1, 0, 1]   [1, 1, 1, 1, 1, 1, 1, 1]   [0, 1, 1, 1, 1, 1, 0, 1]   [0, 1, 1, 1, 1, 1, 0, 1]   1                   1
2.75e-08     [1, 1, 0, 1, 1, 1, 1, 1]   [1, 0, 1, 1, 1, 0, 1, 0]   [0, 0, 0, 1, 1, 0, 1, 0]   [0, 0, 0, 1, 1, 0, 1, 0]   1                   1
3.25e-08     [1, 0, 0, 0, 0, 0, 1, 0]   [1, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0, 1, 0]   [0, 1, 0, 0, 0, 0, 1, 0]   0                   0
3.75e-08     [0, 0, 1, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 1, 0, 1]   [0, 1, 1, 0, 0, 1, 0, 1]   [0, 1, 1, 0, 0, 1, 0, 1]   0                   0
4.25e-08     [0, 1, 1, 1, 0, 1, 0, 0]   [0, 1, 0, 1, 0, 1, 1, 1]   [0, 0, 0, 1, 1, 0, 0, 0]   [0, 0, 0, 1, 1, 0, 0, 0]   1                   1
4.75e-08     [0, 1, 1, 1, 1, 1, 0, 1]   [0, 1, 1, 1, 1, 1, 1, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   1                   1
5.25e-08     [1, 0, 0, 1, 1, 1, 1, 1]   [1, 0, 1, 1, 1, 0, 0, 0]   [0, 1, 1, 0, 1, 0, 0, 0]   [0, 1, 1, 0, 1, 0, 0, 0]   1                   1
5.75e-08     [1, 0, 0, 0, 0, 0, 1, 0]   [1, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0, 1, 0]   [0, 1, 0, 0, 0, 0, 1, 0]   0                   0
6.25e-08     [1, 0, 0, 0, 0, 0, 1, 0]   [1, 0, 0, 0, 0, 0, 0, 0]   [0, 1, 0, 0, 0, 0, 1, 0]   [0, 1, 0, 0, 0, 0, 1, 0]   0                   0
6.75e-08     [0, 1, 1, 1, 1, 1, 0, 1]   [0, 1, 1, 1, 1, 1, 1, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   1                   1
7.25e-08     [0, 1, 1, 1, 1, 1, 0, 1]   [0, 1, 1, 1, 1, 1, 1, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   1                   1
7.75e-08     [0, 1, 1, 1, 1, 1, 0, 1]   [0, 1, 1, 1, 1, 1, 1, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   [0, 0, 1, 1, 1, 1, 0, 1]   1                   1
8-bit adder logic verified successfully. No issues detected.
```

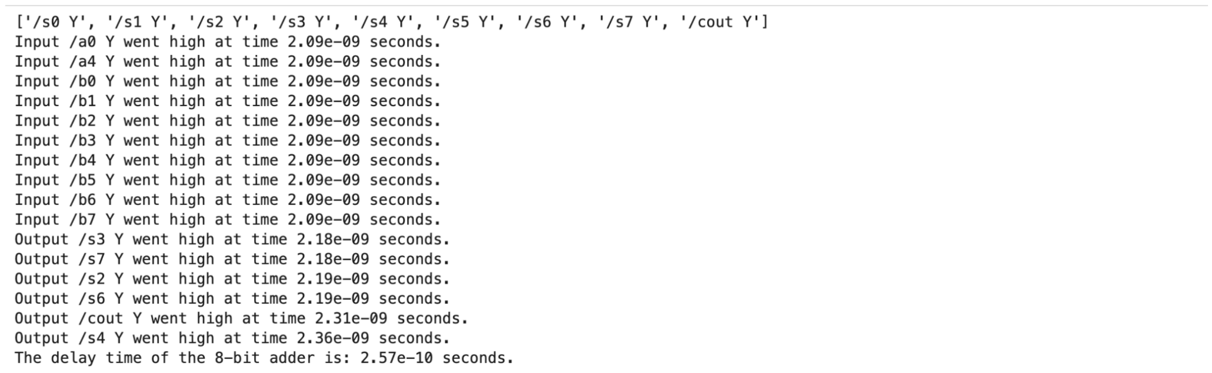
Figure 18 - Eight Bit Carry Lookahead Adder Verification Results

Timing Simulation

The most important part of this project is to find how fast/slow is our fast adder. After verifying that all logical operations work in the design I build a simulation and code to find the worst rise time for the 8-Bit carry lookahead adder.

The worst case time happens when the carry c4 is generated by the longest path which goes through 3 AND gates and 1 OR gate as seen in Figure 6.

Note: the script considers a HIGH to be a signal that crossed the 0.9V threshold, compiling with rise time for a capacitor.



```
['/s0 Y', '/s1 Y', '/s2 Y', '/s3 Y', '/s4 Y', '/s5 Y', '/s6 Y', '/s7 Y', '/cout Y']
 Input /a0 Y went high at time 2.09e-09 seconds.
 Input /a4 Y went high at time 2.09e-09 seconds.
 Input /b0 Y went high at time 2.09e-09 seconds.
 Input /b1 Y went high at time 2.09e-09 seconds.
 Input /b2 Y went high at time 2.09e-09 seconds.
 Input /b3 Y went high at time 2.09e-09 seconds.
 Input /b4 Y went high at time 2.09e-09 seconds.
 Input /b5 Y went high at time 2.09e-09 seconds.
 Input /b6 Y went high at time 2.09e-09 seconds.
 Input /b7 Y went high at time 2.09e-09 seconds.
 Output /s3 Y went high at time 2.18e-09 seconds.
 Output /s7 Y went high at time 2.18e-09 seconds.
 Output /s2 Y went high at time 2.19e-09 seconds.
 Output /s6 Y went high at time 2.19e-09 seconds.
 Output /cout Y went high at time 2.31e-09 seconds.
 Output /s4 Y went high at time 2.36e-09 seconds.
 The delay time of the 8-bit adder is: 2.57e-10 seconds.
```

Figure 19 - Timing Verification Result

Conclusion

A key to improving this design is to create more verification methodology. Tests should be automated for all the different cells in the design. In addition the delay time of each individual cell should be measured in order to identify the critical component that can be optimised. More timing tests for the final design should be made to verify the assumption that I have found the actual worst case delay.

The design of this carry look ahead seems to be successful. A delay of ~0.257ns is considered very good. This indicates that the gate sizing was done correctly and that the design choices paid off. If we tak our design as the critical path of a larger design, i.e. of the most critical path between two flip flop, we can achieve a clock frequency of 1/(worst case delay) which will be 3.9GHz

Appendix and Resources

Simulation and Graphing Code

Location	Description
sim/Graphing.ipynb	Code graphing all csv files in folder Graphing Data/.
sim/Carry Lookahead Comb Verification.ipynb	Verification code for the 4-Bit combinational unit of the carry lookahead adder.
sim/4-Bit Adder Verification.ipynb	Verification code for the 4-Bit carry lookahead adder.
sim/8-Bit Adder Timing Simulation.ipynb	Timing simulation code to find worse time in an 8-Bit carry lookahead data.
sim/8-Bit Adder Verifiction.ipynb	Verification code for an 8-Bit carry lookahead adder.
Graphing Data/	All simulation csv files that don’t have automatic verification. Simulation images/graphs will be saved here.
Simulation Data/	All simulation files that have an automatic verification

Please note, if you are finding the picture or labels to be too small on this document, refer to the corresponding script or folder to open a more clear version of it.

Cadence

The cadence library can be found under carry_lookahead_lib/ .
All cells, schematics, maestro simulations, and symbols can be found there.

Resources

The following resources were used for this project

[1] https://www.geeksforgeeks.org/carry-look-ahead-adder/
[2] https://en.wikipedia.org/wiki/Carry-lookahead_adder
[3] https://www.youtube.com/watch?v=SQKdnxysXnw
[4] Digital Design and Computer Architecture, David Harris