VLSI Design: 16-bit Carry Select Adder

Purdue University Graduate School of Electrical and Computer Engineering
Joey Squillaci

Abstract— **Goal: In this project, you will design a 16-bit Carry-Select Adder (CSA), including its schematic, layout, simulation, functionality verification and energy / timing / area analysis.**

Carry-Select Adder Overview:

The Carry-Select Adder (CSA) is designed to enhance performance by reducing the carry propagation delay, a limitation in traditional ripple-carry adders. In a ripple-carry adder, the carry must propagate through every bit position, which can significantly slow down the operation for larger bit-widths (N). The CSA addresses this by splitting the addition into smaller sub-adders, each calculating two possible sums: one assuming the carry-in is '0' and the other assuming it's '1'. A multiplexer (MUX) then selects the correct sum and carry-out based on the actual carry-in, significantly speeding up the process.

For this project, the 16-bit adder will be divided into smaller blocks (with M=4) to balance speed and complexity. The first four bits (A[3:0], B[3:0]) are processed by a simple 4-bit ripple-carry adder because the carry-in is known at the start. The remaining 12 bits (A[15:4], B[15:4]) are divided into three carry-select blocks, each handling 4 bits. Each carry-select block computes two results in parallel: one with an assumed carry-in of '0' and the other with an assumed carry-in of '1'. A MUX then selects the correct result.

Project Specifications and Guidelines:

Adder Design:

• N = 16 (for the 16-bit adder).
• M = 4 (divider for sub-adders in the CSA).
• Input rise/fall time: 50ps.
• Output load: 2fF.

Layout Design:

• Layout must be DRC-clean and LVS-clean.
• Extract parasitics from the layout and report post-layout energy and propagation delay values.

Evaluation Metrics:

• Pass functionality check.
• Worst-case propagation delay: ≤ 2ns.
• Energy consumption (considering the same input vectors as the delay analysis): ≤ 700fJ.
• Minimize the area while adhering to the above performance constraints.

Worst-Case Scenario:

• Identify the worst-case conditions for delay (e.g., longest carry chain) and report the corresponding metrics.

You will need to justify your worst-case input(s).

I. Introduction

Detailed in this report will be the design, construction, and analysis of a 16-bit carry select adder, while the abstract outlined the design requirements and guidelines for the system. The 16-bit carry select adder is made of blocks, whose makeup consists of adders and logic gates, whose makeup consists of NMOS and PMOS transistors. The design incorporates an equal number of NMOS and PMOS transistors, allowing for a perfect CMOS implementation.

II. Design of Gates

A. *Schematic Design & Transistor Sizing*

The selected design of the 16-bit CSA requires the following logic gates: inverter, NAND, AND, NOR, OR. These gates are required for the construction of the MUX variants, which are integral to the function of the carry-select system.

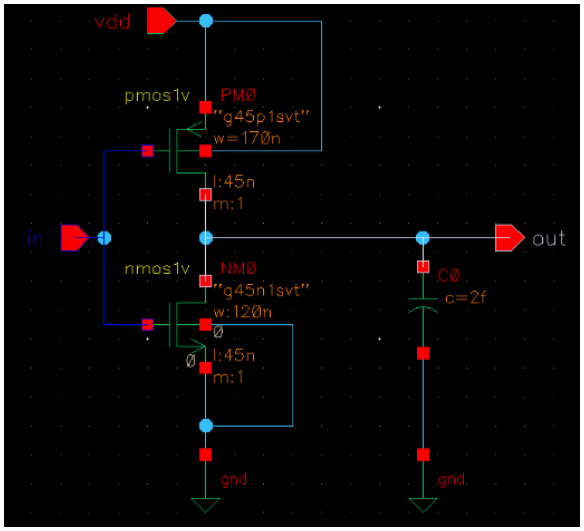


Figure [1]: Example of CMOS-based inverter schematic.

Considering the provided constraints (refer to abstract), the logic gates were designed with equal rise and fall times in mind. The table below outlines the transistor sizing used for each gate; obtained by sweeping the PMOS: NMOS widths in each logic

gate schematic. *Note: AND/OR transistor sizing is not available as these gates were made using the respective NAND/NOR equivalent with an inverter.

Logic Gate	P:N Ratio	Propagation Delays (Rise/Fall)
Inverter	1.4:1	35.4ps/35.81ps
NAND	0.75:1	47.87ps/47.72ps
NOR	2.8:1	37.81ps/38.12ps

Figure [2]: Logic gate ratios for equal rise/fall propagation delays.

B. *Logic Gate Functional Analysis*

For the sake of providing an example of functional analysis, the screenshot below shows the input/output combination for the OR gate. All logic gates passed a functional analysis check similar to the screenshot below:

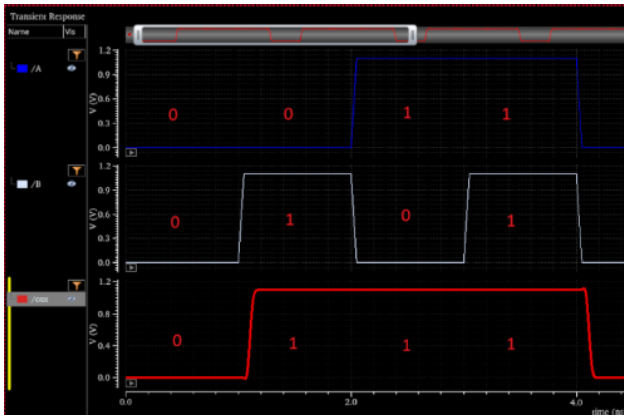


Figure [3]: Logic gate functional analysis example: OR gate

III. Design of Adder, MUX Variants

A. *Full Adder*

The design of the adder system is a two-part approach: creation of both a full adder used to create a 4-bit ripple carry adder. Though the logic gates can be used to construct the full adder, it was found that utilizing the mirror topology was the most efficient implementation of the full adder. Comparing these two approaches had shown a vast difference in propagation delay – where the logic gate design implementation exceeded the project's ≤ 2ns delay requirement, and the mirror topology yielded < 200ps propagation delay.

B. *4-bit Ripple Carry Adder*

The 4-bit Ripple Carry Adder was designed using four full adders; chained together to allow the carryout of the previous adder to feed into the carry in of the following adder.

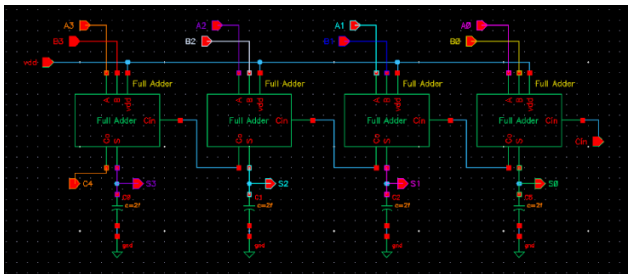


Figure [4]: 4-bit Ripple Carry Adder schematic.

C. *MUX Variants*

The 16-bit CSA requires the use of a 2-bit and 4-bit MUX to select the proper carry-in and summation bits. The primary use for the 2-bit MUX is to select the proper carry-out to be passed in as the next RCA's carry-in selector. The select for this MUX is based on the previous (correctly selected) RCA's carry-out. A secondary use for the 2-bit MUX is to be used as a building block for the 4-bit MUX. The 4-bit MUX is used to select which summed bits are passed to the output, coming from the RCA with the correct carry-in of either 0 or 1 (selected using the 2-bit MUX carry-out from the previous block). As a generalization, the MUXes are used to choose the proper carry-out and summed bits.

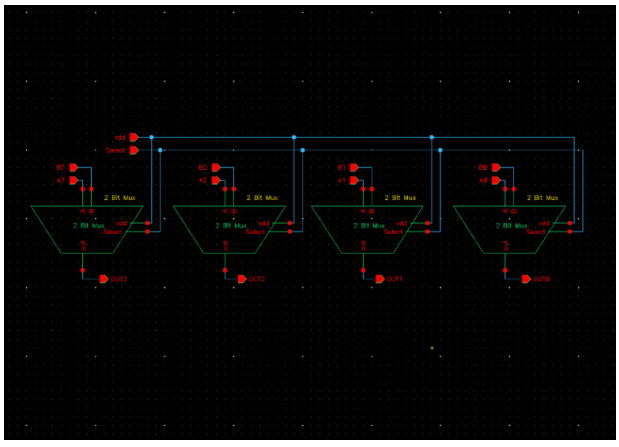


Figure [5]: 4-bit MUX constructed using four 2-bit MUXes.

D. *RCA & MUX Block*

Considering the RCA and MUX combination are consistent for each 4-bit block for the 12 most significant bits (three identical blocks of four bits each), it was decided to create an RCA and MUX combination block. This block helps to simplify the final schematic down to three "RCA and MUX Selector Blocks" and one 4-bit RCA. The RCA/MUX block consists of two RCAs (one corresponding to carry-in: 0, one corresponding to carry-in: 1), a 2-bit MUX to select the proper carry-out, and a 4-bit MUX to select the proper summed bits.

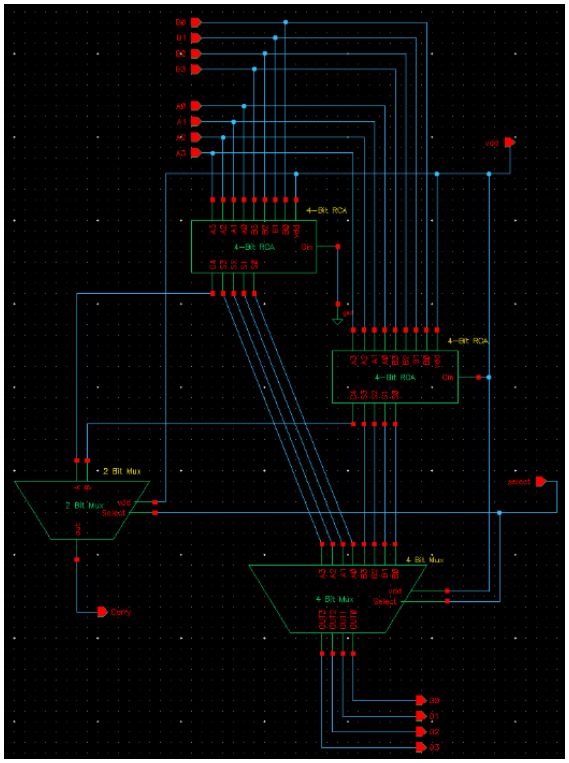


Figure [6]: RCA and MUX Selector Block

IV. 16-bit Carry Select Adder Schematic

A. *16-bit CSA Overview*

The 16-bit Carry Select Adder is made up of seven 4-bit RCAs, three 2-bit MUXes, and three 4-bit MUXes. When adding the two 16-bit inputs, the first four bits are handled by a 4-bit RCA, while each of the next 4-bit blocks (from OUT4:OUT15, 12 bits – total of three 4-bit blocks) are handled by the remaining RCAs and MUXes in the RCA and MUX Selector Block.

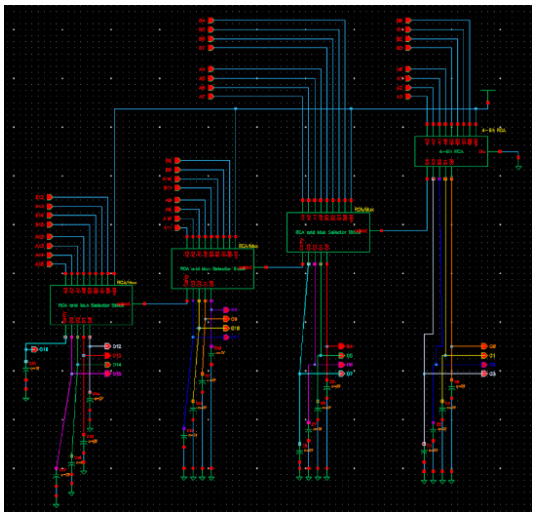


Figure [7]: 16-bit CSA Schematic

B. *16-bit CSA Schematic Functional Analysis*

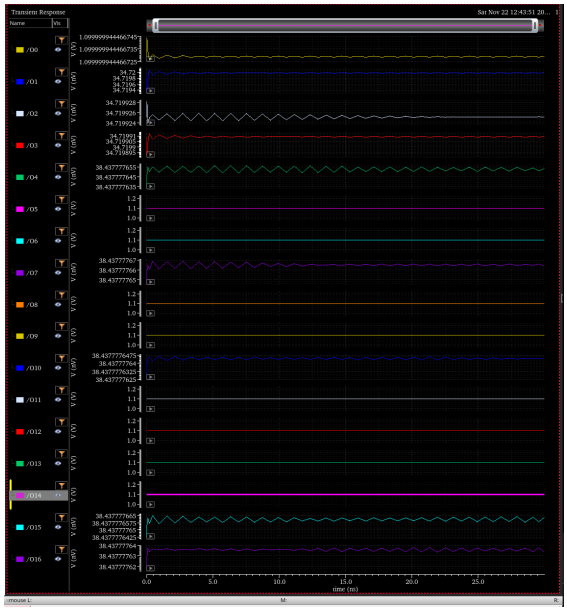


Figure [8]: CSA Schematic Functional Analysis Case One

Three cases were tested for functional analysis, the first case is shown above. The second and third case will be detailed following the first case, where the corresponding figures can be found in the references section.

Case One:

A	=	0010110100101100
B	=	0100111000110101
Output	=	0111101101100001

Case Two:

A	=	0001001001000101
B	=	1010111100001110
Output	=	1100000101010011

C. *16-bit CSA Worst Case Delay and Energy Analysis*

Now that the functionality of the **schematic** has been confirmed, it's time to take the worst-case approach and analyze the performance of the CSA. The worst case occurs when the most number of inputs need to flip to the opposite bit, while having the most carries possible. With this in mind, the worst case inputs would be:

A_{old} = 0000 0000 0000 0000
B_{old} = 0000 0000 0000 0000
Out_{old} = 0000 0000 0000 0000

A_{new} = 1111 1111 1111 1111
B_{new} = 0000 0000 0000 0001
Out_{new} = 1 0000 0000 0000 0000

Running these stimuli in the simulation yields propagation delay of (rising/falling) of 394.1ps/426.7ps (average 410.4ps). Given the parameters of the project allowing for at most a 2ns delay, it can be said that this CSA design falls within the acceptable bounds.

Now that the worst case inputs have been defined, collecting the energy expenditure is as simple as integrating the power of the system over a specified period of time where the current spike is observed.

$$\int_{t1}^{t2} p(t)\,dt = \int_{t1}^{t2} v(t) * i(t)dt$$

Figure [9]: Energy Analysis Equation

It was observed that the current spiked from ~ 1ns → 2ns, which will be the bounds for the integration. With the equation and bounds identified, the worst case energy analysis yielded 121.1fJ. Similar to the worst case delay, 121.1fJ falls within the "≤ 700fJ" requirement, again verifying the design of this 16-bit CSA.

V. 16-bit Carry Select Adder Layout

A. *16-bit CSA Layout Overview*

With a basis in mind for how the schematic performs, the next step was to physically lay out the transistors to gather more accurate, real world performance. This was completed using the Cadence Virtuoso "XL Layout" editor, utilizing two layers of metal to ensure functionality. The CSA was laid out in a hierarchical manner; first the logic gates were laid out, used to lay out the adders and MUXes, which were finally used as blocks for the final layout of the CSA. Each block was constructed with ample space in mind to allow for minimal errors. The final layout area measured 388.83 units x 257.47 units, resulting in a total active area of ~100,112 units^2. Though the ruler tool did not specify the units, it is believed that these units are in terms of micrometers (µm). The upside to this approach is faster design time as less errors are encountered, though the downside is that longer and larger traces, paths, and blocks mean longer delay times and larger energy expenditure.

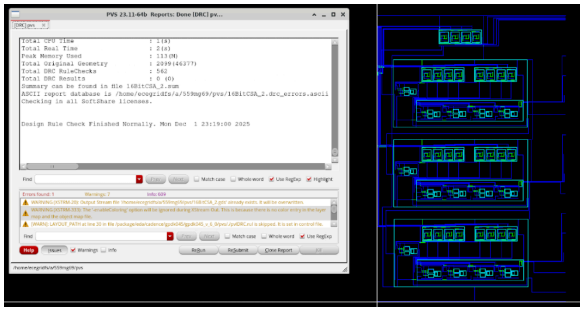


Figure [10]: 16-bit CSA DRC Clean Check

Figure [11]: 16-bit CSA LVS Clean Check

B. *16-bit CSA Layout Worst Case Delay and Energy Analysis*

Utilizing the system's completed and DRC/LVS clean layout, the worst case delay and energy analysis can now be collected. Furthermore, a visual comparison of the signals can be observed by running the schematic-based simulation, and then appending the "AV Extracted" (layout-based)

simulated outputs to the same graph. The compilation of these curves allows the user to visually compare the ideal square-like inputs/outputs of the schematic to the more realistic curve-like waveforms of its physical counterpart.

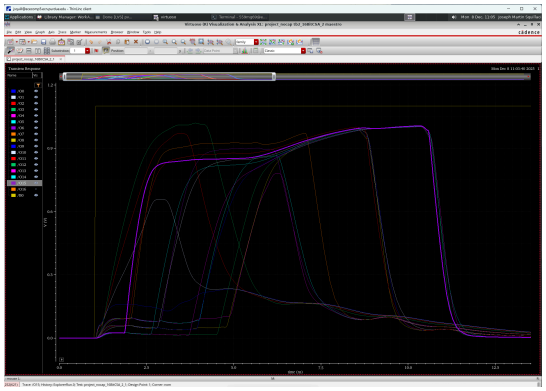


Figure [12]: Graph of 16-bit CSA layout outputs

The figure above shows all the outputs of the "AV Extracted" (layout) view. In this figure, the one square wave is the LSB of the input (B0) changing from low to high (0 to 1) and the highlighted purple curve is the MSB of the output (OUT15). Using a similar process to obtaining calculations from the schematic, the worst case propagation delay from the LSB of the input (B0) to the MSB of the output (OUT15) comes out to 1.245ns. This value falls within the requirements of the system and thus validates the layout.

The energy analysis yielded more issues, as the simulation consistently returned an "eval error". Figure shown below, it is believed that the reason for this is that the VDD instance was changed to a VDD pin during the layout creation. This change means that the node the simulation looks for for v(t) and i(t) data cannot be found, and thus it can only return null data.

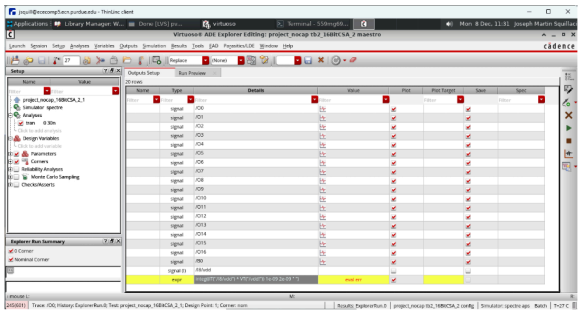


Figure [13]: 16-bit CSA Layout Energy "Eval Error"

Though the calculation returns an "eval error", one can still estimate energy expenditure based on proportionalities obtained through the worst case delay analysis. This estimation is *not* an accurate representation of the real world layout, however it proves as a supplementation to the report given there is no calculated energy value.

The estimation method is as follows: analyze the proportion of how much slower the delay was from the layout to the schematic, and apply this proportion to the energy analysis.

	Schematic	Layout	Proportion
Worst Case Delay	410.4ps	1.245ns	1:3.03

Figure [14]: Proportion Details of Schematic vs Layout Delays

Knowing the delay was 3.03x slower for the layout compared to the schematic, this proportion can be applied to the schematic's energy expenditure to estimate the energy used in the layout. Taking the 121.1fJ from the schematic and multiplying it by 3.03 yields an estimated energy used by the layout of 366.9fJ, a value that still falls within the "≤ 700fJ" requirement.

VI. Conclusions

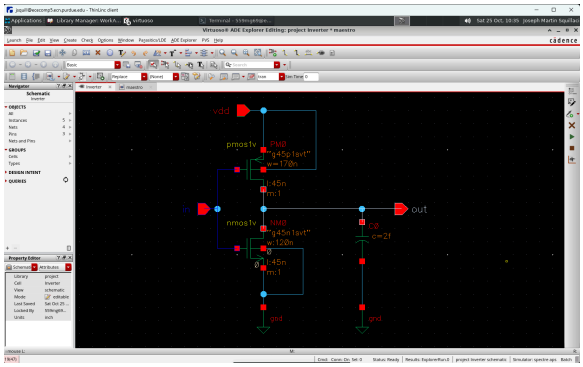
To conclude this experiment, it can be said that the 16-bit Carry Select Adder was successfully designed and verified, meeting all of the delay and energy requirements. The CSA demonstrated its ability to perform efficiently, with the final metrics resulted in:

- The schematic achieved an average delay of 410.4ps and energy of 121.1fJ.
- The final DRC/LVS-clean layout showed a worst case delay of 1.245ns and estimated energy analysis of 366.9fJ.

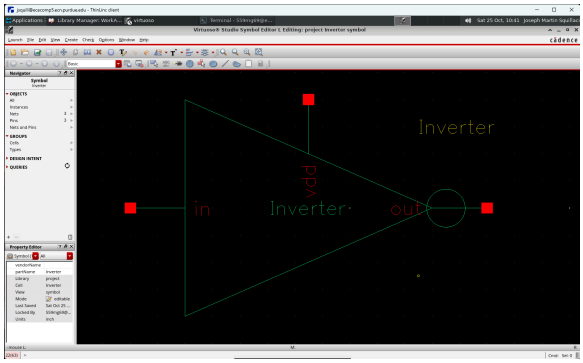
Despite encountering an "eval error" during layout calculations, the layout's energy was able to be estimated using a method centered around the proportionality of the delay increase, leading us to the conclusion that the physical design also falls within the bounds of the specifications. The construction of this system validates that the CSA is an efficient solution in VLSI design, though future emphasis should focus on improving the layout design to reduce the overall worst case delay and energy analyses.

VII. References

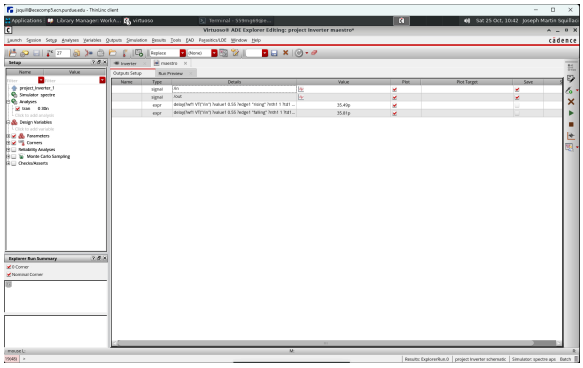
(Not explicitly mentioned/used in report)



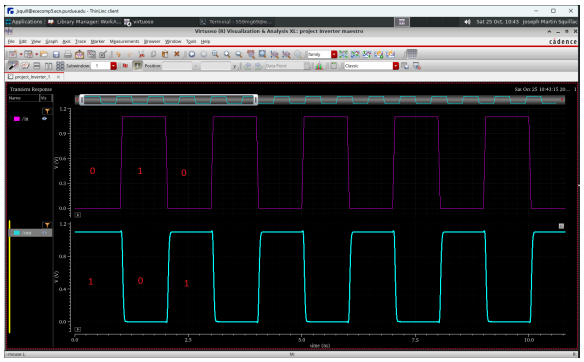
[1] Inverter schematic with Wp = 170nm, Wn = 120nm



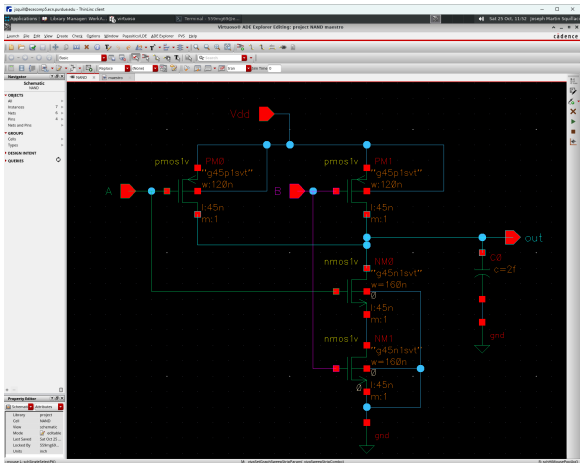
[2] Inverter symbol that will be used for future construction of NAND and MUX.



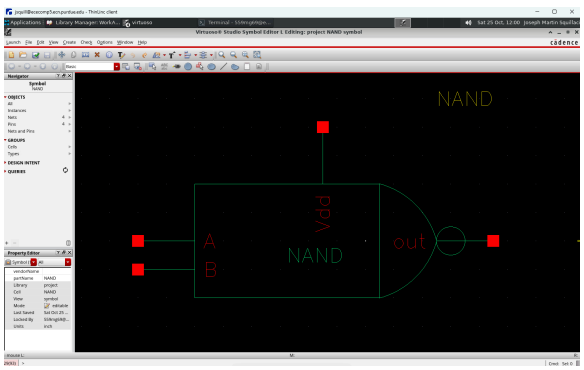
[3] Inverter delay measurements proving equal low-to-high and high-to-low delays at a PMOS to NMOS width ratio of 1.4x.



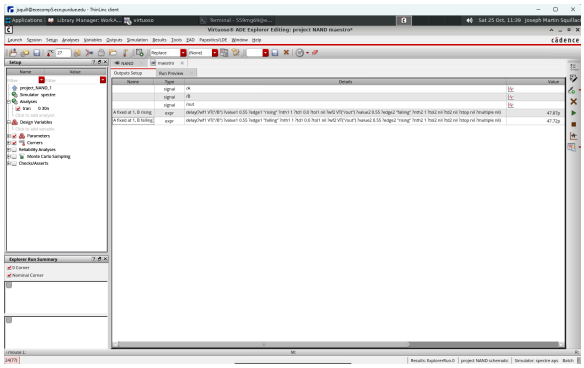
[4] Waveforms proving the inverter's functionality.



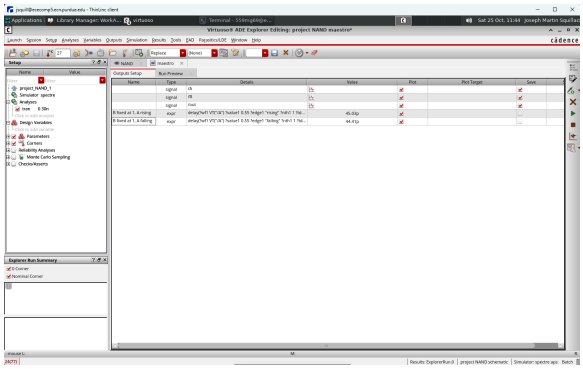
[5] NAND schematic with Wp = 120nm, Wn = 160nm



[6] NAND symbol that will be used for future construction of AND and Full Adder.

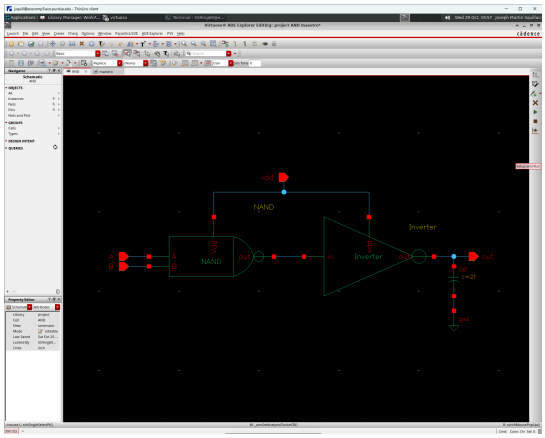


[7] NAND delay measurements proving equal delays for A fixed at 1, B rising/falling for PMOS to NMOS width ratio of 0.75x.

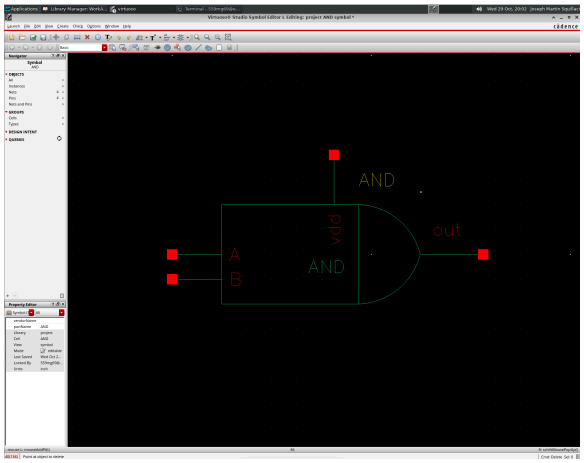


[8] NAND delay measurements proving equal delays for B fixed at 1, A rising/falling for PMOS to NMOS width ratio of 0.75x.

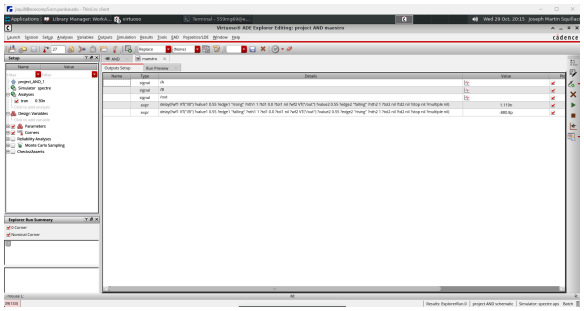
[9] Waveforms proving the NAND's functionality.



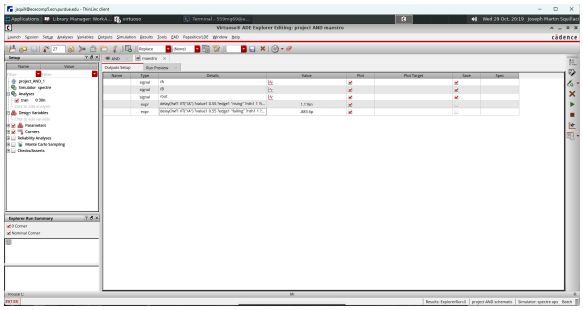
[10] AND schematic implementing the NAND and Inverter symbols.



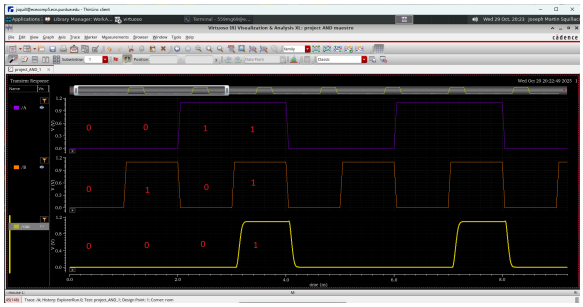
[11] AND symbol that will be used for future construction of AND and Full Adder.



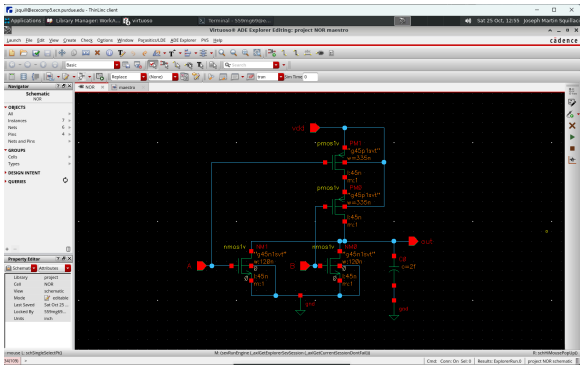
[12] AND delay measurements proving as close to equal delays as possible for A fixed at 1, B rising/falling.



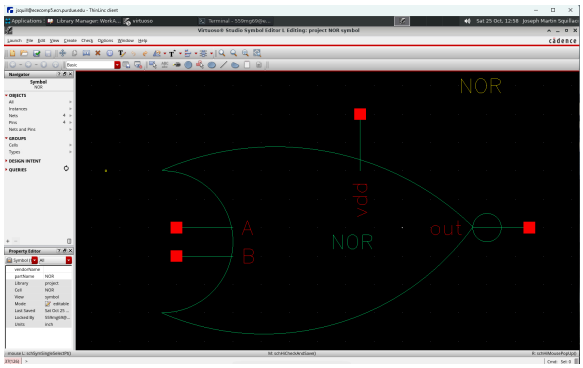
[13] AND delay measurements proving as close to equal delays as possible for B fixed at 1, A rising/falling.



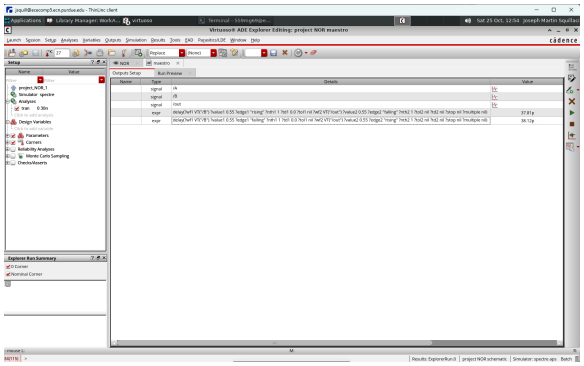
[14] Waveforms proving the AND's functionality.



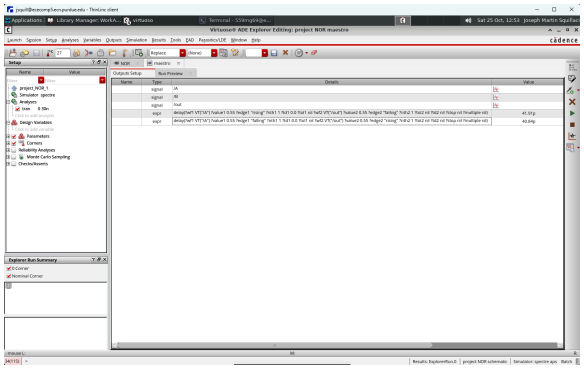
[15] NOR schematic with Wp = 335nm, Wn = 120nm



[16] NOR symbol that will be used for future construction of OR.

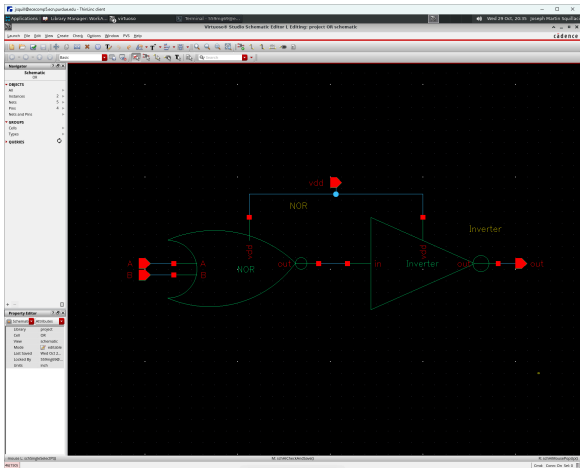


[17] NOR delay measurements proving nearly equal delays for A fixed at 0, B rising/falling for PMOS to NMOS width ratio of ~2.8x.

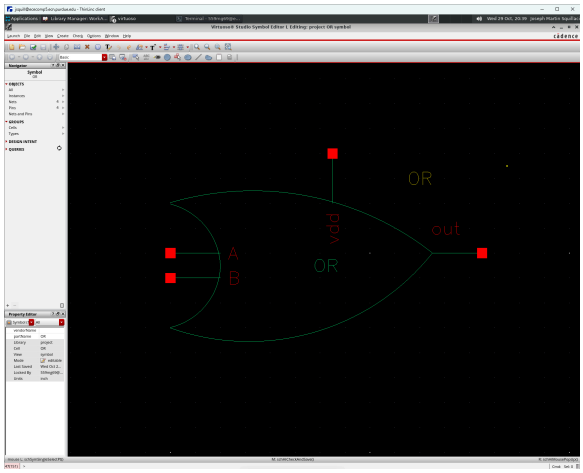


[18] NOR delay measurements proving nearly equal delays for B fixed at 0, A rising/falling for PMOS to NMOS width ratio of ~2.8x.

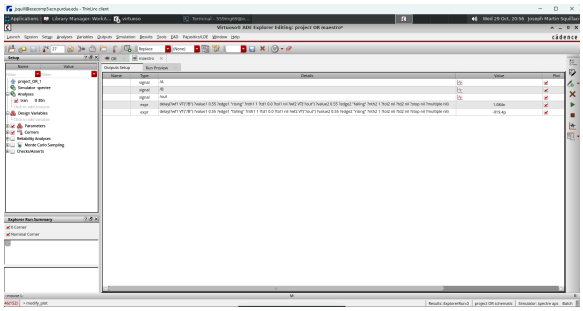
[19] Waveforms proving the NOR's functionality.



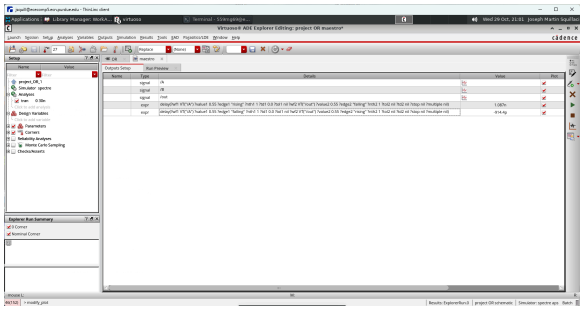
[20] OR schematic implementing the NOR and Inverter symbols.



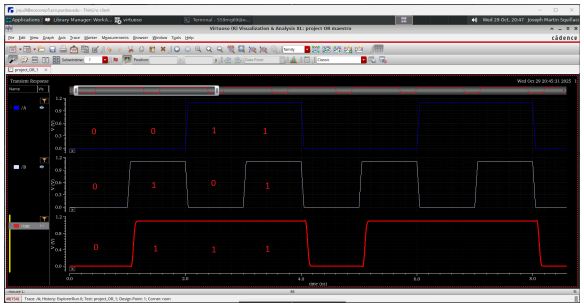
[21] OR symbol that will be used for future construction of other logic blocks.



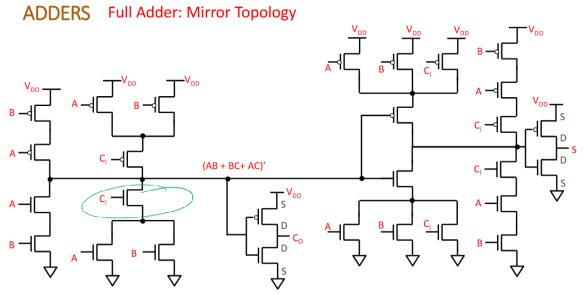
[22] OR delay measurements proving as close to equal delays as possible for A fixed at 0, B rising/falling.



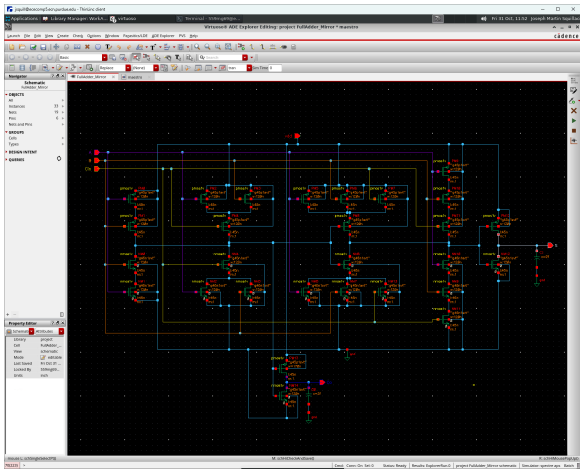
[23] OR delay measurements proving as close to equal delays as possible for B fixed at 0, A rising/falling.



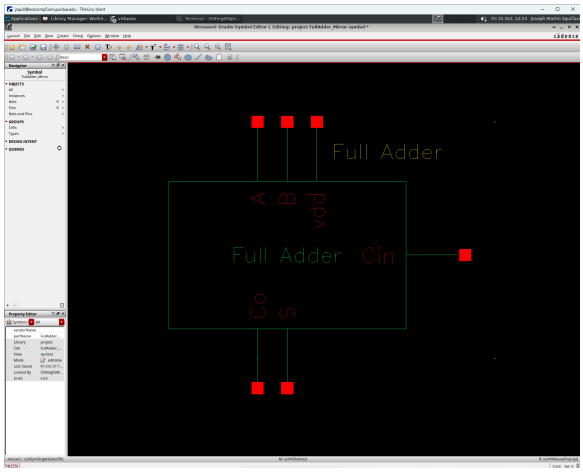
[24] Waveforms proving the OR's functionality.



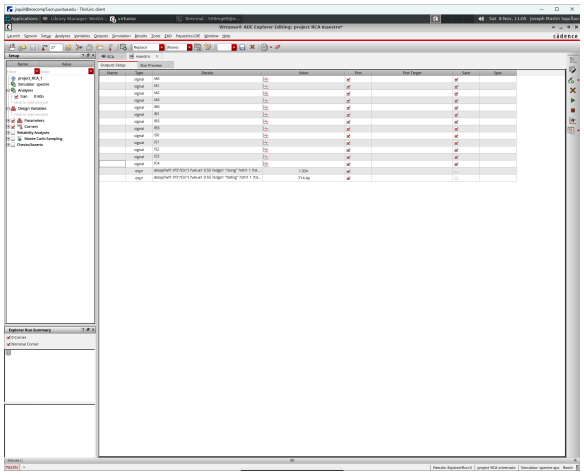
[25] Mirror Topology from Lecture 2: Fundamentals of CMOS Circuits



[26] Full Adder schematic utilizing mirror topology.

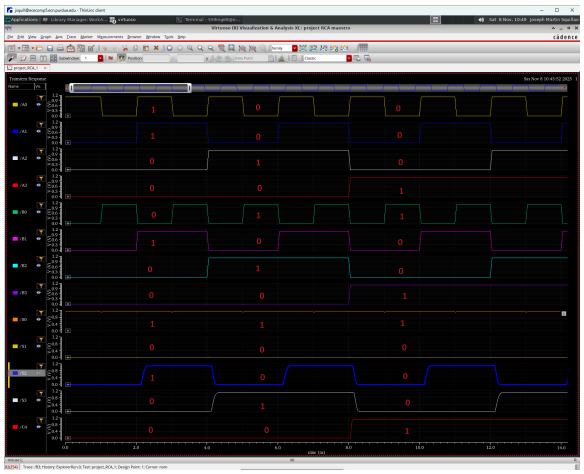


[27] Full Adder symbol that will be used for future construction of the RCA.

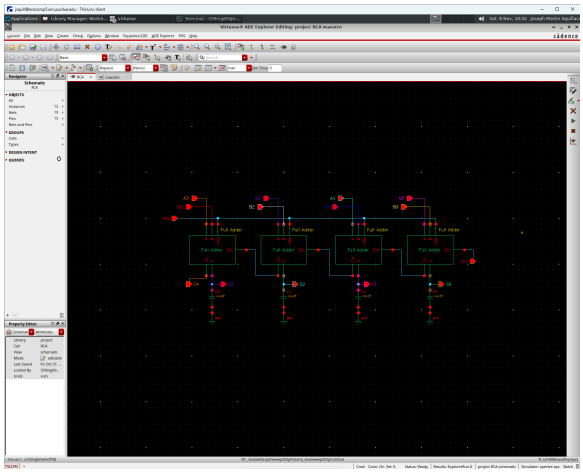


[30] 4-bit RCA worst case delay, corresponds to A = 1111, B = 0000, Cin = 1.

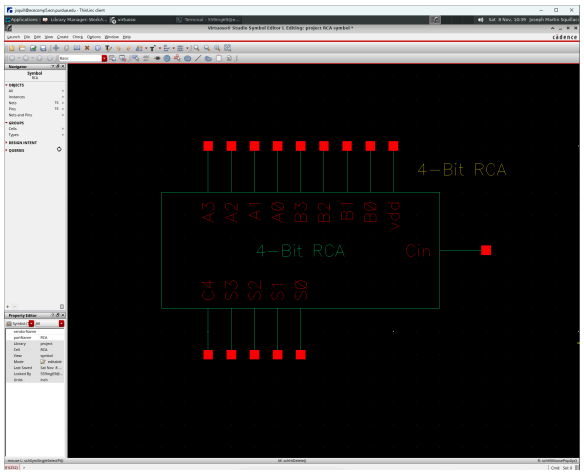
[28] Waveforms proving the Full Adder's functionality.



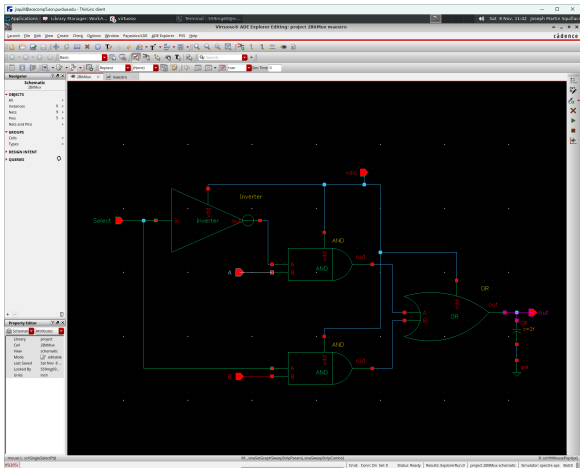
[31] Example of Functional Validation for 4 Bit RCA



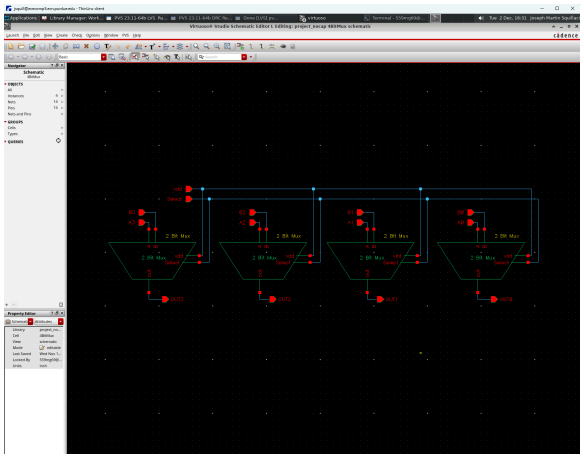
[29] RCA schematic utilizing full adders



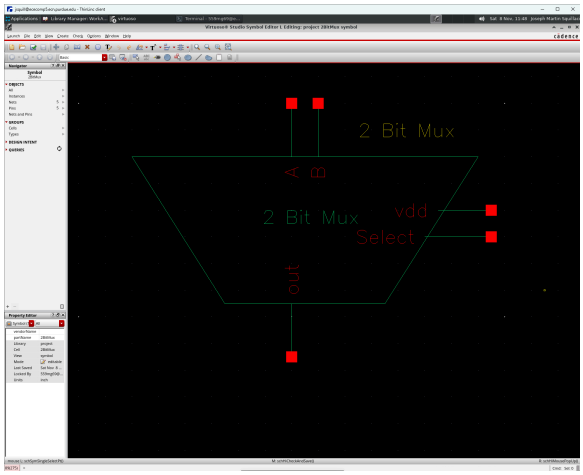
[32] 4-bit RCA Symbol



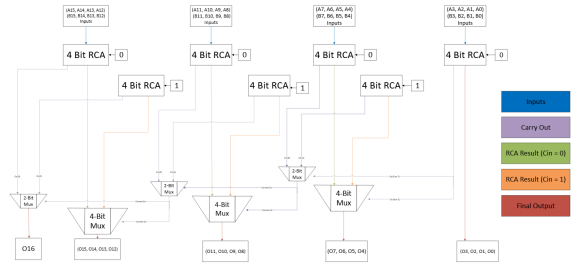
[33] 2-bit MUX Schematic



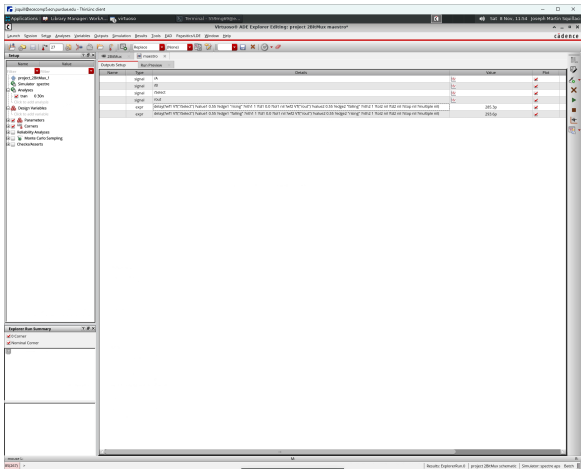
[36] 4-bit MUX Schematic using 2-bit MUXes



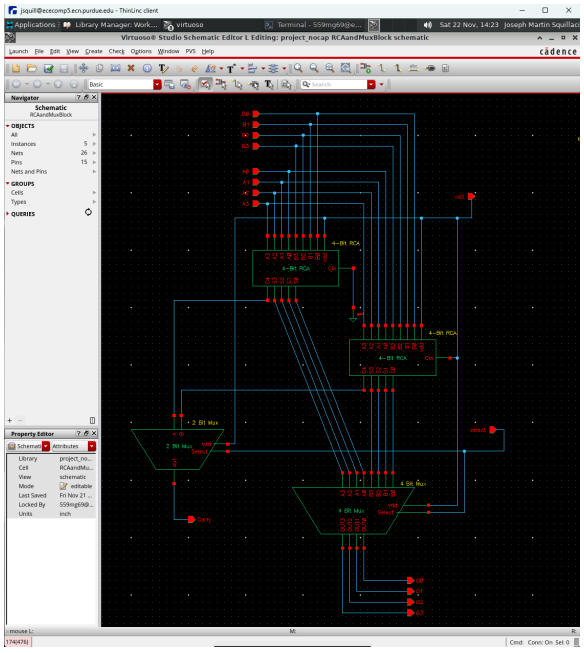
[34] 2-bit MUX Symbol



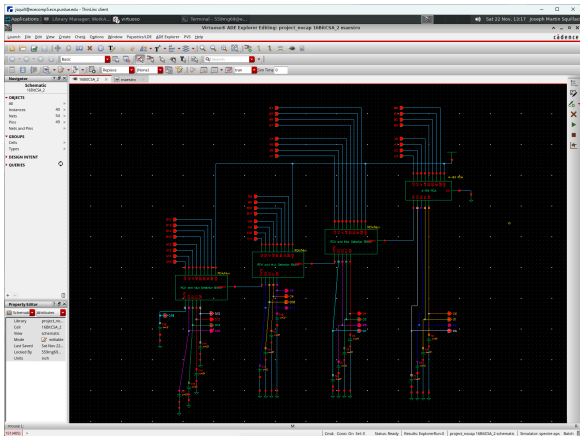
[37] 16-bit Carry-Select Adder Block Diagram, created by Joey Squillaci



[35] 2-bit MUX A fixed at 1, B fixed at 0, Select Rising and Falling

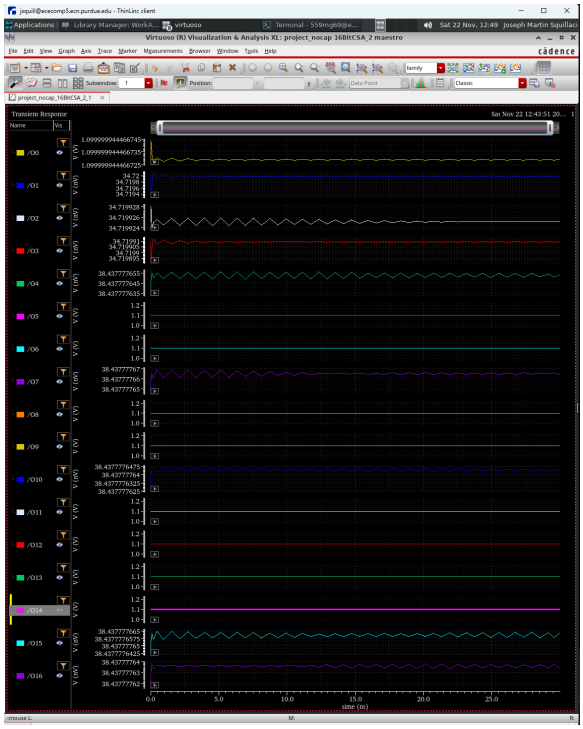


[38] RCA and Mux Selector Block Schematic

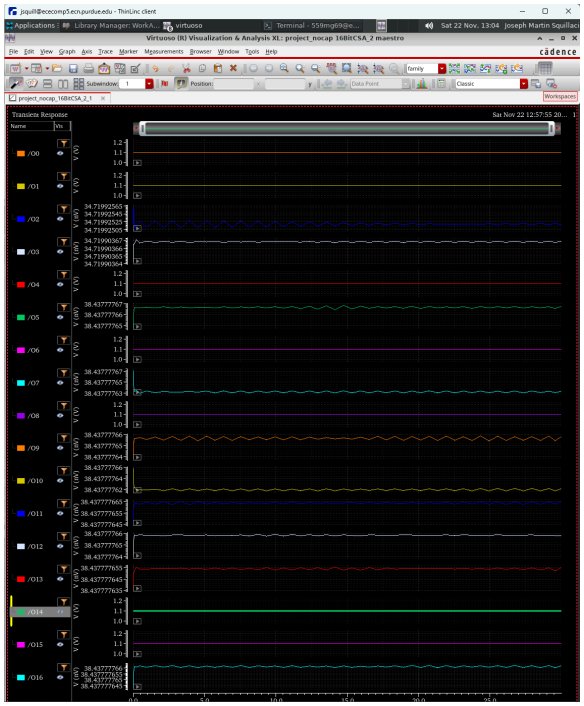


[39] 16-bit CSA Schematic

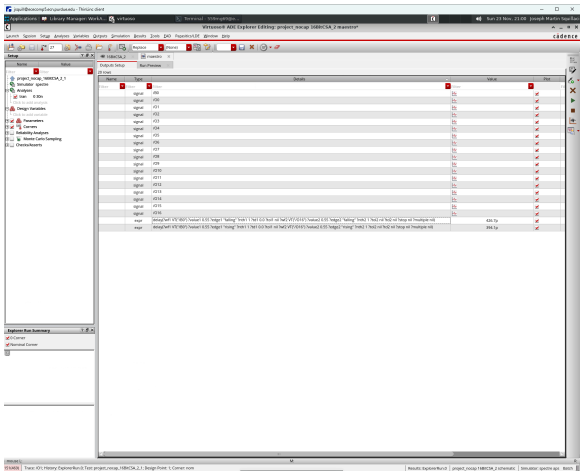
Functional Validation
A = 0010110100101100
B = 0100111000110101
Output = 0111101101100001



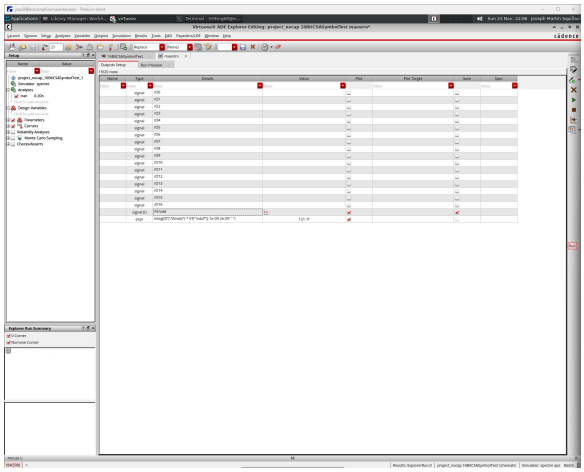
[40] 16-bit CSA Functional Validation for:
A = 0010110100101100, B = 0100111000110101,
Output = 0111101101100001



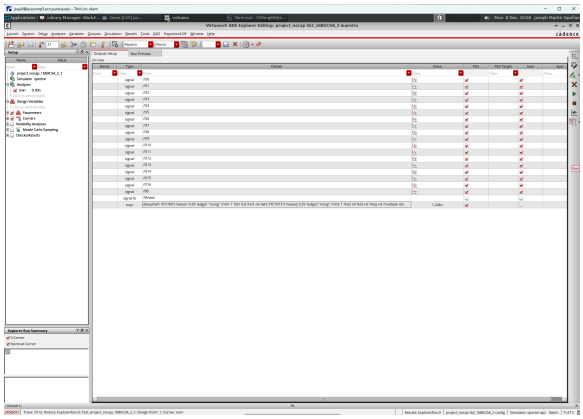
[41] 16-bit CSA Functional Validation for: A = 0001001001000101, B = 1010111100001110,
Output = 1100000101010011



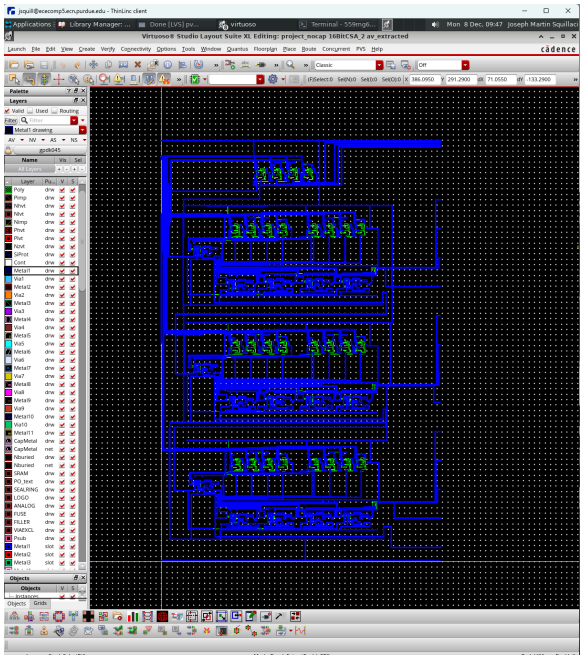
[42] 16-bit CSA worst case delay (schematic)



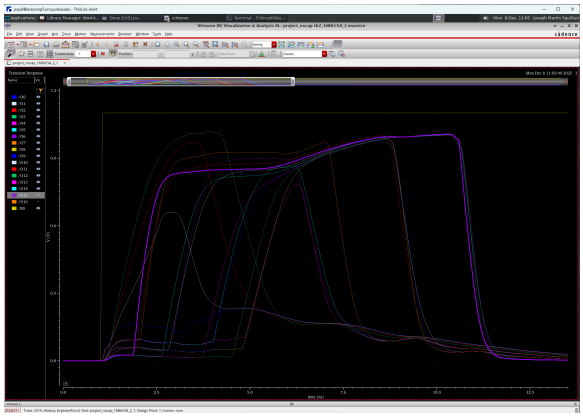
[43] 16-bit CSA worst case energy analysis (schematic)



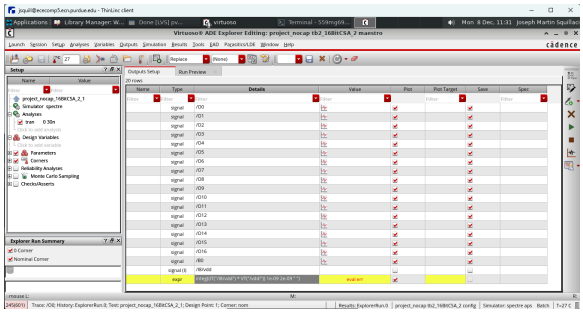
[46] 16-bit CSA AV Extracted Worst Case Timing Delay



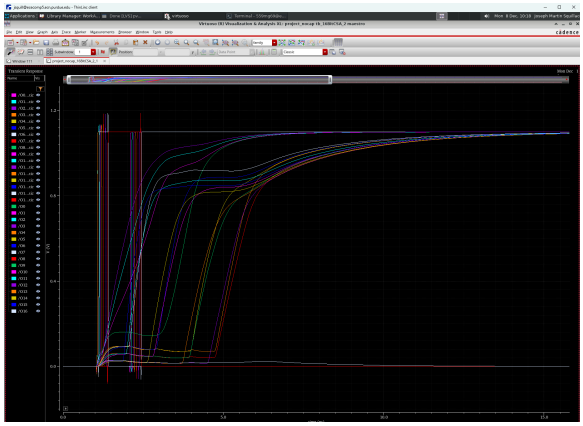
[44] 16-bit CSA AV Extracted/Layout view



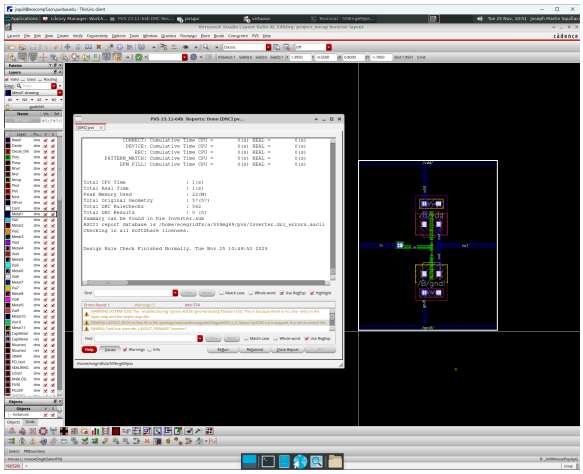
[47] 16-bit CSA worst case propagation delay curves



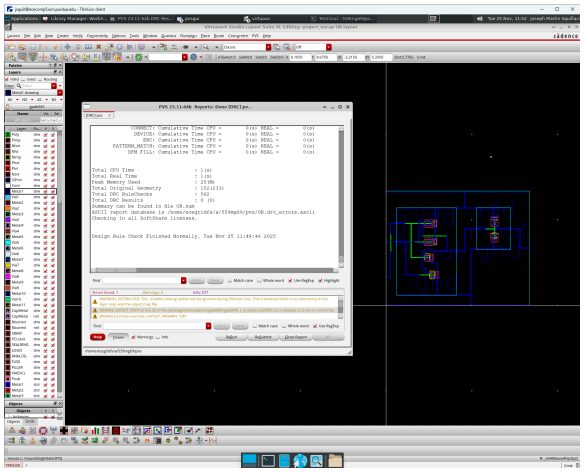
[48] 16-bit CSA AV Extracted “eval error” energy calculation



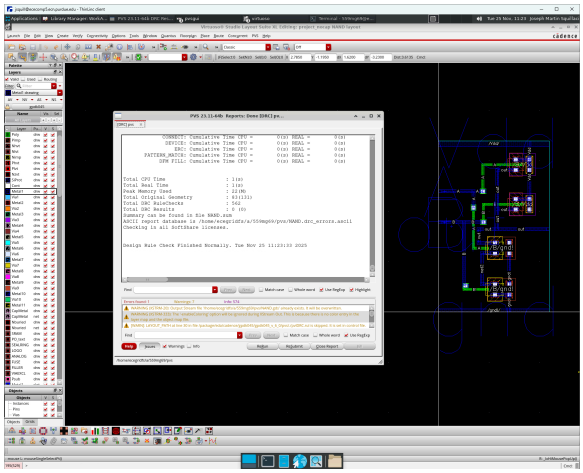
[45] 16-bit CSA difference of schematic vs AV Extracted curves
(left half is schematic and right half is AV Extracted)



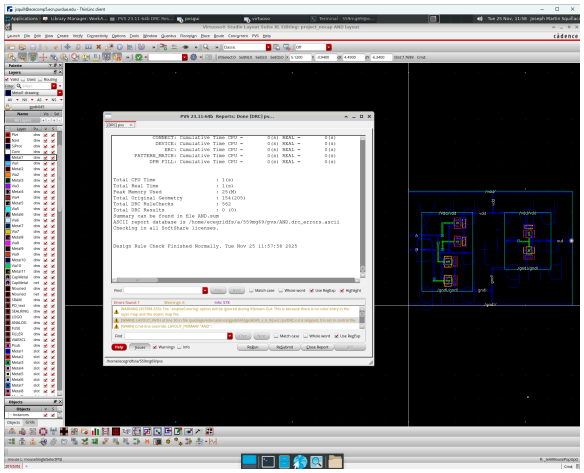
[49] Inverter DRC Check



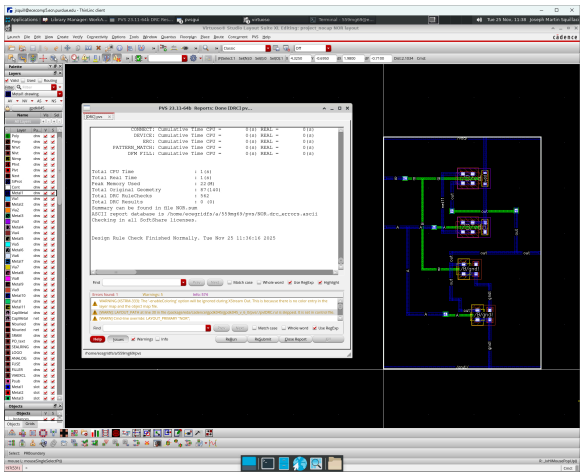
[52] OR DRC Check



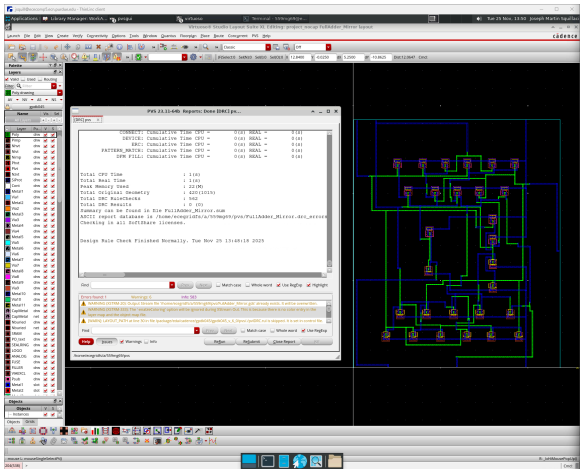
[50] NAND DRC Check



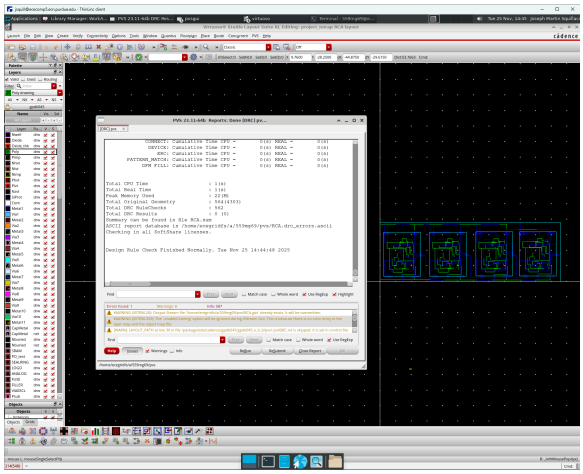
[53] AND DRC Check



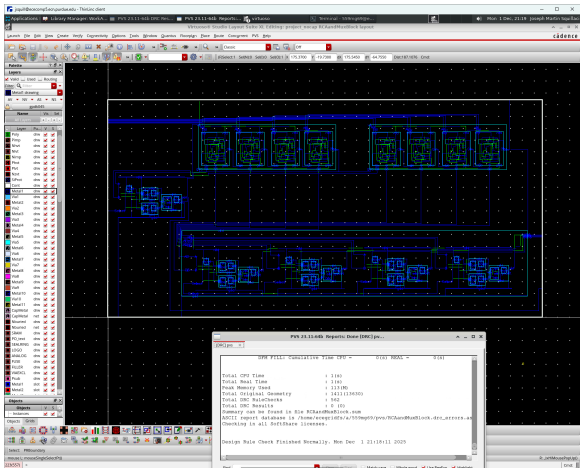
[51] NOR DRC Check



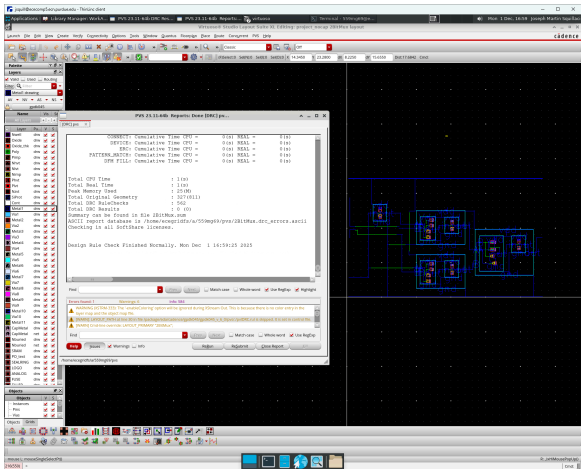
[54] Full Adder DRC Check



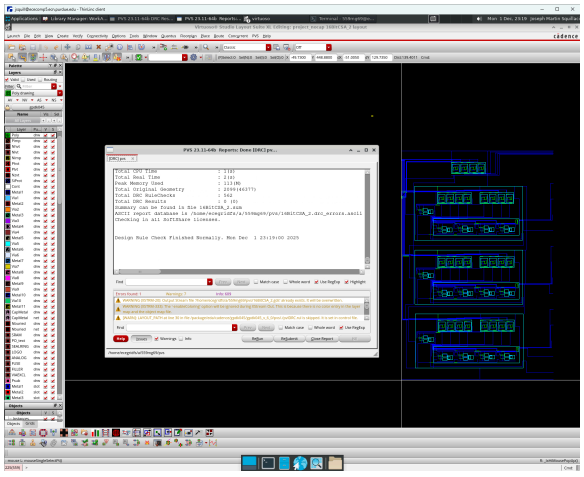
[55] RCA DRC Check



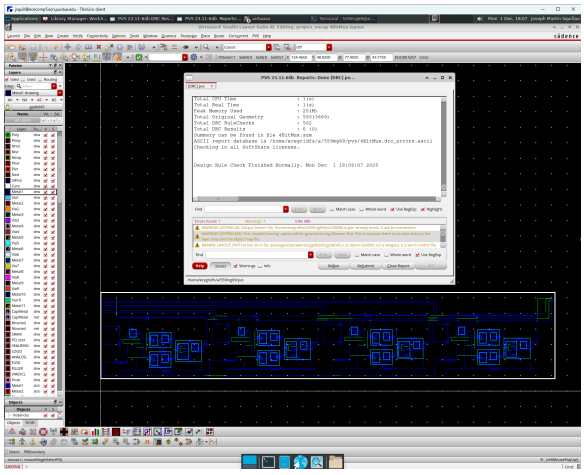
[58] RCA and Mux Block DRC Check



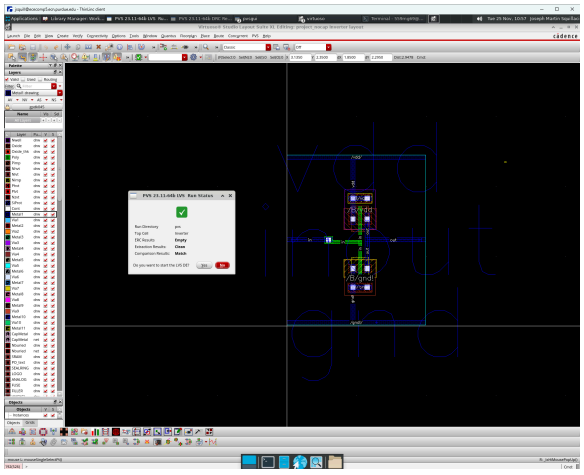
[56] 2 Bit Multiplexor DRC Check



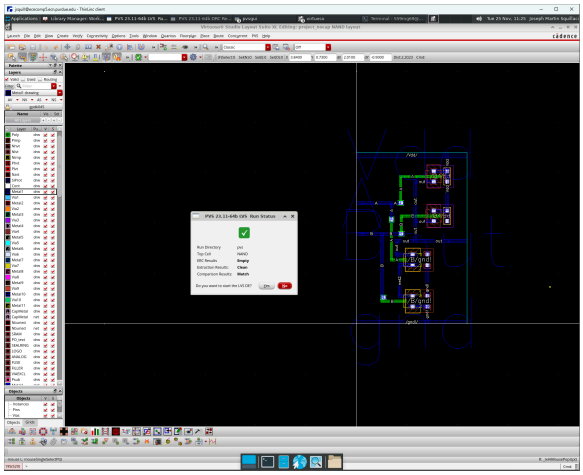
[59] 16-bit CSA DRC Check



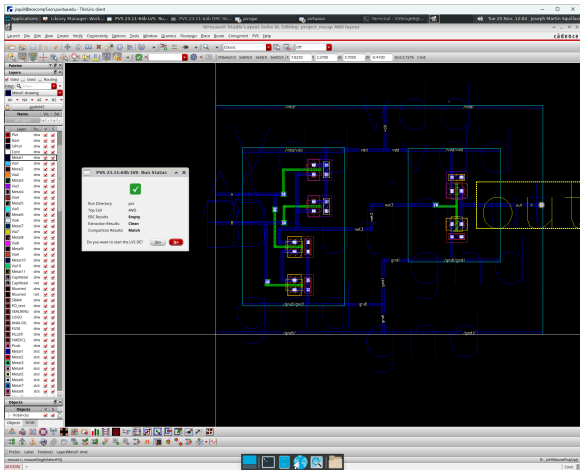
[57] 4 Bit Multiplexor DRC Check



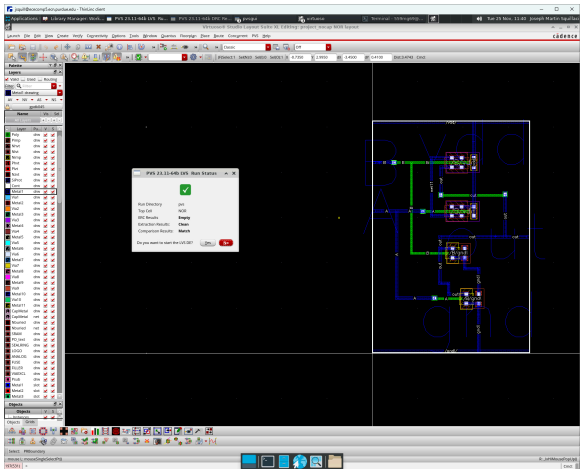
[60] Inverter LVS Check



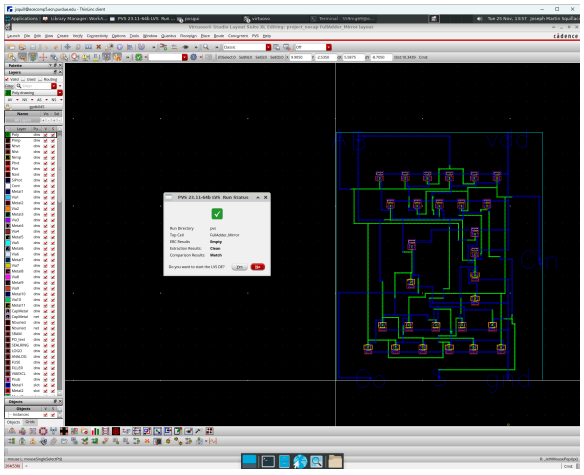
[61] NAND LVS Check



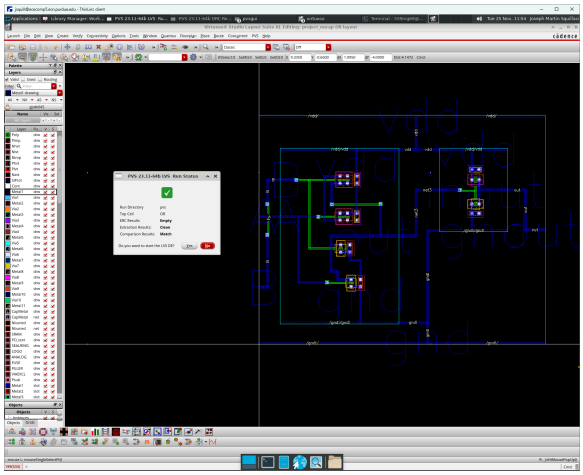
[64] AND LVS Check



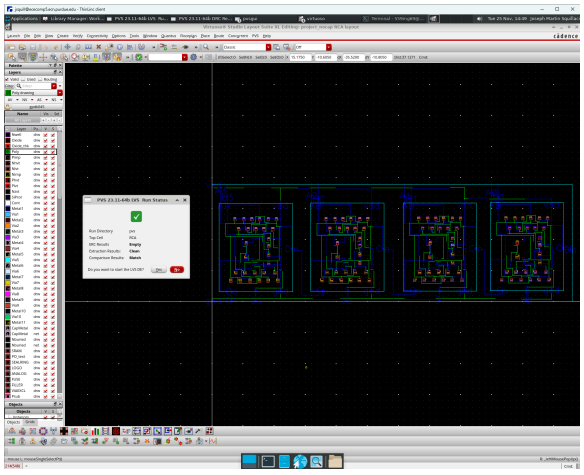
[62] NOR LVS Check



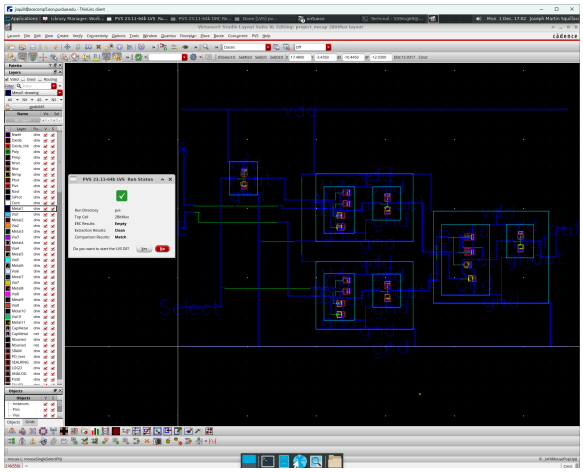
[65] Full Adder LVS Check



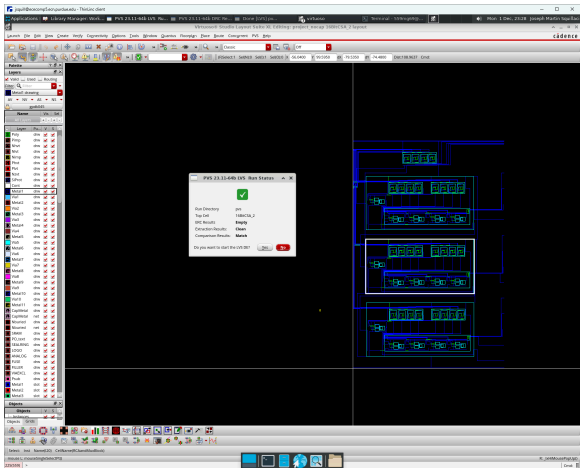
[63] OR LVS Check



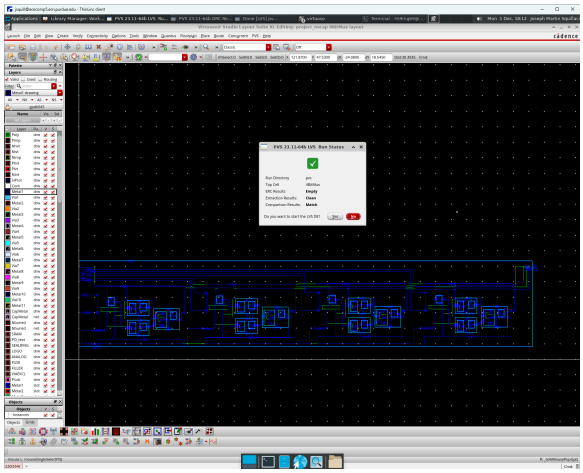
[66] RCA LVS Check



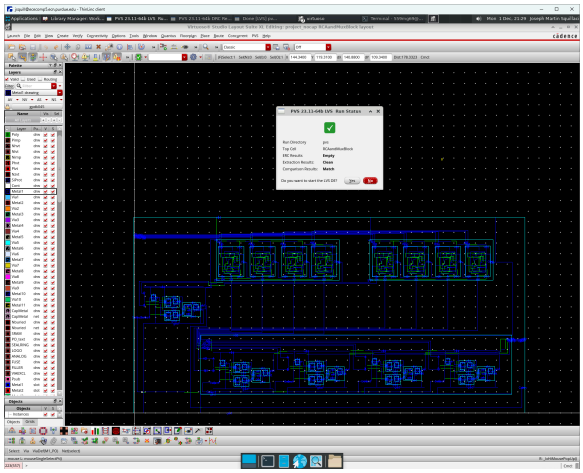
[67] 2 Bit Multiplexor LVS Check



[70] 16-bit CSA LVS Check



[68] 4 Bit Multiplexor LVS Check



[69] RCA and Mux Block LVS Check