ECE 467

Final Project Report
4-bit ALU Design

Fall 2013

Kai Zhao
Aswin Gonzalez
Sepideh Roghanchi
Soroush Khaleghi

Part 1) Final ALU Design:

There are 6 different functions implemented in this ALU:

1) 4 bit Addition (op. code 000)
2) 2's Complement of A (op. code 001)
3) 4 bit Add-traction (op. code 010)
4) 4 input NAND operation using static NAND gate (op. code 011)
5) 4 input NOR operation using dynamic NOR gate (op. code 100)
6) 1's Complement of B (op. code 101)

Figure 1 shows the final schematic of the ALU. There are 3 additional functionalities that we have used to design the final ALU:

- 3to6 Decoder: This block will select one of the 6 main units based on the operation code.
- 4x2-bit AND (AKA 5-bit AND in our project files): There are six 4x2-bit AND blocks, one for each main function. It has 5-bit input, consisting of four 2-bit AND gates inside. The 4 outputs of each unit are connected to 4 inputs of the 4 AND gates. The last input comes from the decoder an is connected to every AND gate to select the signal of the ALU. If one unit is selected, then the output of the 5-bit AND unit matches the output of the selected unit. The outputs of other units the 5-bit AND units will be silenced to zero.
- 6-1_or: There exist 4 instances of this gate, each for one of the final outputs. There are 6 inputs to each OR gate, coming from the six 5-bit AND blocks. For example, the (least significant bit) LSB of each 5-bit AND is connected to the inputs of the first 6-1_or gate to compute the final LSB.

Basically, we have used the above additional blocks to implement a 4-bit 6:1 MUX (needed for selecting the desired unit and propagating its outputs to primary outputs of the design) at a smaller hardware cost.

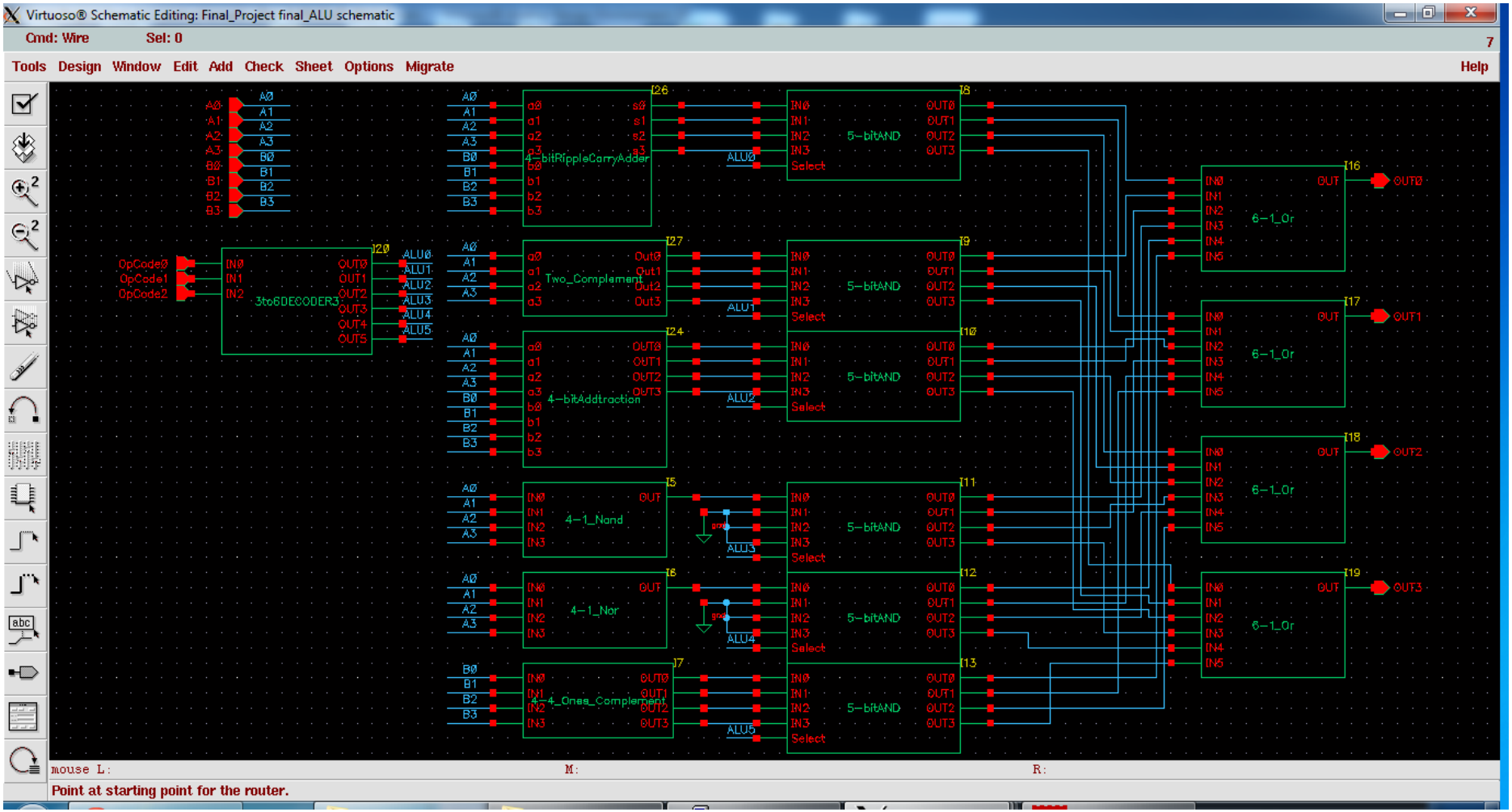

Figure 1: Final Schematic of the ALU

Figure 2 shows the final layout of the ALU.

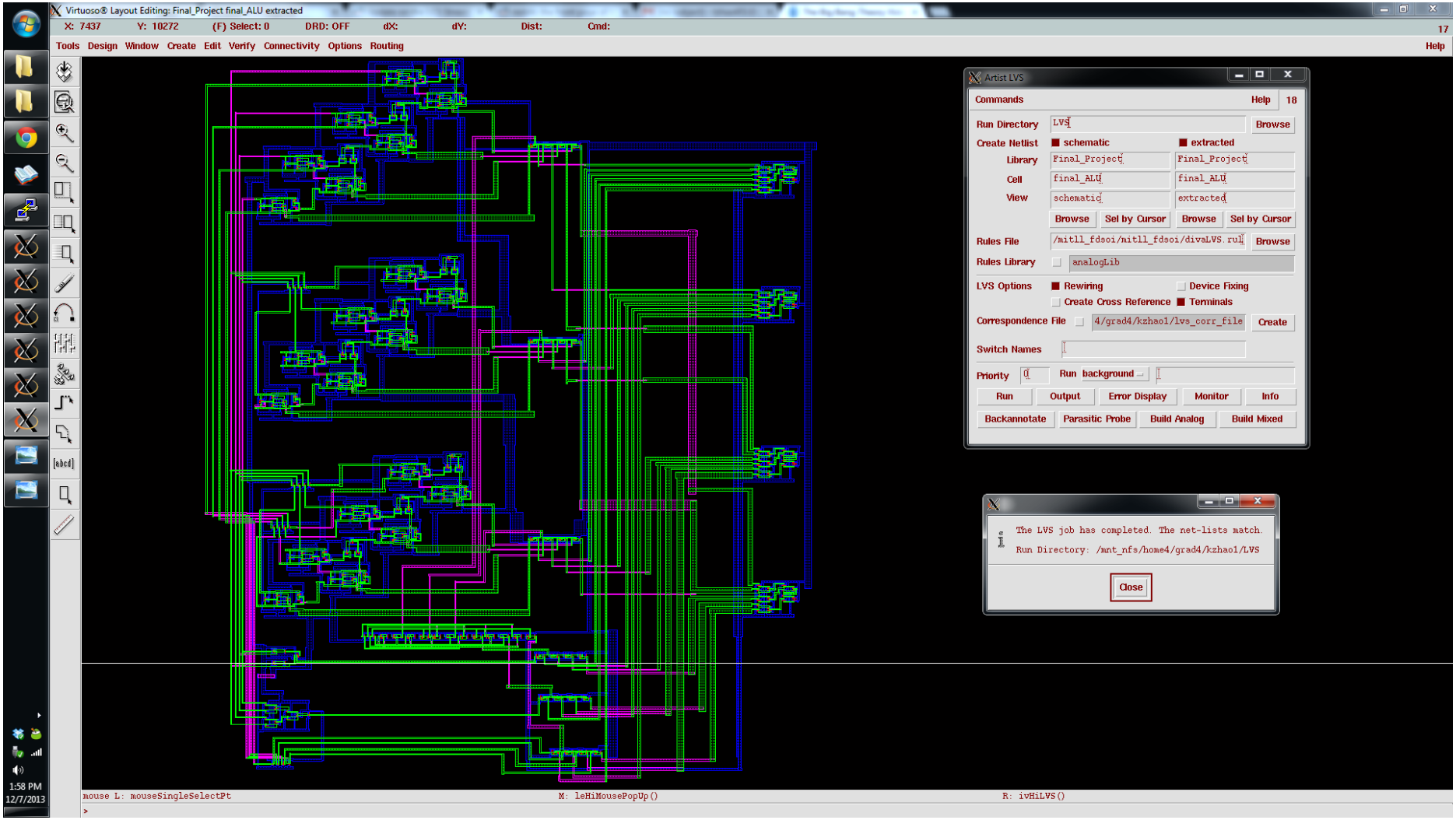

Figure 2: The final layout of the ALU

Now, we will show the simulation results performed on the final ALU. We have verified the outputs of the final ALU with 3 different input patterns. For each input pattern, the results have been verified for all 6 components.

For each test vector, we first show the corresponding truth table. Then, we will show the schematic of the setup and finally the simulations results.

Note: We have simulated every block with **all** possible input combinations separately as well. The results will be shown in the next section.

First Test Vector Truth Table:

A[3:0]	**B[3:0]**	**opCode**	**Output[3:0]**	**Operation**
0000	0101	000	**0101**	Addition
0000	0101	001	**0000**	2's Complement
0000	0101	010	**1001**	Addtraction
0000	0101	011	**0001**	NAND
0000	0101	100	**0001**	NOR
0000	0101	101	**1010**	1's Complement
0000	0101	110	**0000**	Not defined
0000	0101	111	**0000**	Not defined

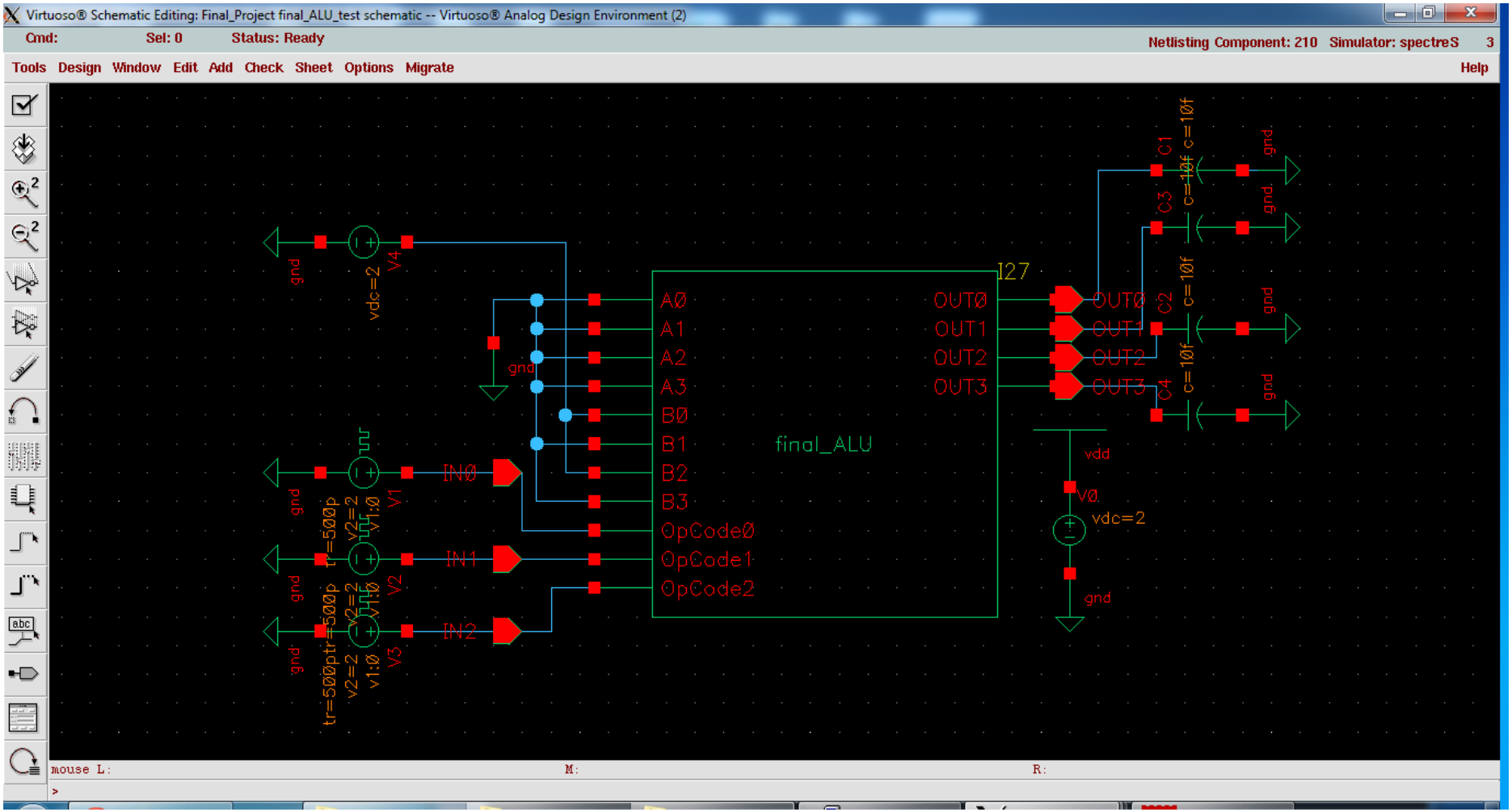

Figure 3: Applying the First TV to the ALU

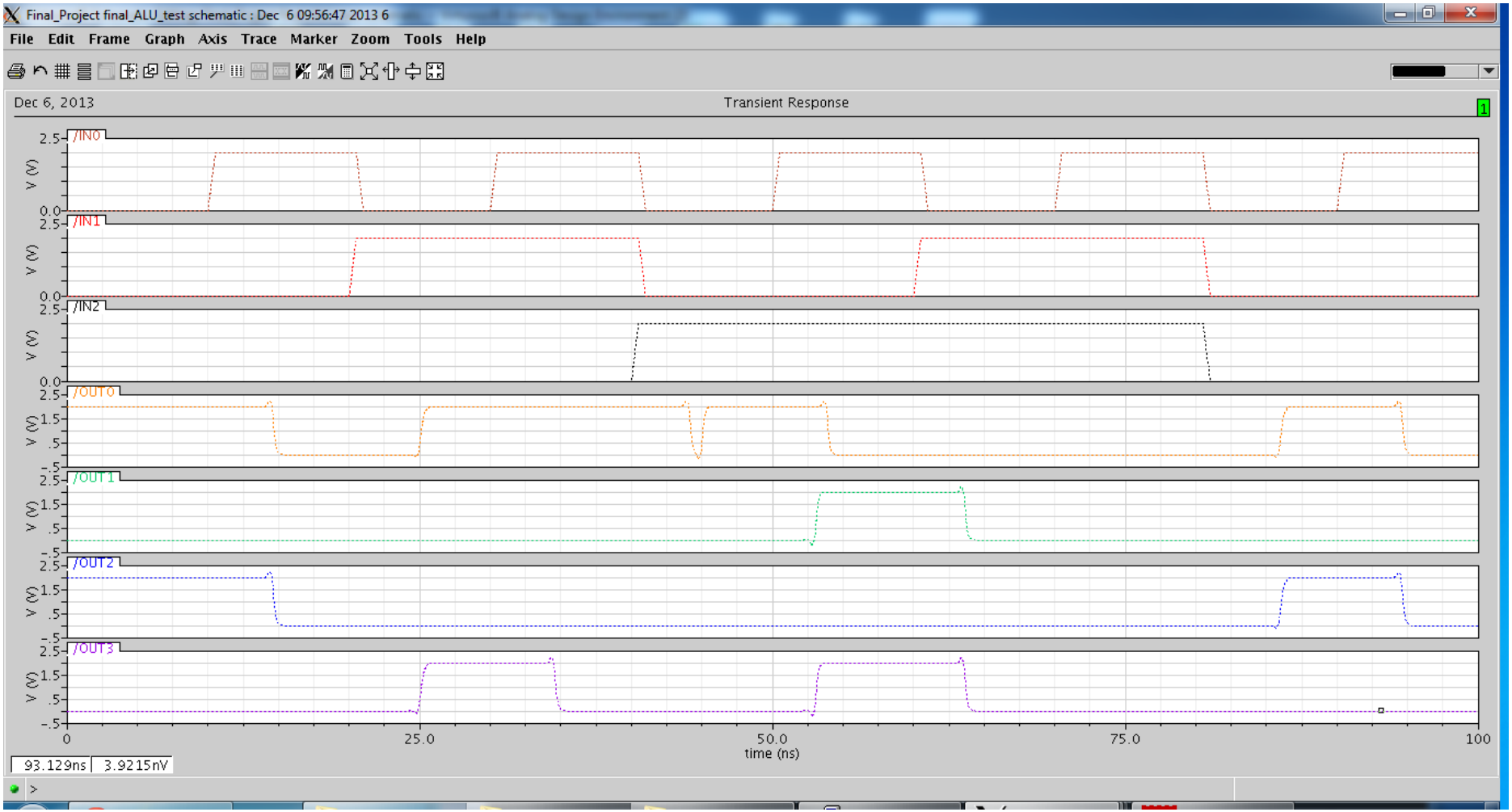

Figure 4. The **pre-layout** simulation results for the first TV

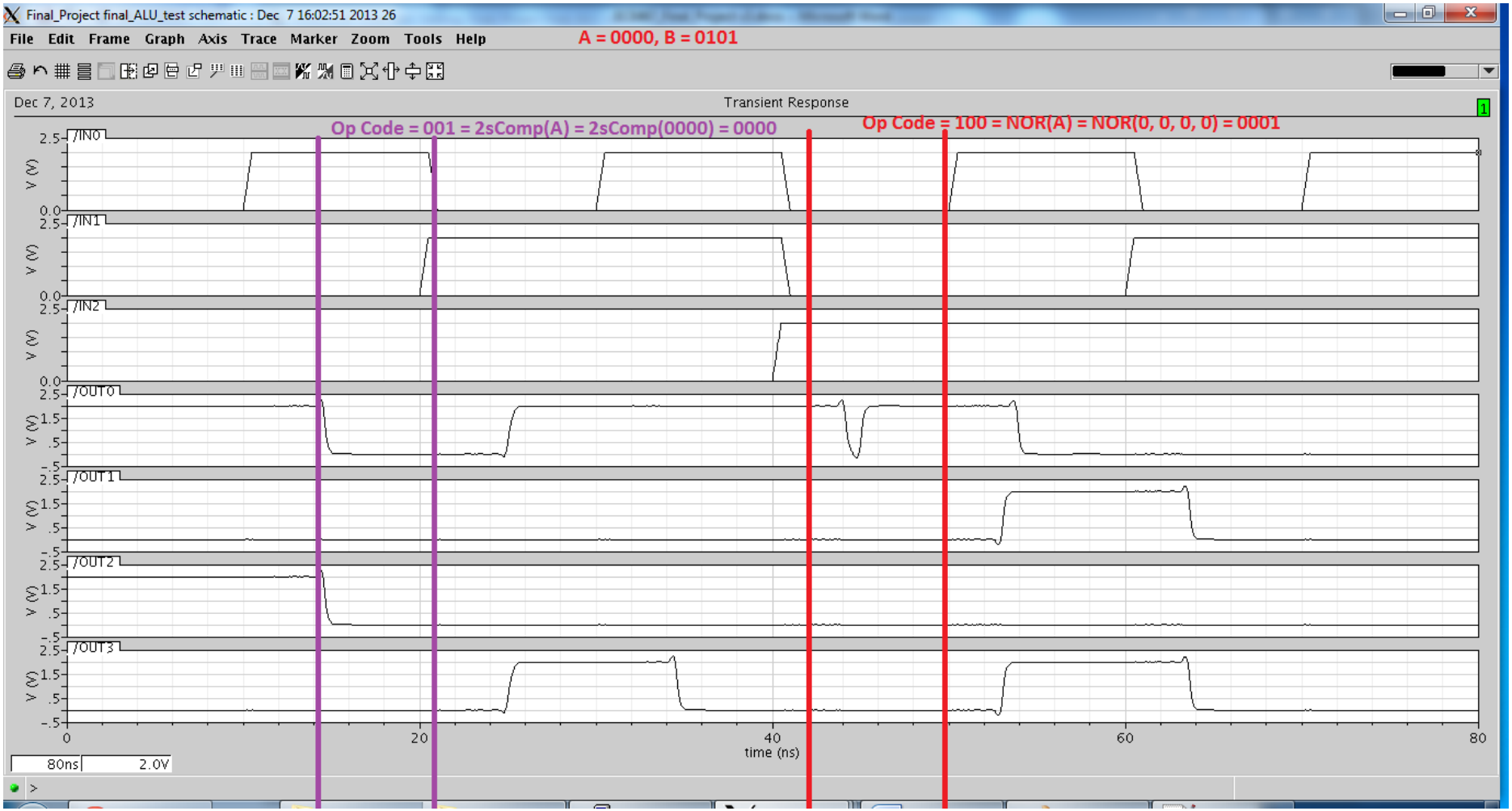

Figure 5: The **post-layout** simulation results for the first TV

In the above graph, IN2, IN1, and IN0 are the opcodes (from MSB to LSB). OUT3, OUT2, OUT1, OUT0 are representing the final outputs (from MSB to LSB). As it can be seen from some sample calculations on the figure, all the truth table results match the simulation results.

Second Test Vector Truth Table:

A[3:0]	B[3:0]	opCode	Output[3:0]	Operation
1111	0110	000	**0101**	Addition
1111	0110	001	**0001**	2's Complement
1111	0110	010	**1001**	Addtraction
1111	0110	011	**0000**	NAND
1111	0110	100	**0000**	NOR
1111	0110	101	**1001**	1's Complement
1111	0110	110	**0000**	Not defined
1111	0110	111	**0000**	Not defined

We fix A3-0 to be all 1's to check the correctness of the NAND gate, when the output is 0. We also fixed it to be 0 to check that the NOR gate correctly outputs 1.

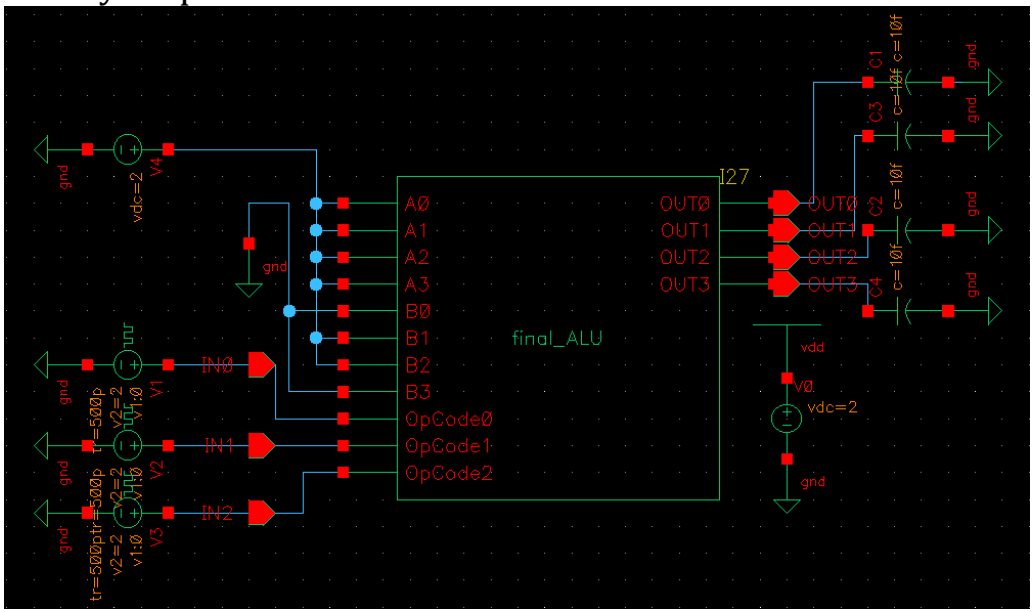

Figure 6: Applying the 2nd TV to ALU

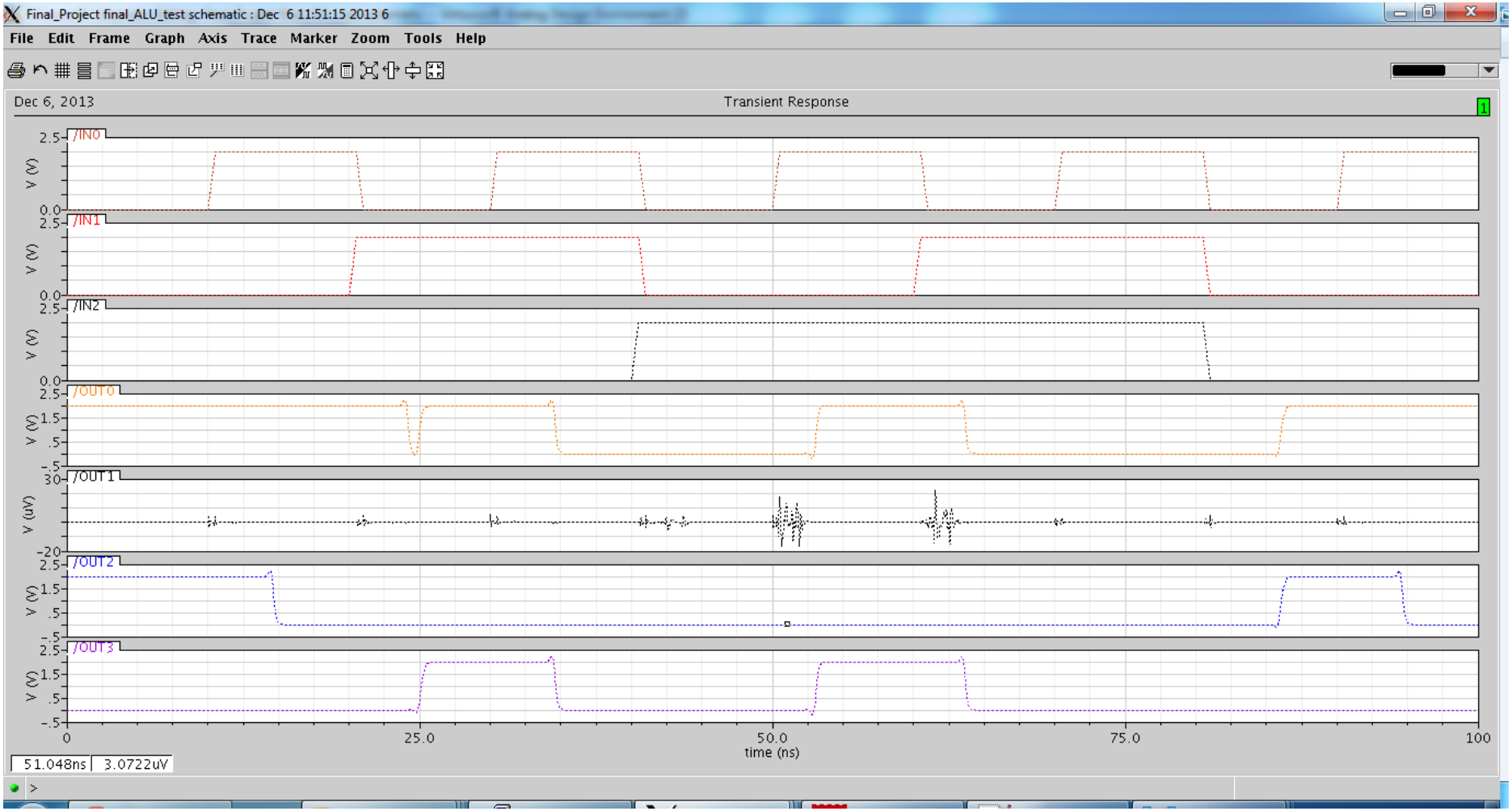

Figure 7: The **pre-layout** simulation results for the 2nd TV

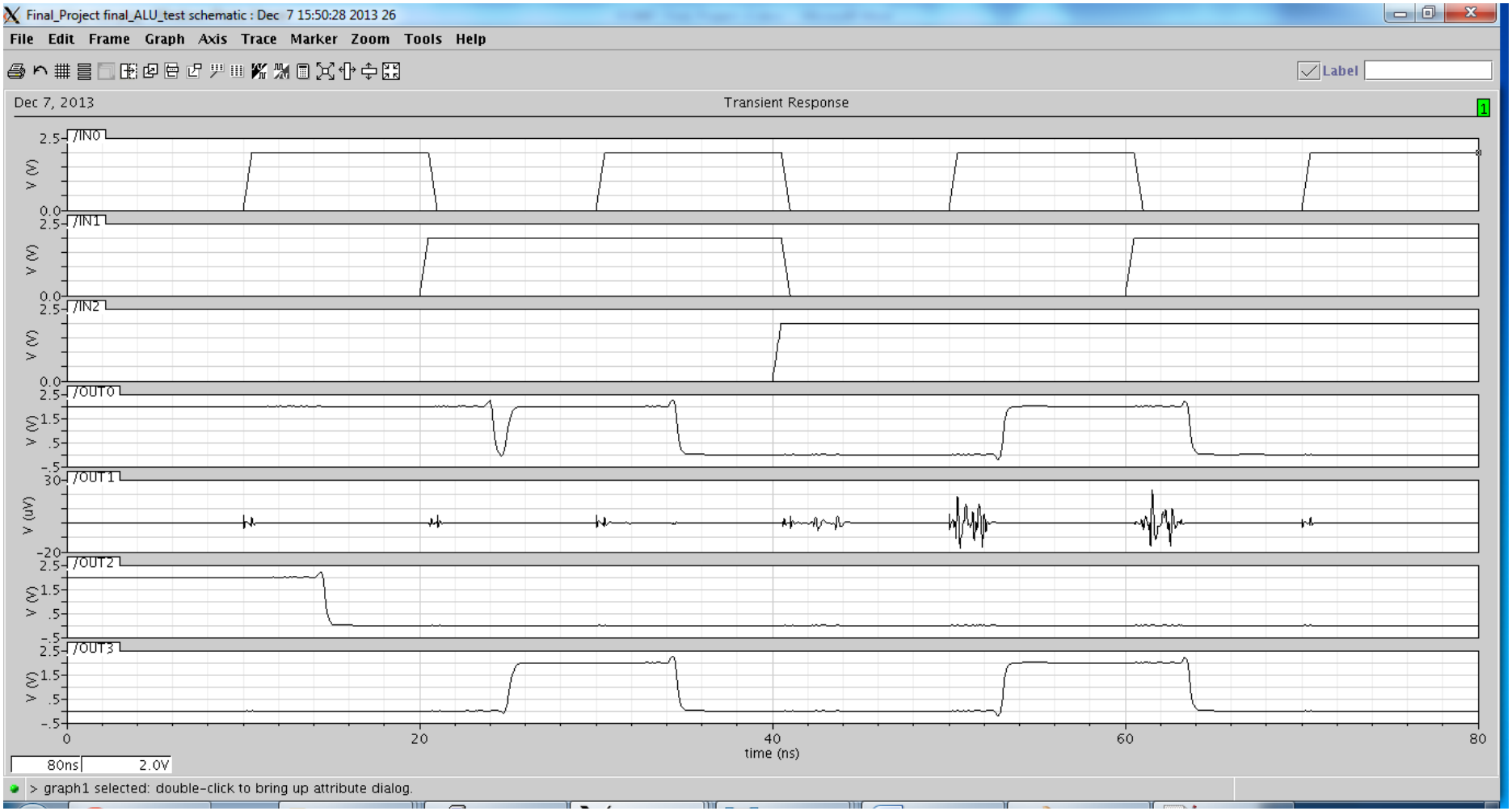

Figure 8: The **post-layout** simulation results for the 2nd TV

Third Test Vector Truth Table:

A[3:0]	B[3:0]	opCode	Output[3:0]	Operation
0010	1001	000	**1011**	Addition
0010	1001	001	**0001**	2's Complement
0010	1001	010	**0011**	Addtraction
0010	1001	011	**0001**	NAND
0010	1001	100	**0000**	NOR
0010	1001	101	**0101**	1's Complement
0010	1001	110	**0000**	Not defined
0010	1001	111	**0000**	Not defined

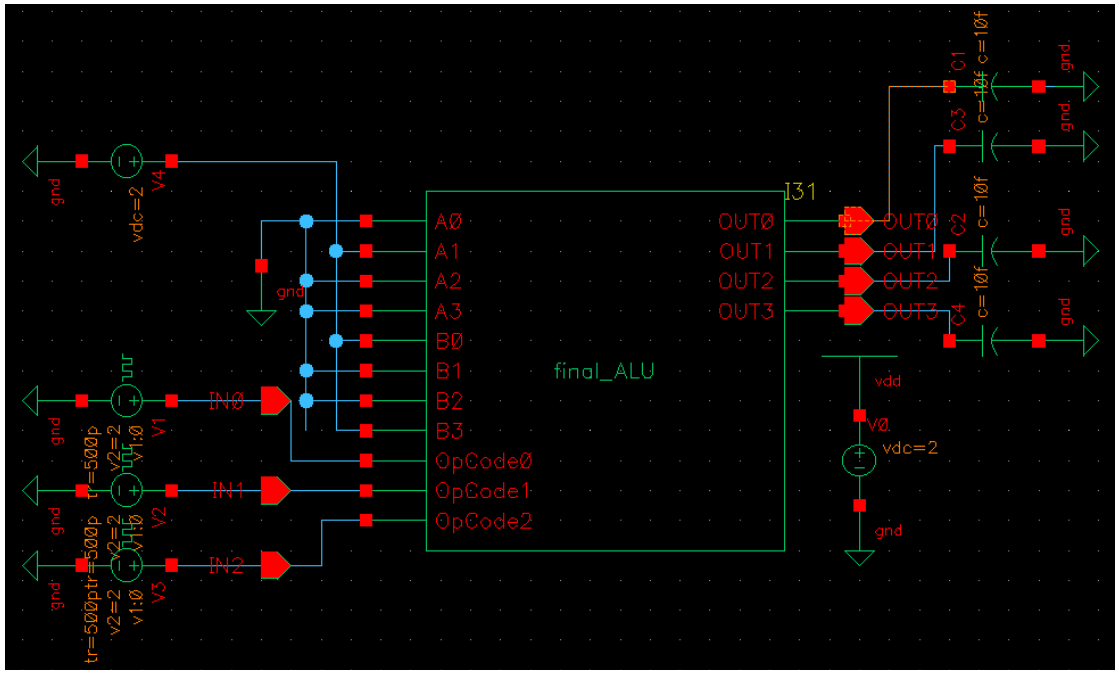

Figure 8: Applying the 3rd TV to ALU

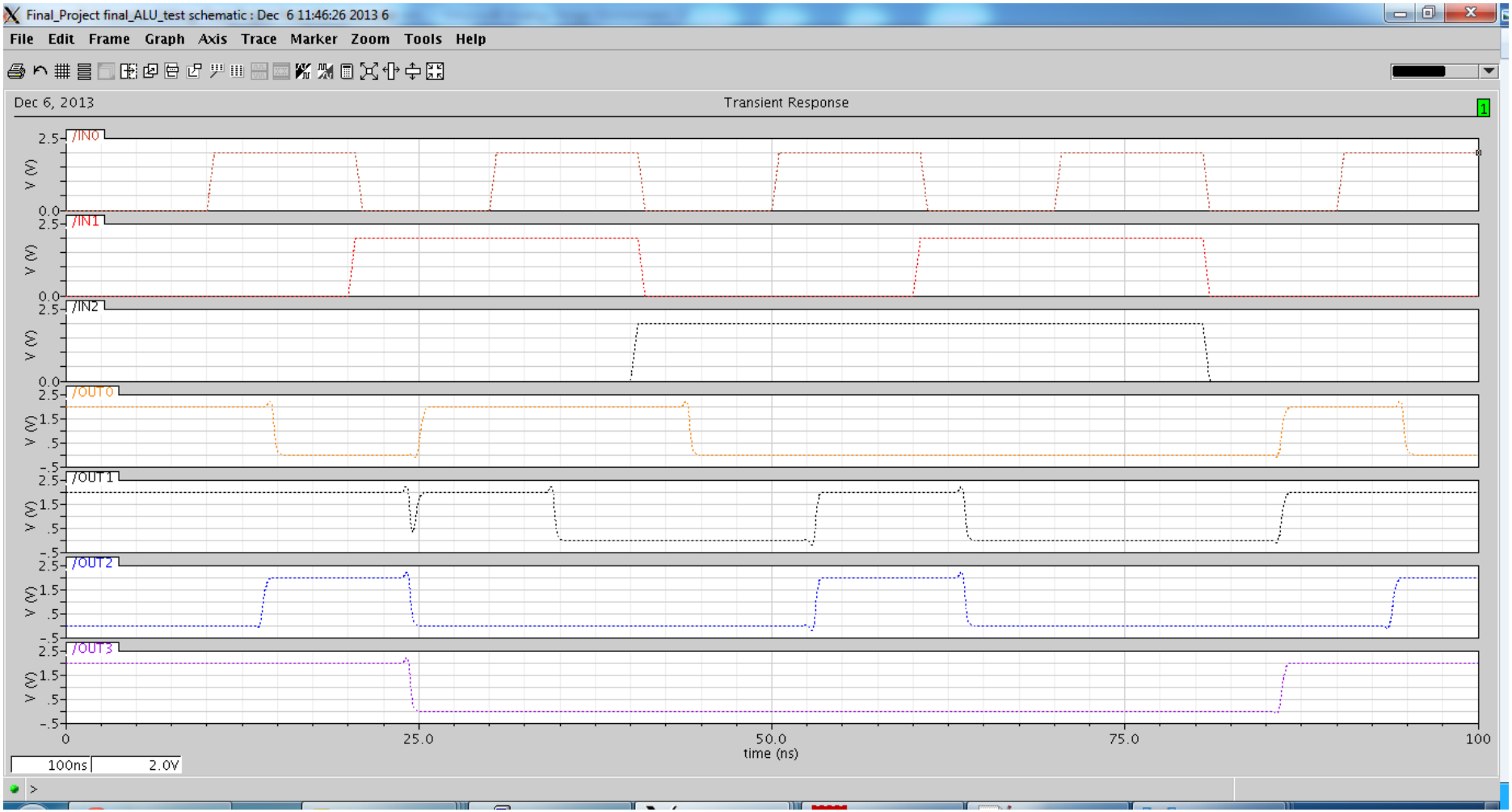

Figure 9: The **pre-layout** simulation results for the 3rd TV

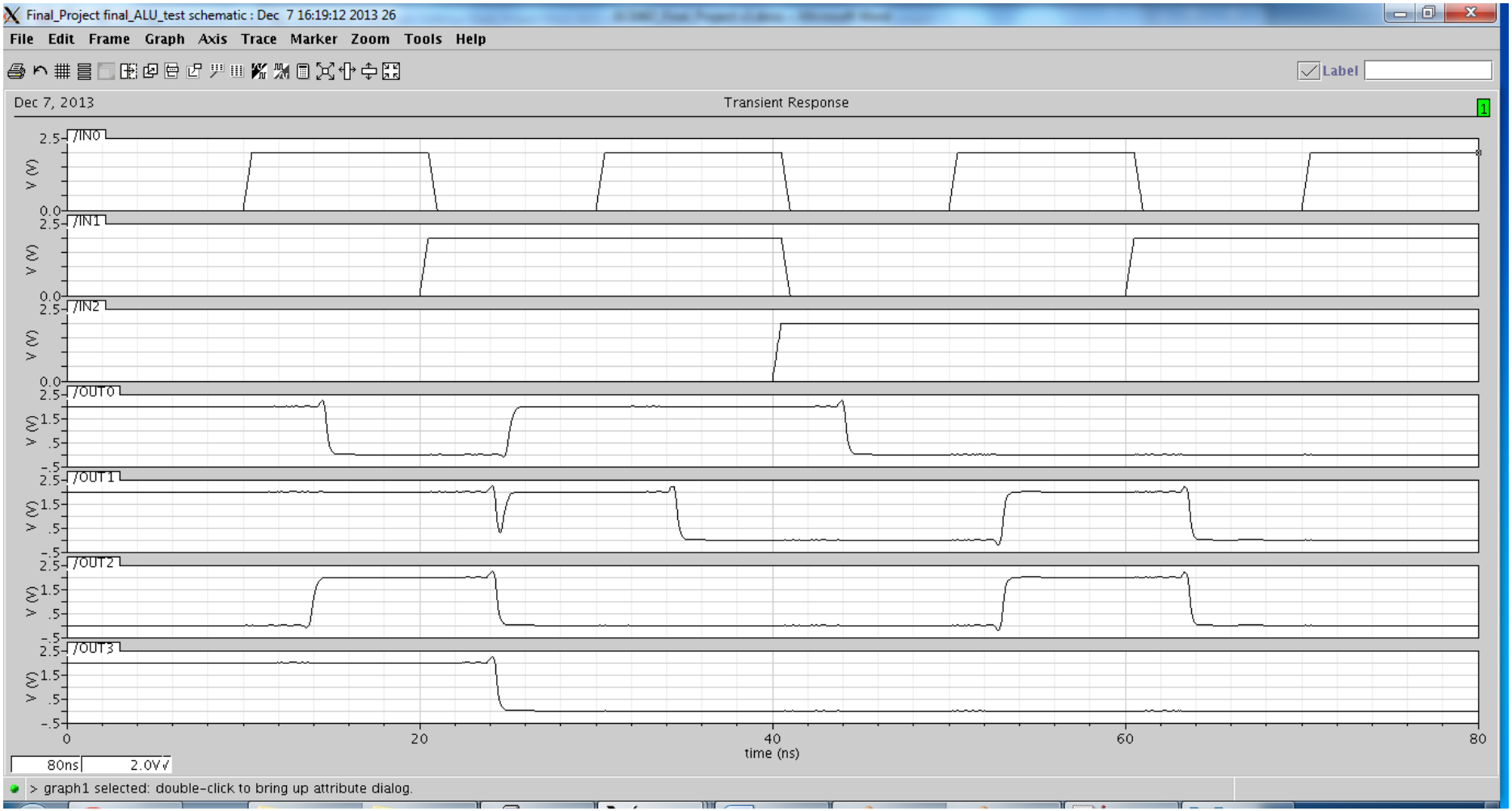

Figure 10: The **post-layout** simulation results for the 3rd TV

Now that we have verified the correctness of the ALU, we will show the schematic, test, and layout of every component in the final design. We will test every component with all possible op code.

Part 2) Main Units Design:

2.1) 4-bit NAND:

Next figure shows the schematic and layout views of the 4-bit NAND gate. This gate is built using three 2-bit NAND gates and two inverters, provided by the TA.

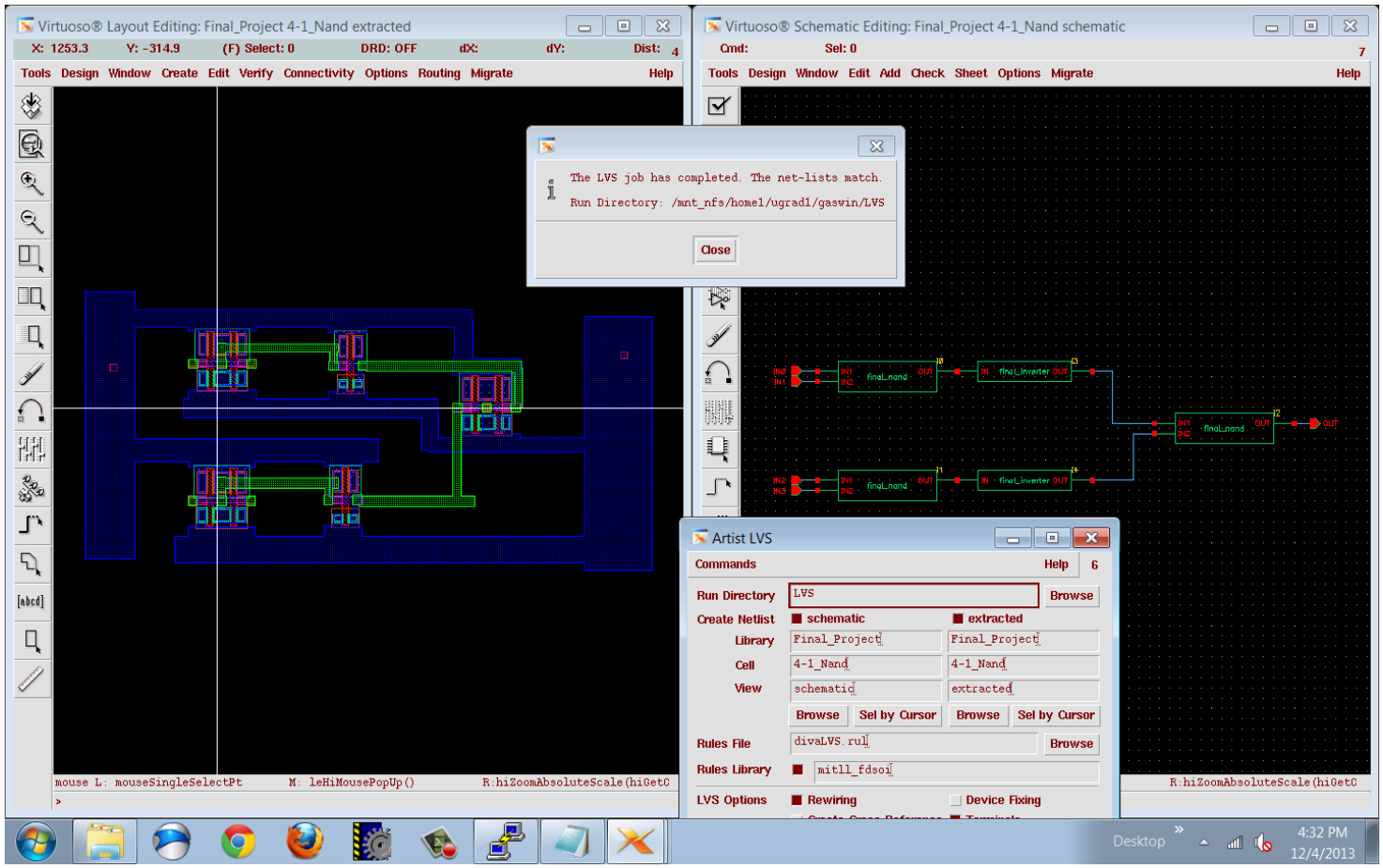

Figure 11: Schematic and Extracted views of 4-bit NAND gate

Next figure shows the simulation results of the NAND gate, for all 16 possible input patterns. The output is 0 if and only if all inputs are 1.

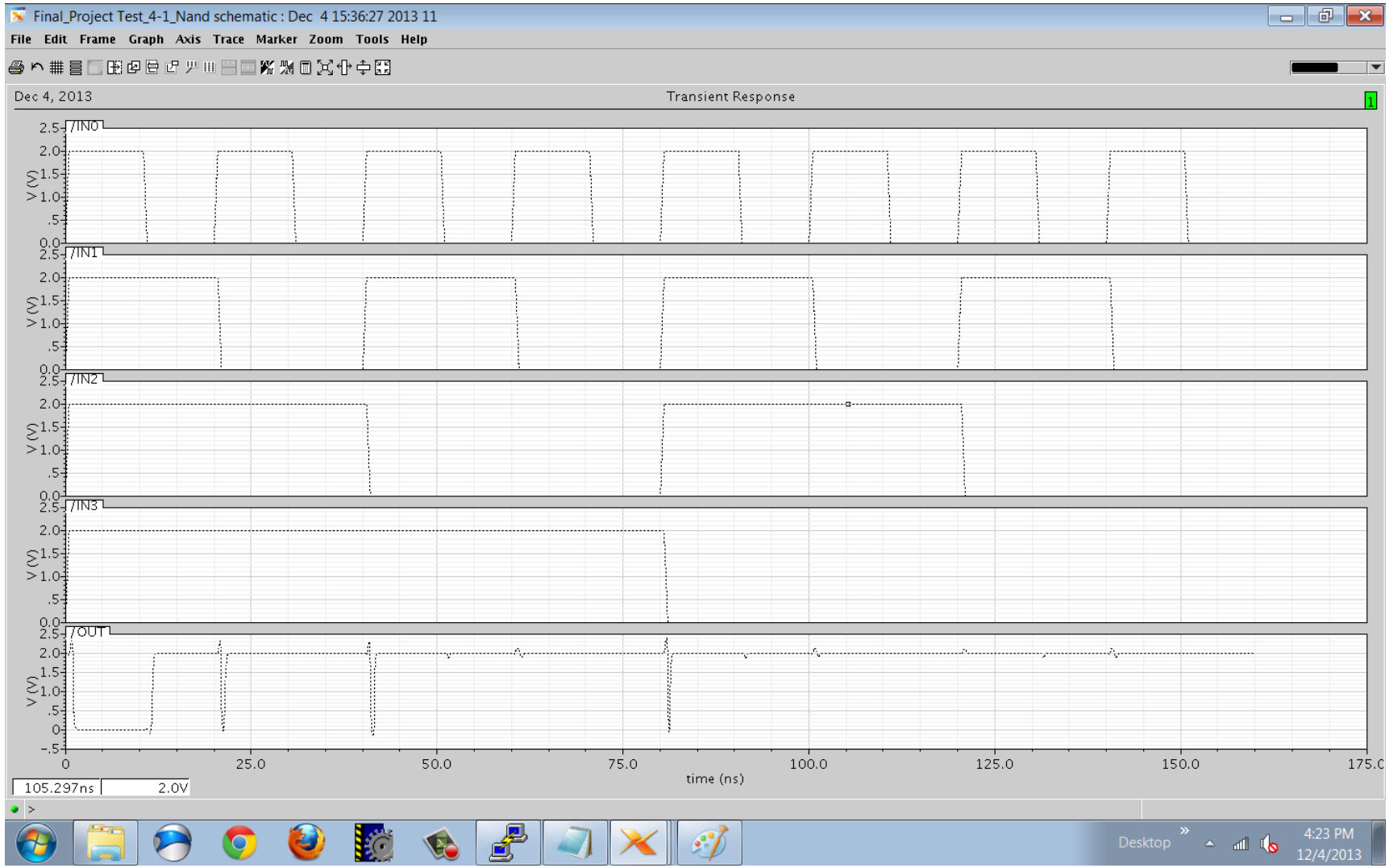

Figure 12: Simulation result for the NAND gate

2.2) 4-bit NOR:

Next figure shows the schematic and layout views of the 4-bit NOR gate. This gate is built using NAND gates and inverters. The NOR gate is implemented using static CMOS, because the dynamic gate are good for sequential circuits.

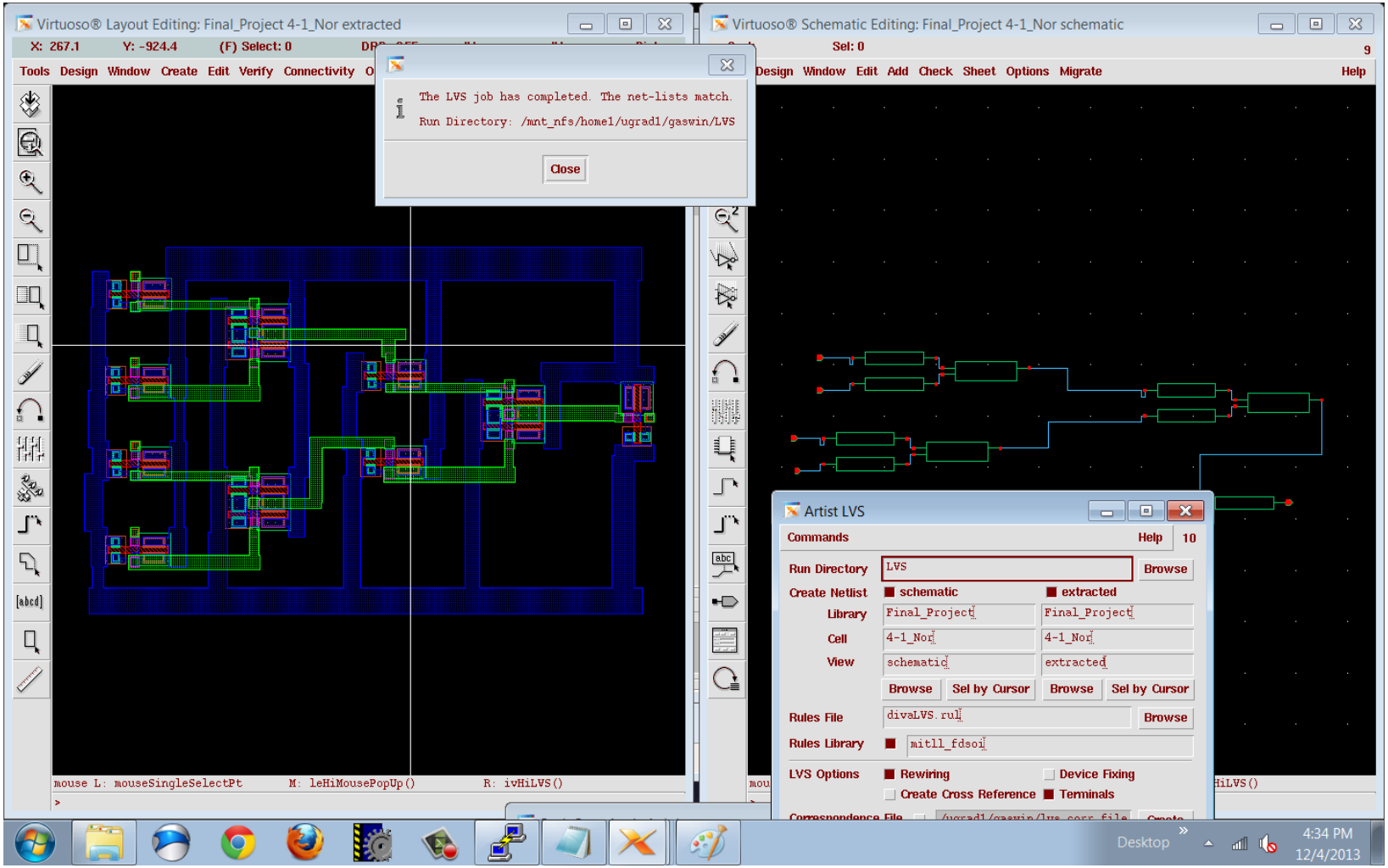

Figure 13: Schematic and Extracted views of the NOR gate

Next figure shows the simulation results of the NOR gate, for all 16 possible input patterns. The output is 1 if and only if all inputs are 0.

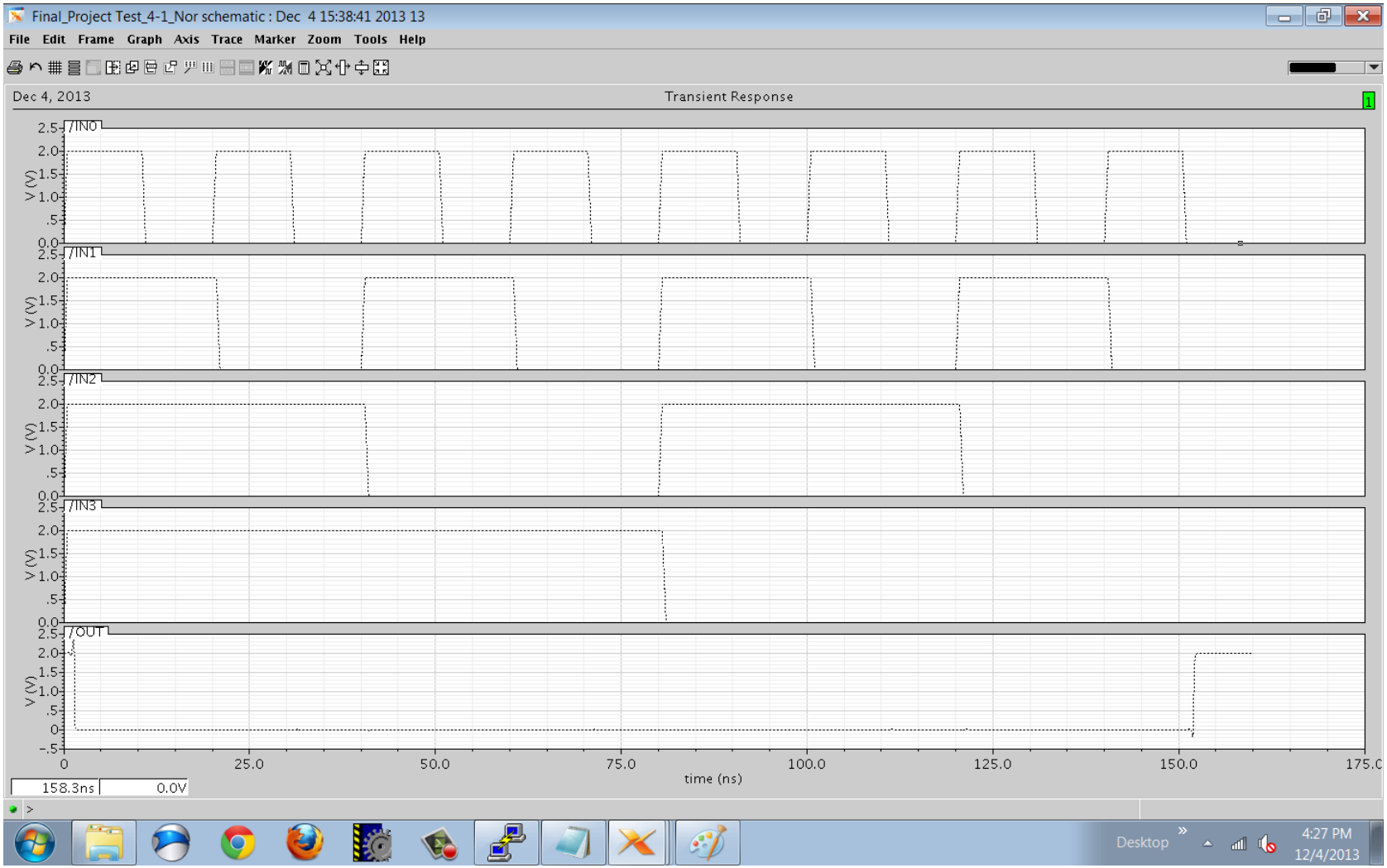

Figure 14: Simulation Results for the NOR gate

2.3) 1's Complement:

Basically, the 1's complement block consists of 4 inverters. Next figure shows the schematic and the layout views of this block.

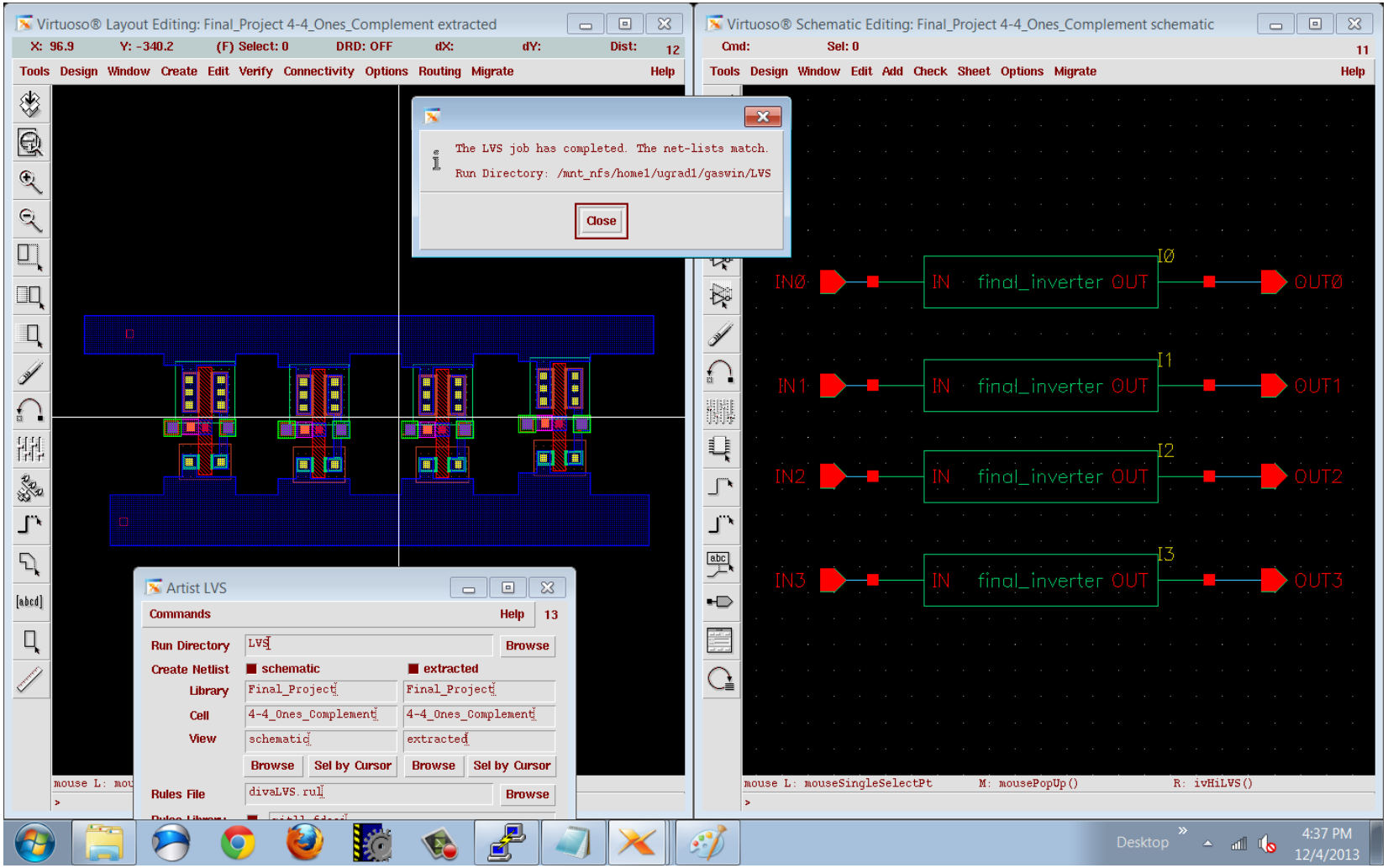

Figure 15: Schematic and the Extracted views of the 1's Complement block

The figure below shows the simulation results of the 1's complement block. Basically, each input must be inverted to get the corresponding output.

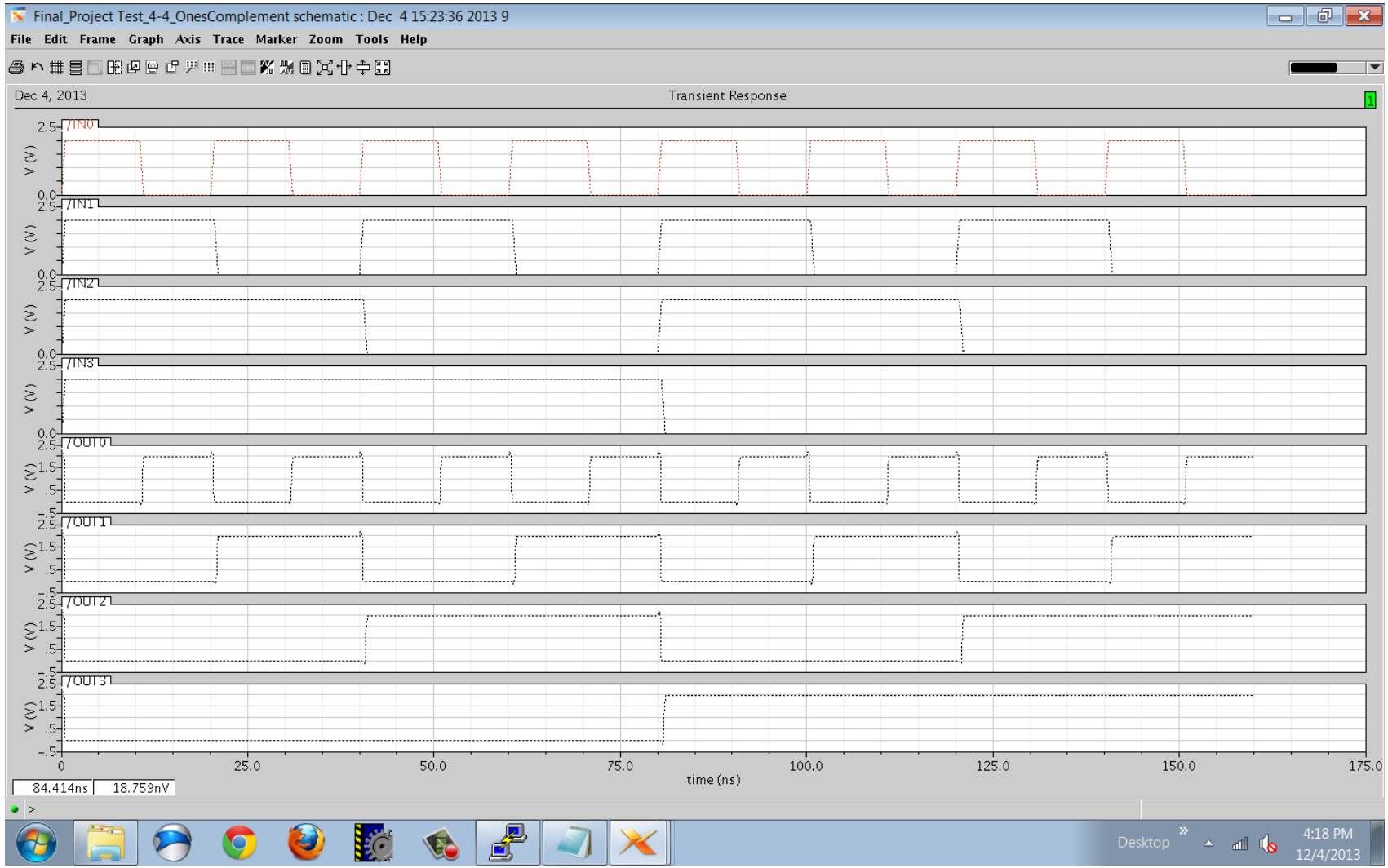

Figure 16: Simulation results for the 1's Complement block

2.4) 4-bit Adder

We have used the 4-bit Ripple Carry Adder (RCA) in this project. The RCA is built by cascading 3 Full adders and 1 half adder. The Full adder itself is built by 2 half adder and one OR gate. The Half adder block is built by an AND gate and an XOR gate. We will show the schematic of each of these blocks.

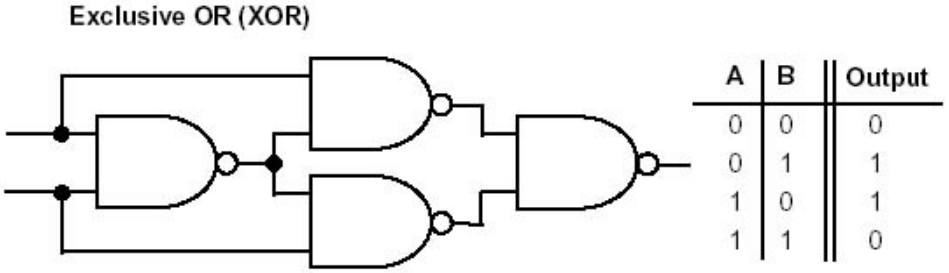

Figure 19: XOR gate implementation using NAND gates

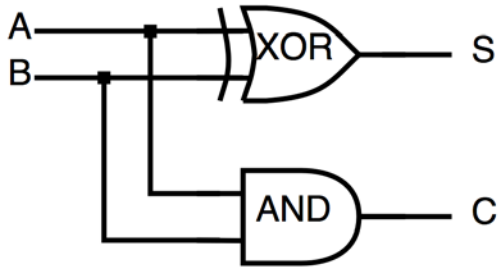

Figure 17: Half adder

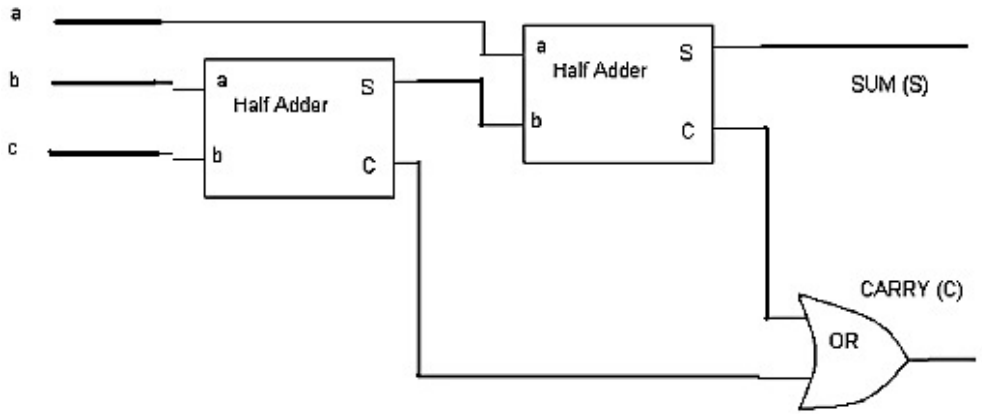

Figure 18: Full Adder using Half Adder

Next 3 figures show the layout of the XOR gate, Half Adder, and Full Adder.

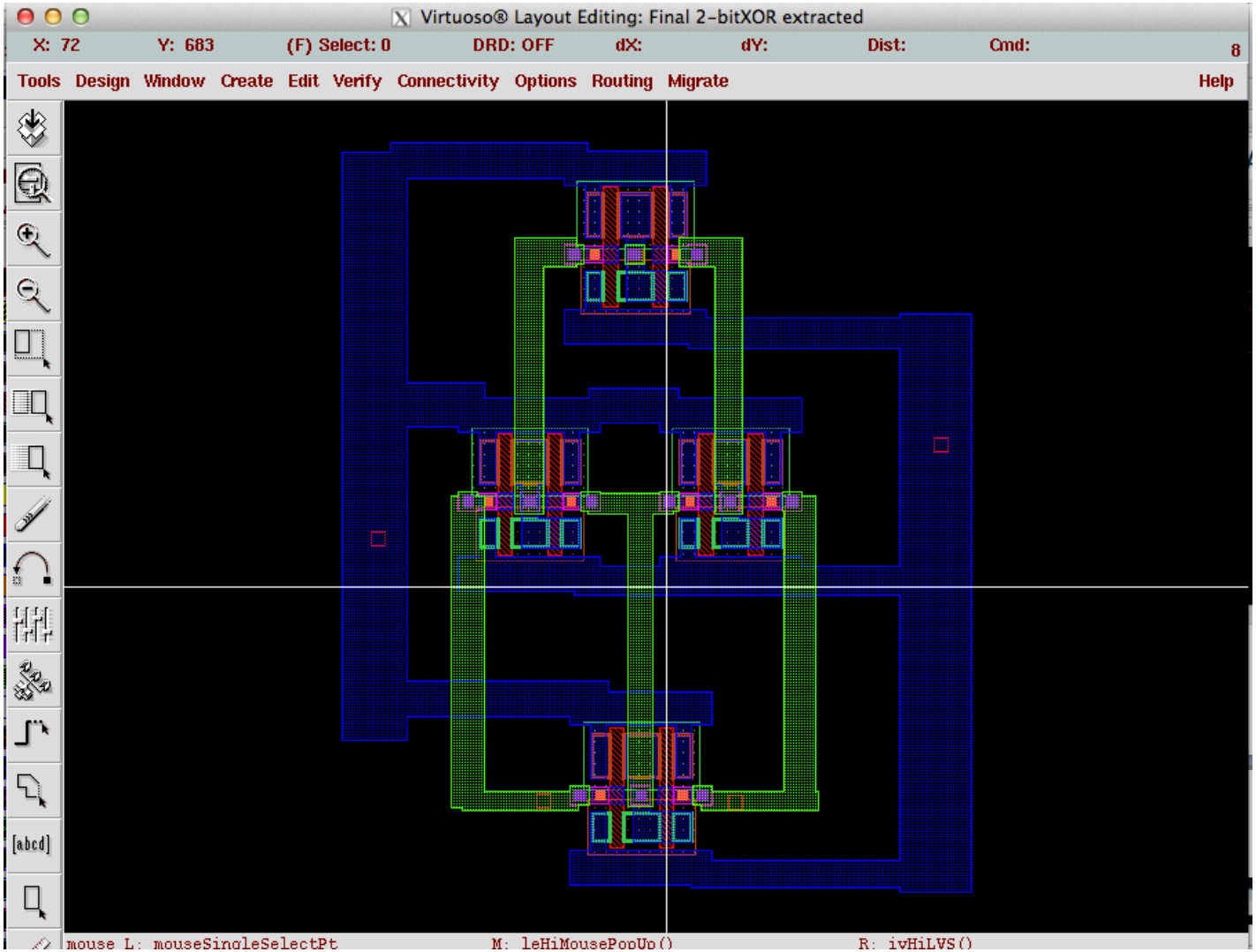

Figure 20: Schematic view of the XOR gate

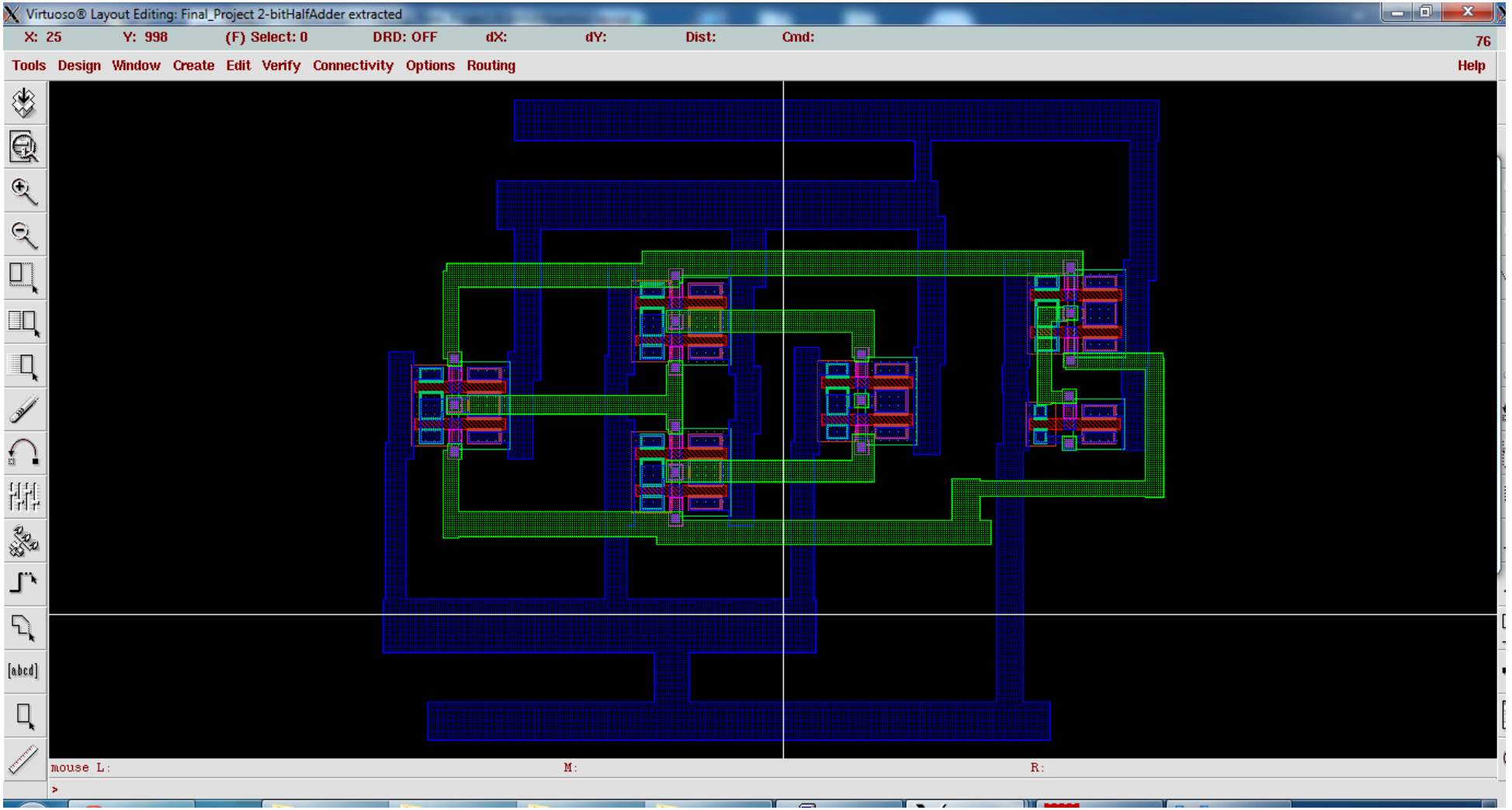

Figure 21: Layout view of the Half Adder

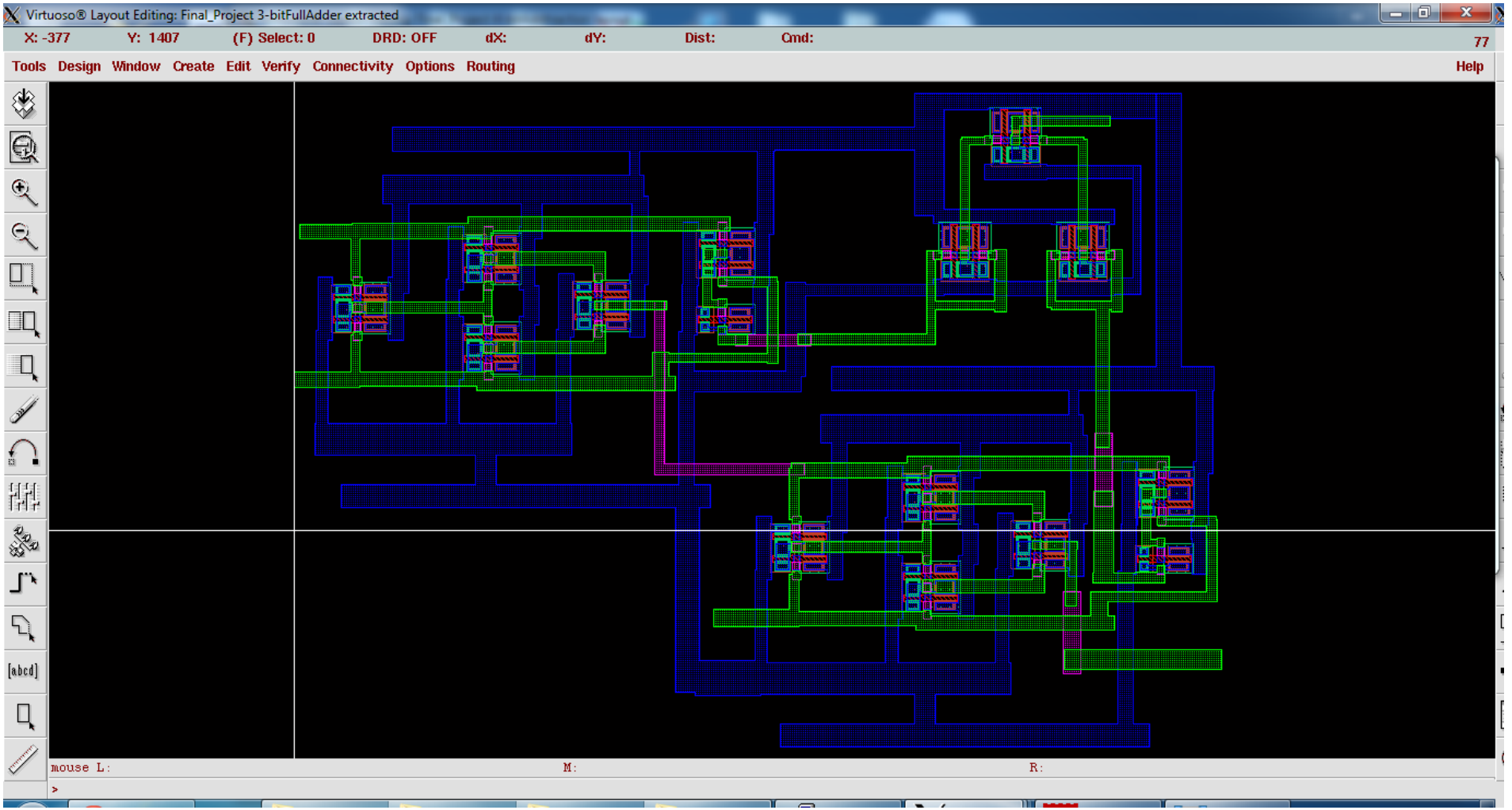

Figure 22: Layout view of the Full Adder

Next figure shows the schematic view of the 4-bit RCA.

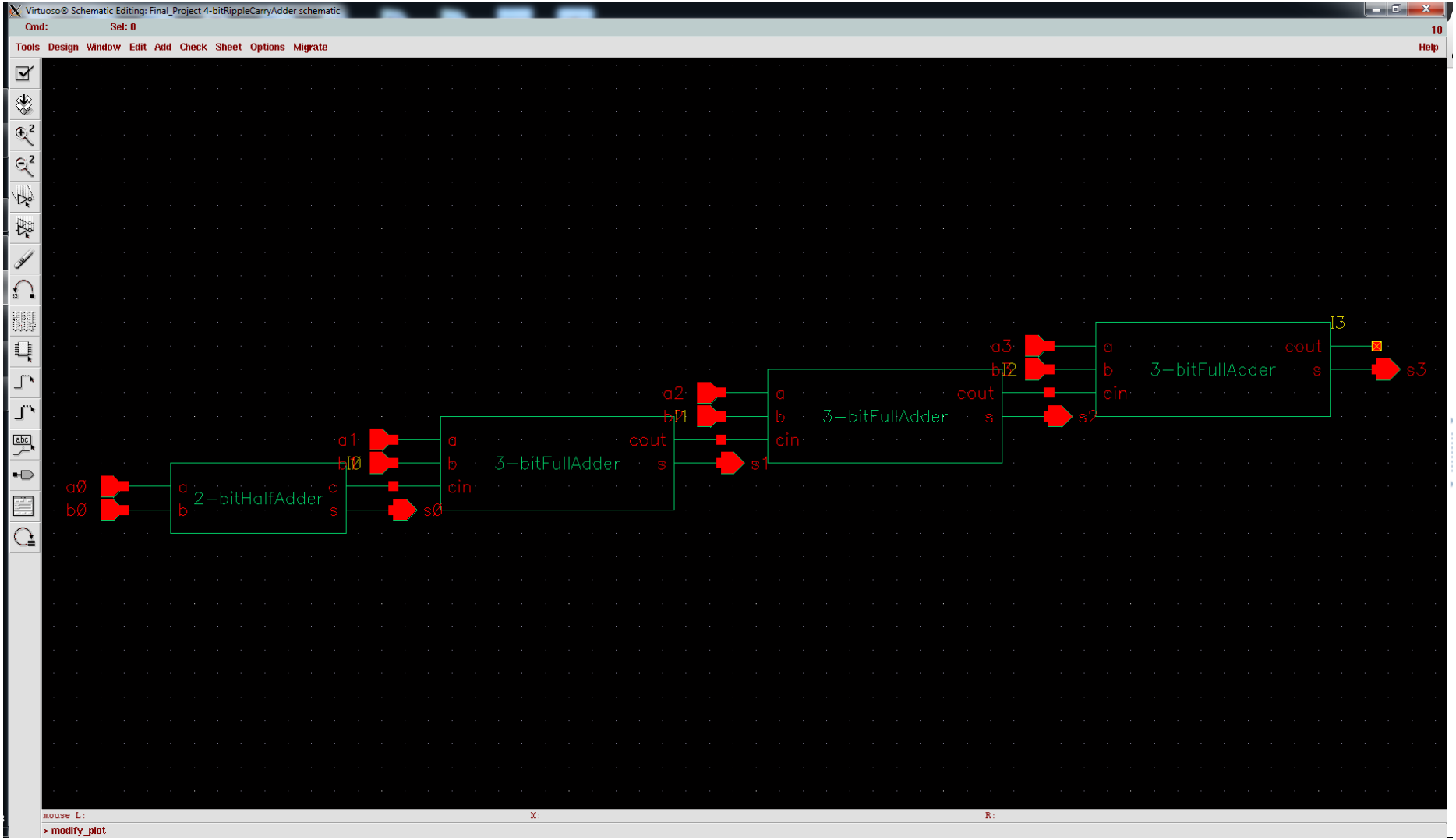

Figure 21: Schematic view of the 4-bit RCA

Next figure shows the simulation results for the 4-bit RCA.

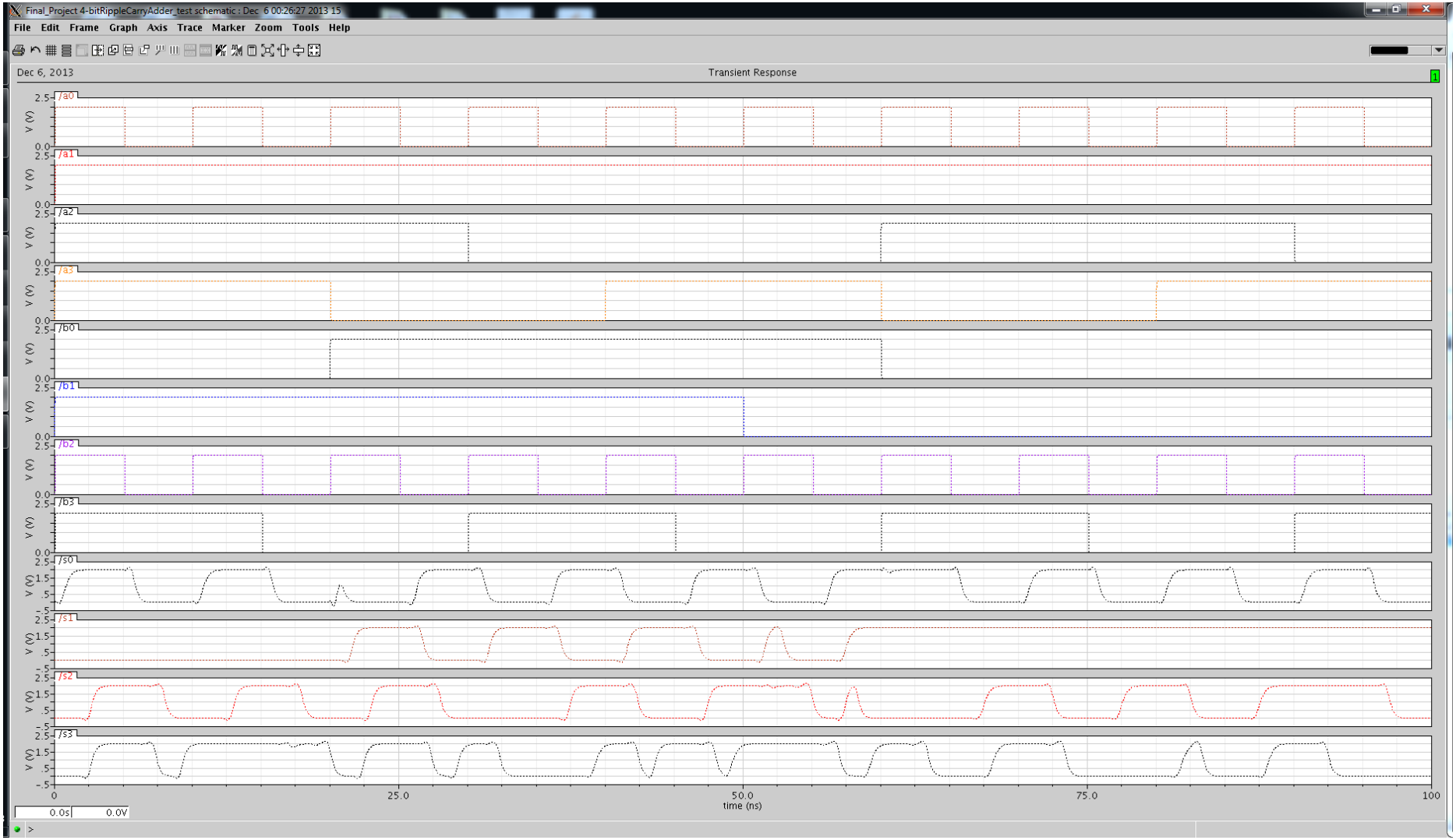

Figure 2223: Simulation result for the 4-bit RCA

Next figure shows the layout of the 4-bit RCA.

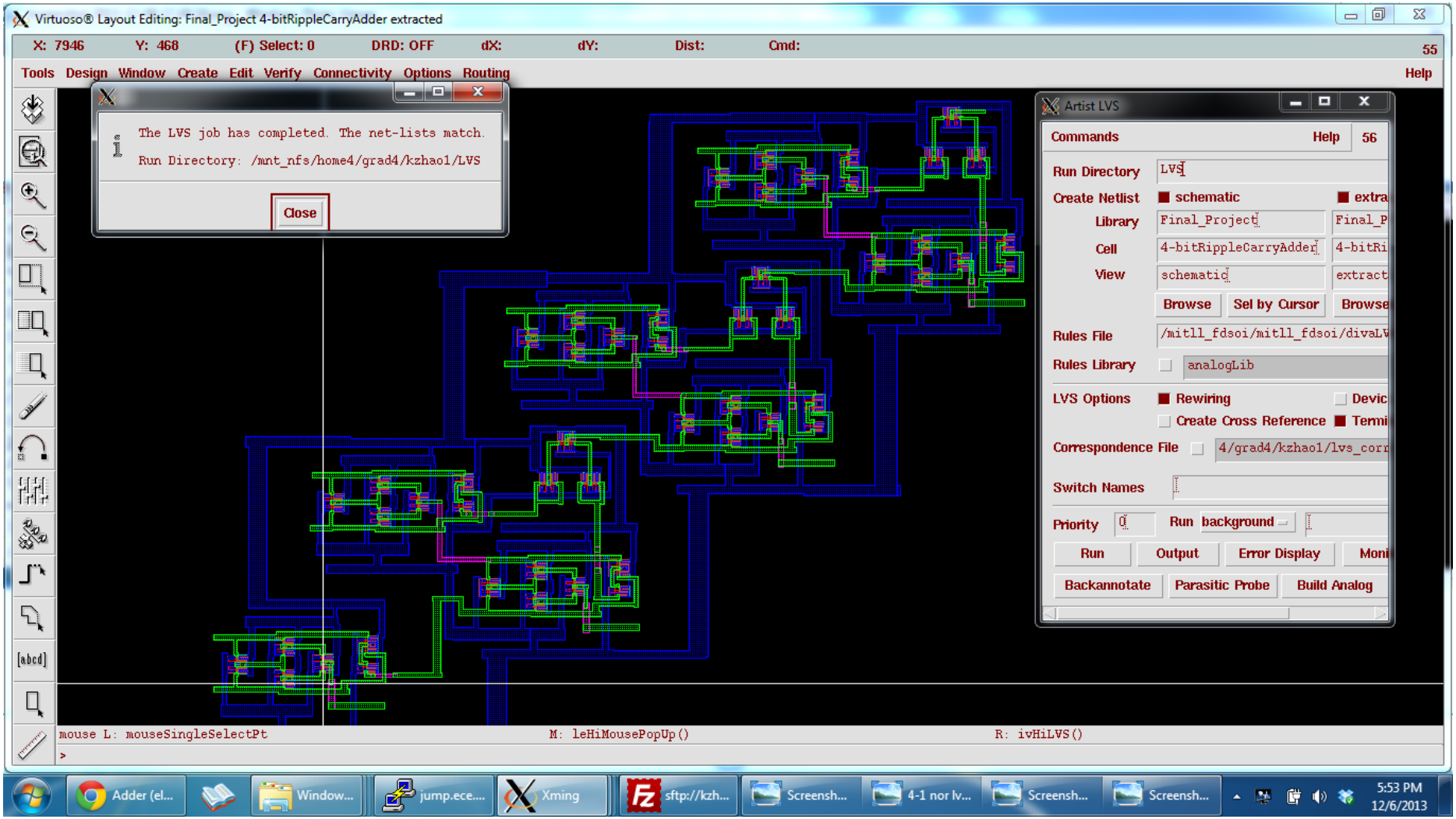

Figure 24: Extracted view of the 4-bit RCA

2.5) 2’s Complement

This block is built using a 1’s complement block and an adder. First, the *A* input becomes inverted. Then, it must be added by 1. Next figure shows the schematic of this block.

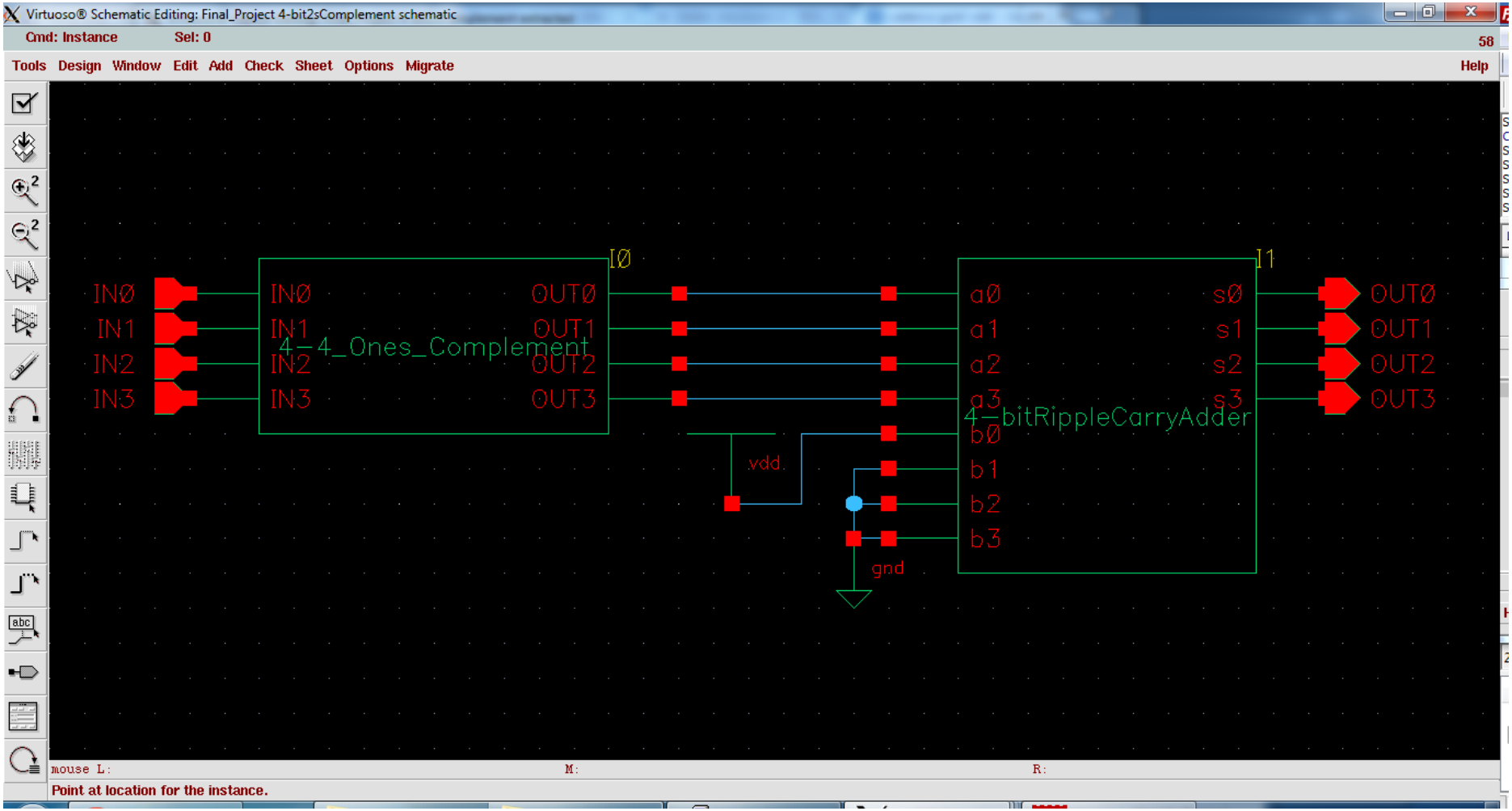

Figure 25: Schematic view of 2's Complement

Next figure shows the simulation result of the 2's Complement block.

Figure 26: Simulation result for the 2's Complement

Next figure shows the layout of the 2's Complement block.

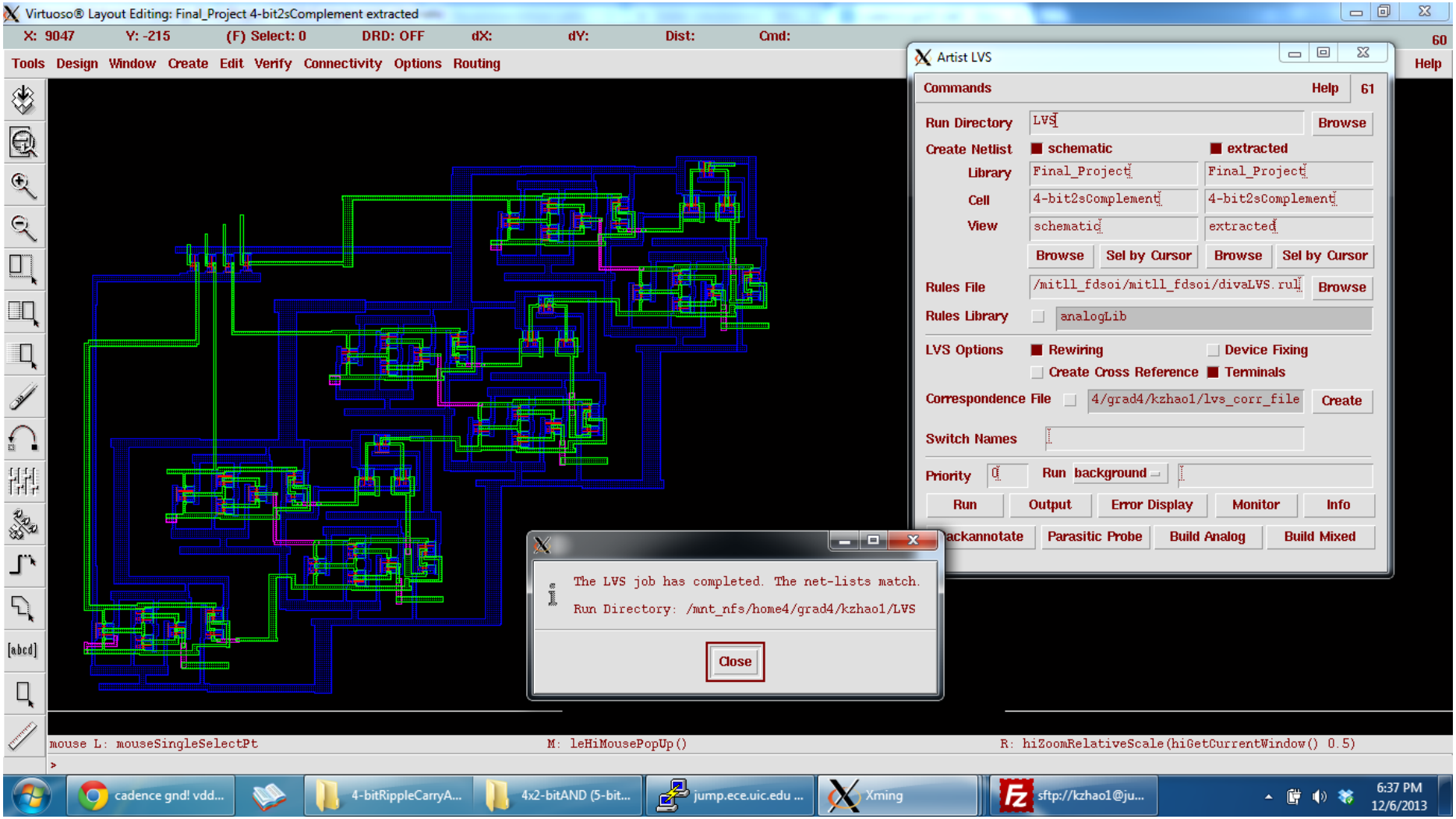

Figure 27: Extracted view of the 2's Complement block

2.6) Addtraction

Subtracting from 1111 is the same as taking the 1's complement. Therefore, this block is built using two 1's Complement block and one adder. Next figure shows the schematic of this block.

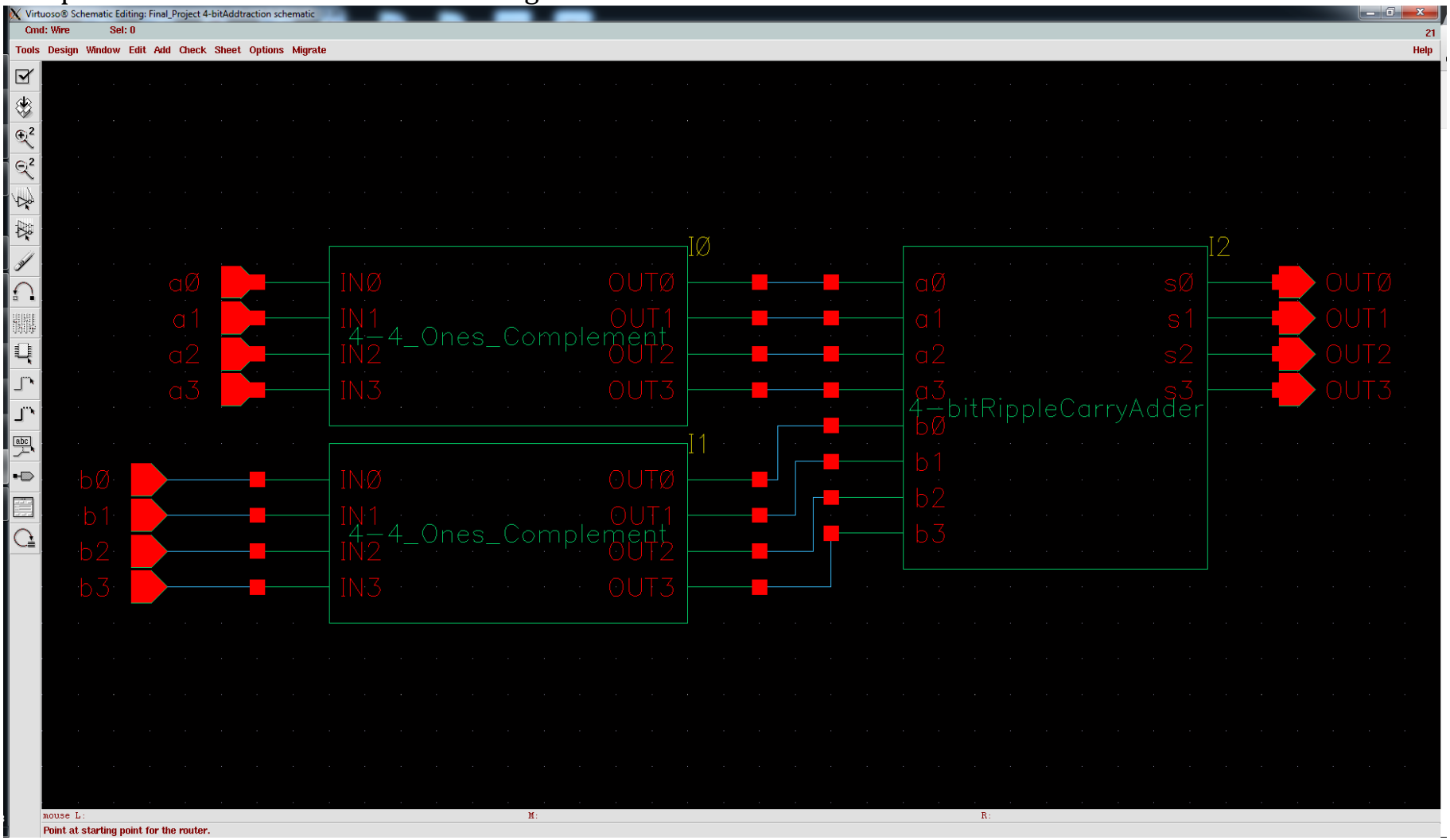

Figure 28: Schematic view of Addtraction block

Next figure shows the simulation result of the Addtraction block.

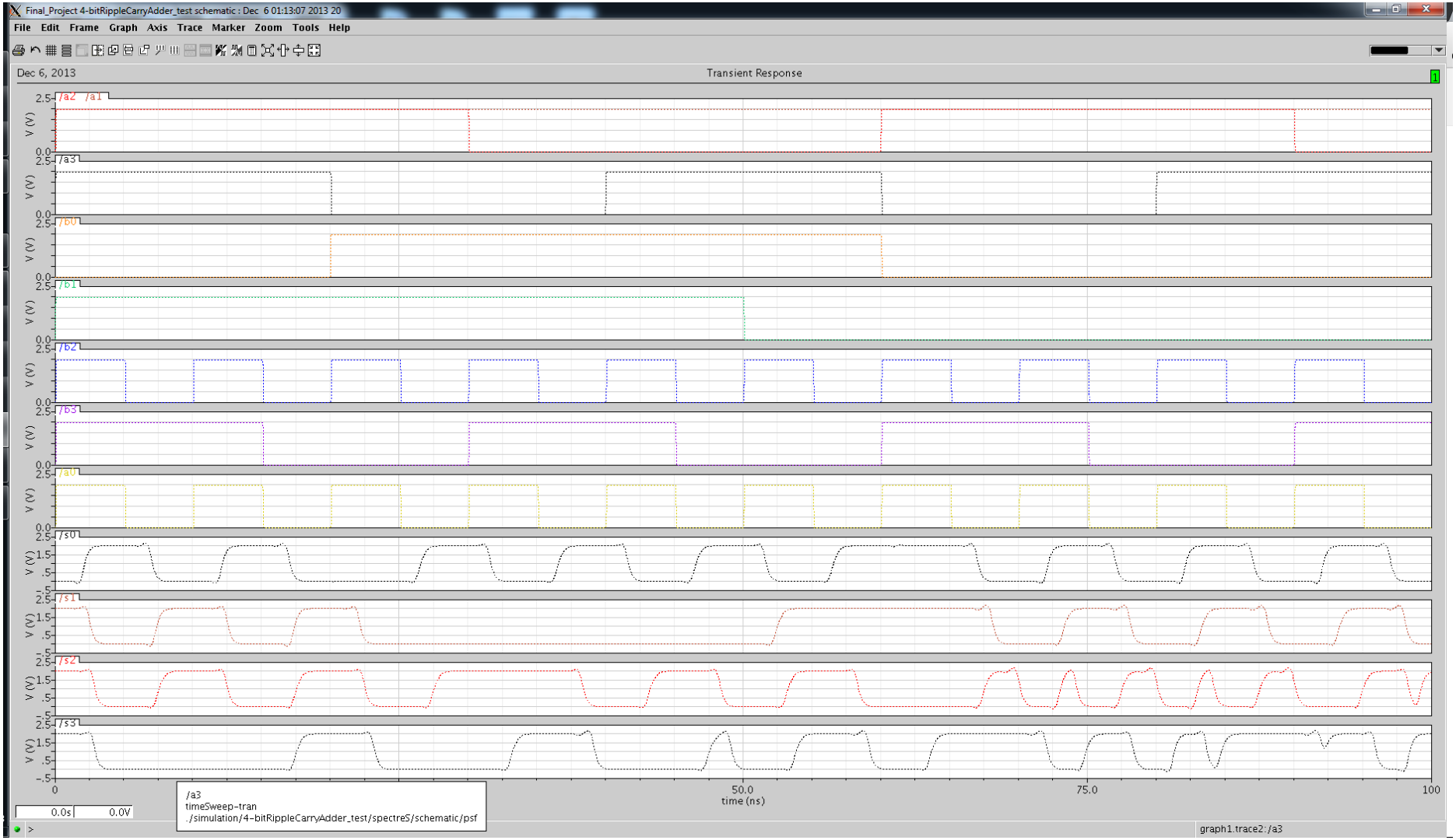

Figure 29: Simulation result for the Addtraction

Next figure shows the layout of the Addtraction block.

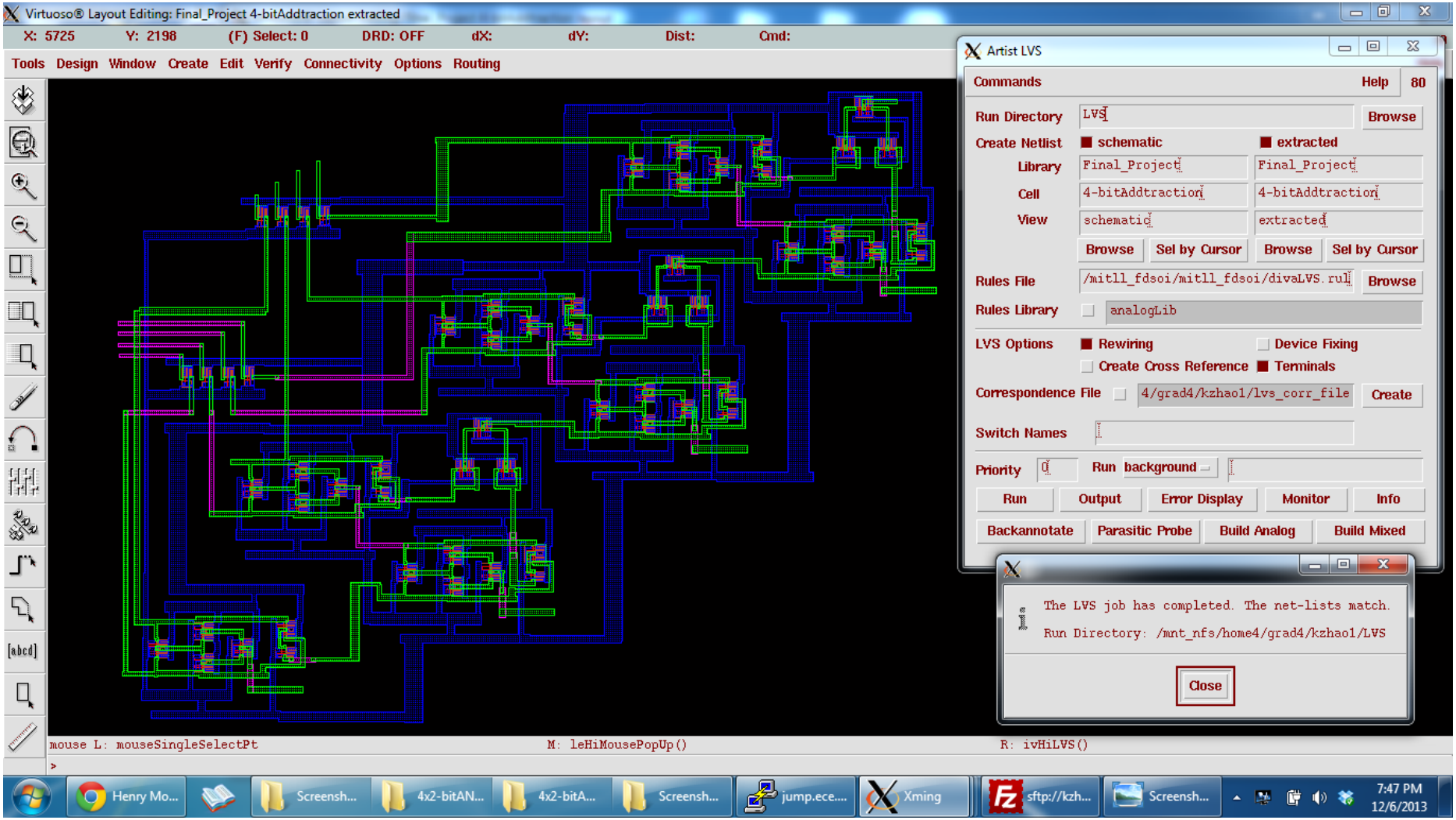

Figure 30: Extracted view of the Addtraction block

Part 3) Additional Units Design:

3.1) 3to6 Decoder:

This block is responsible for generating a signal that selects one of the main units. The inputs to this block are the 3-bit op-codes.

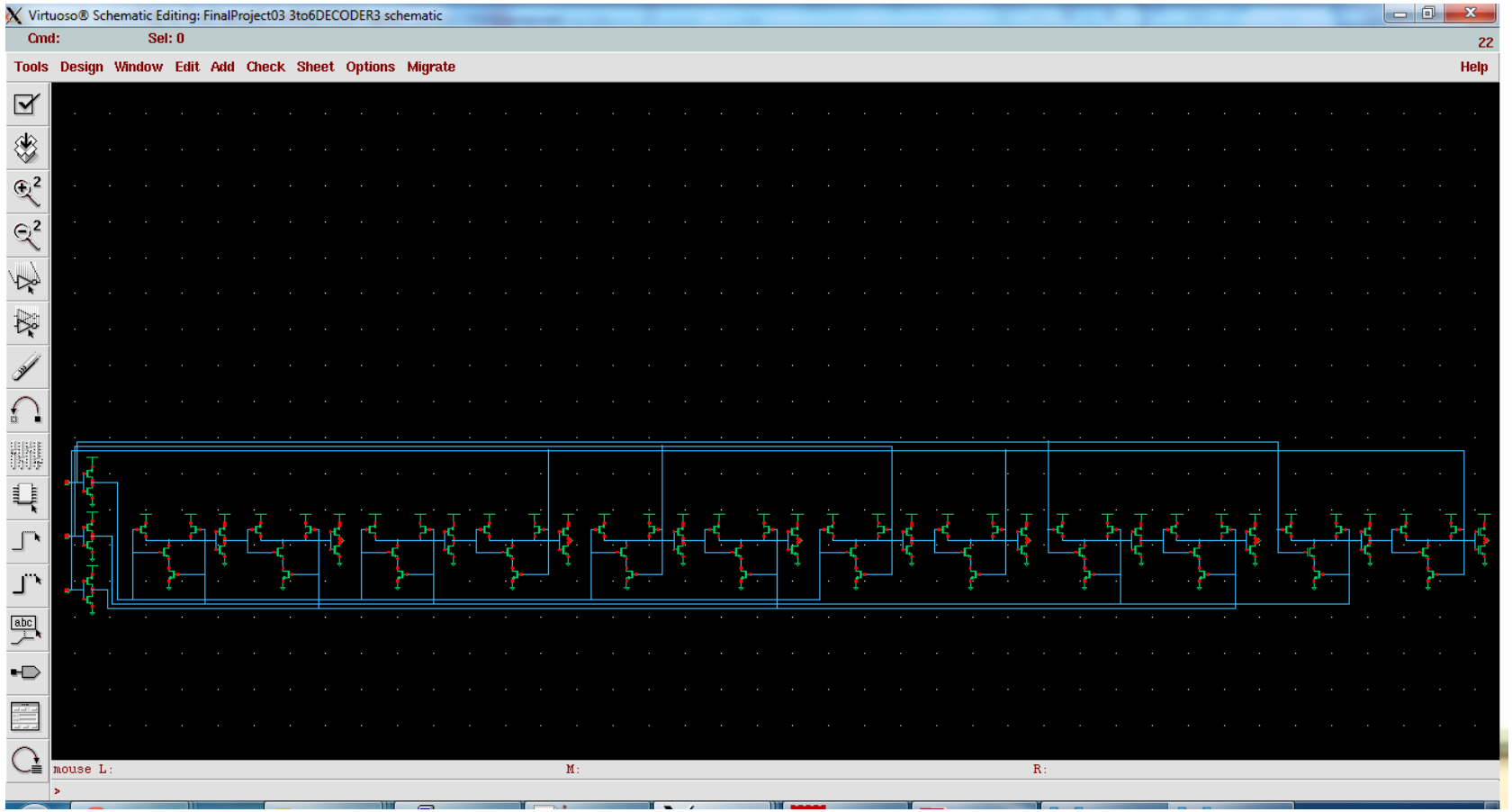

Figure 30: Schematic of the 3 to 6 Decoder

Next figure shows the simulation result of the Decoder. As it can be seen, at any time, only one of the outputs are high

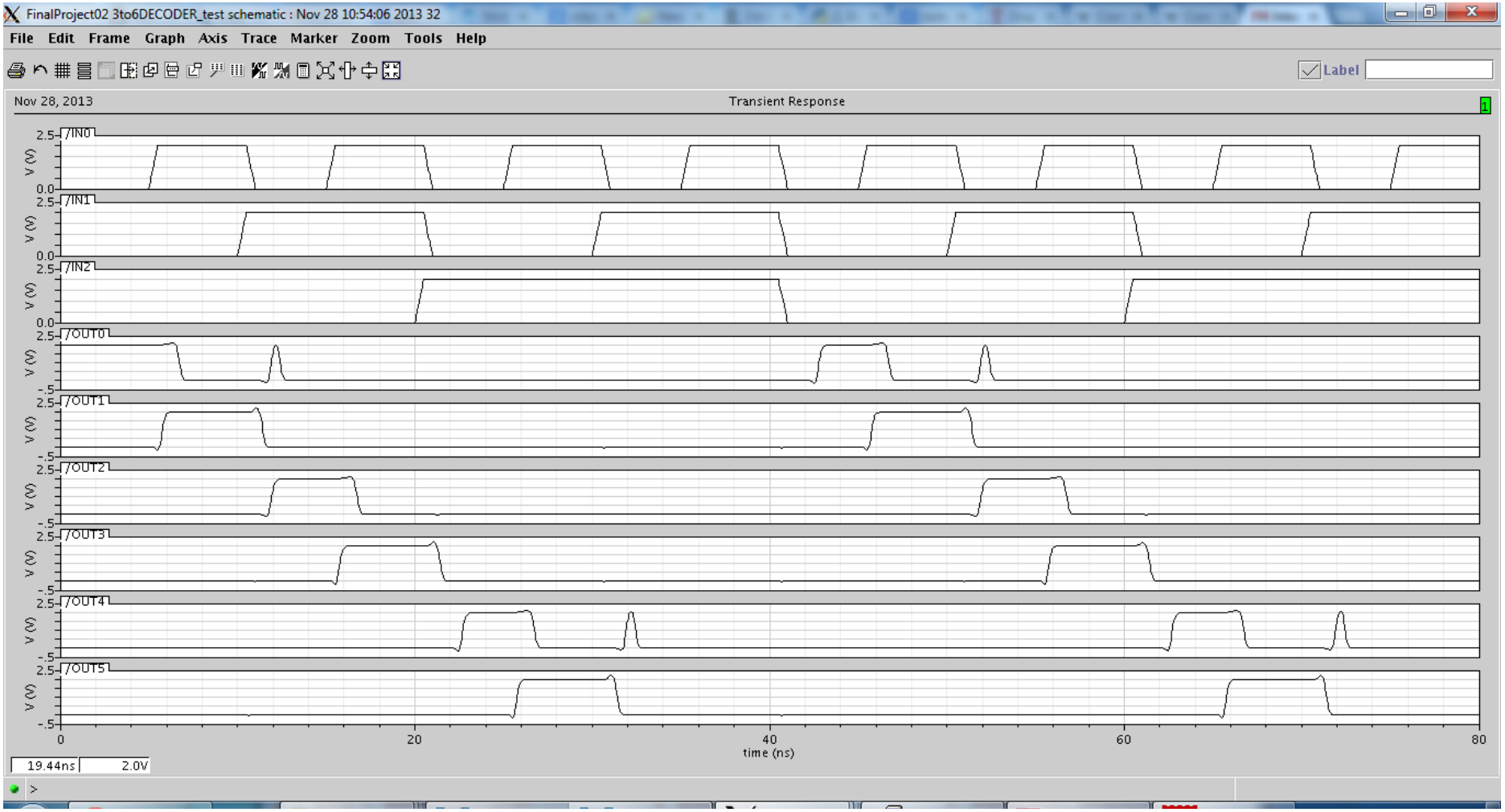

Figure 31: Simulation result for the 3to6 Decoder

Next figure shows the layout of the Decoder.

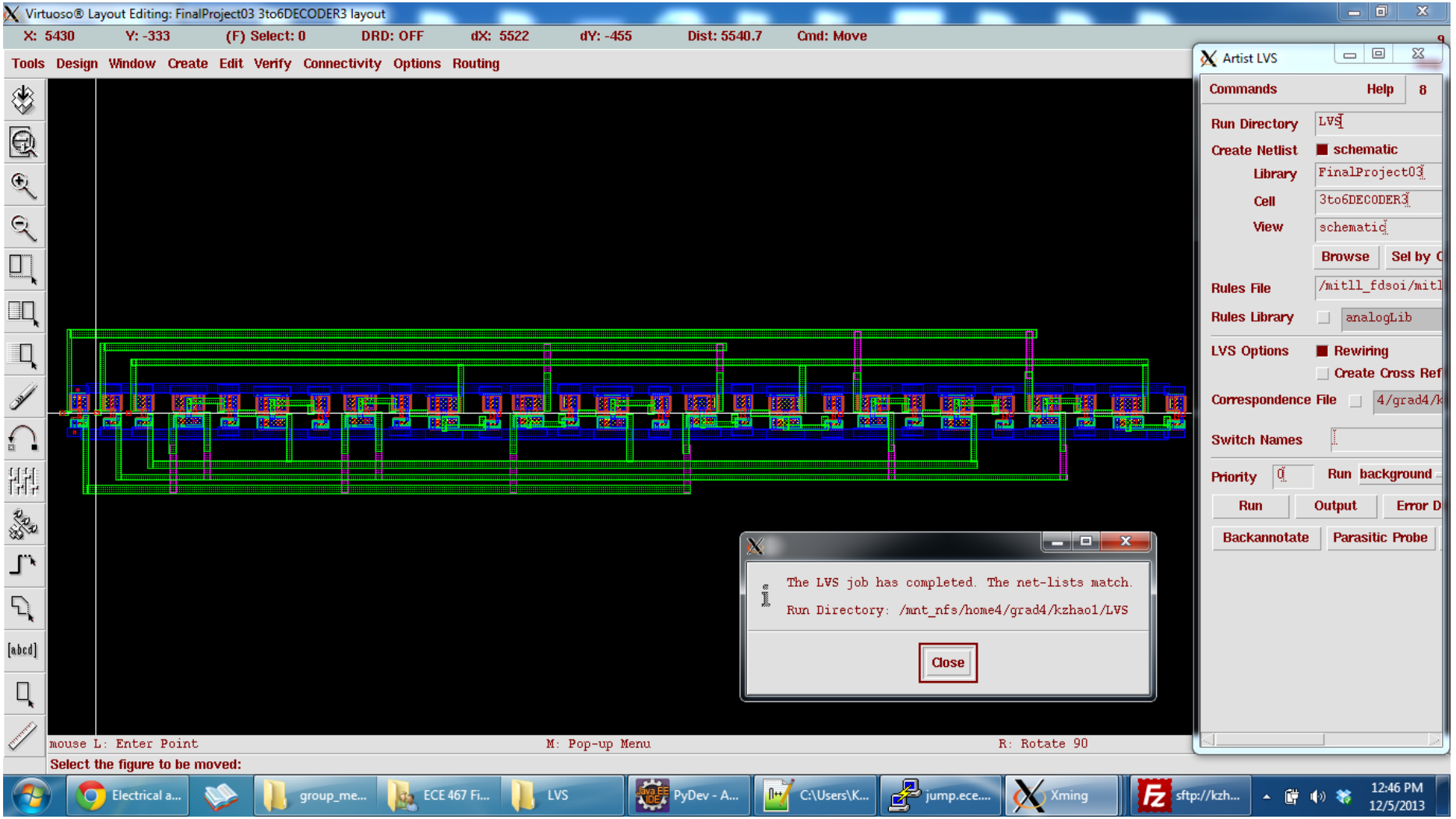

Figure 32: Extracted view of the Decoder block

3.2) 5-bit AND (4x2-bit AND):

This block consists of 4x2-bit AND gates. As explained above, this block is needed as a part of the selecting circuitry. Next figure shows the schematic for a two input AND gate which is used to implement the 4x2-bit AND gate

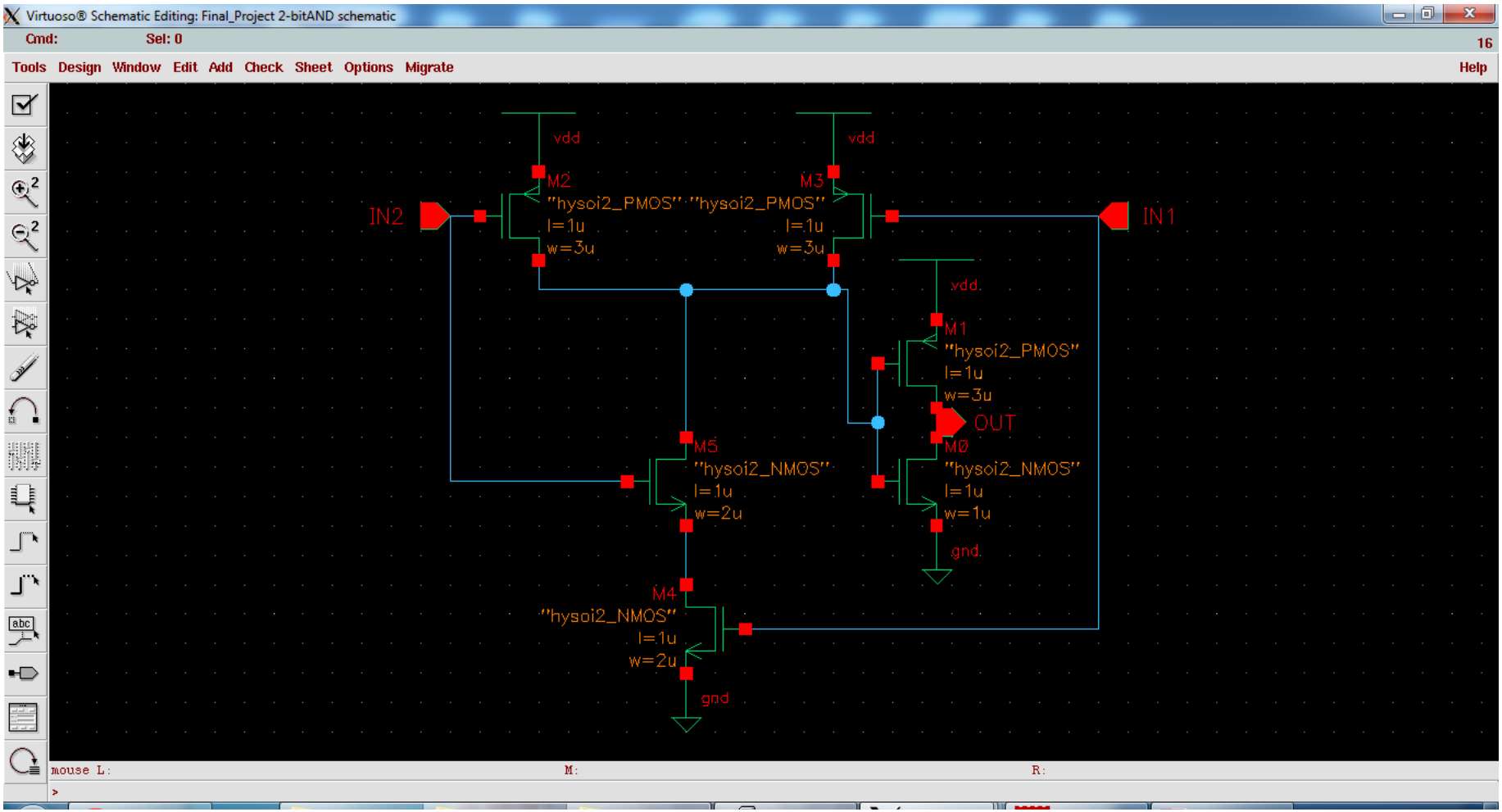

Figure 31: Schematic of 2-bit AND gate

Next figure shows the schematic the 4x2-bit AND gate.

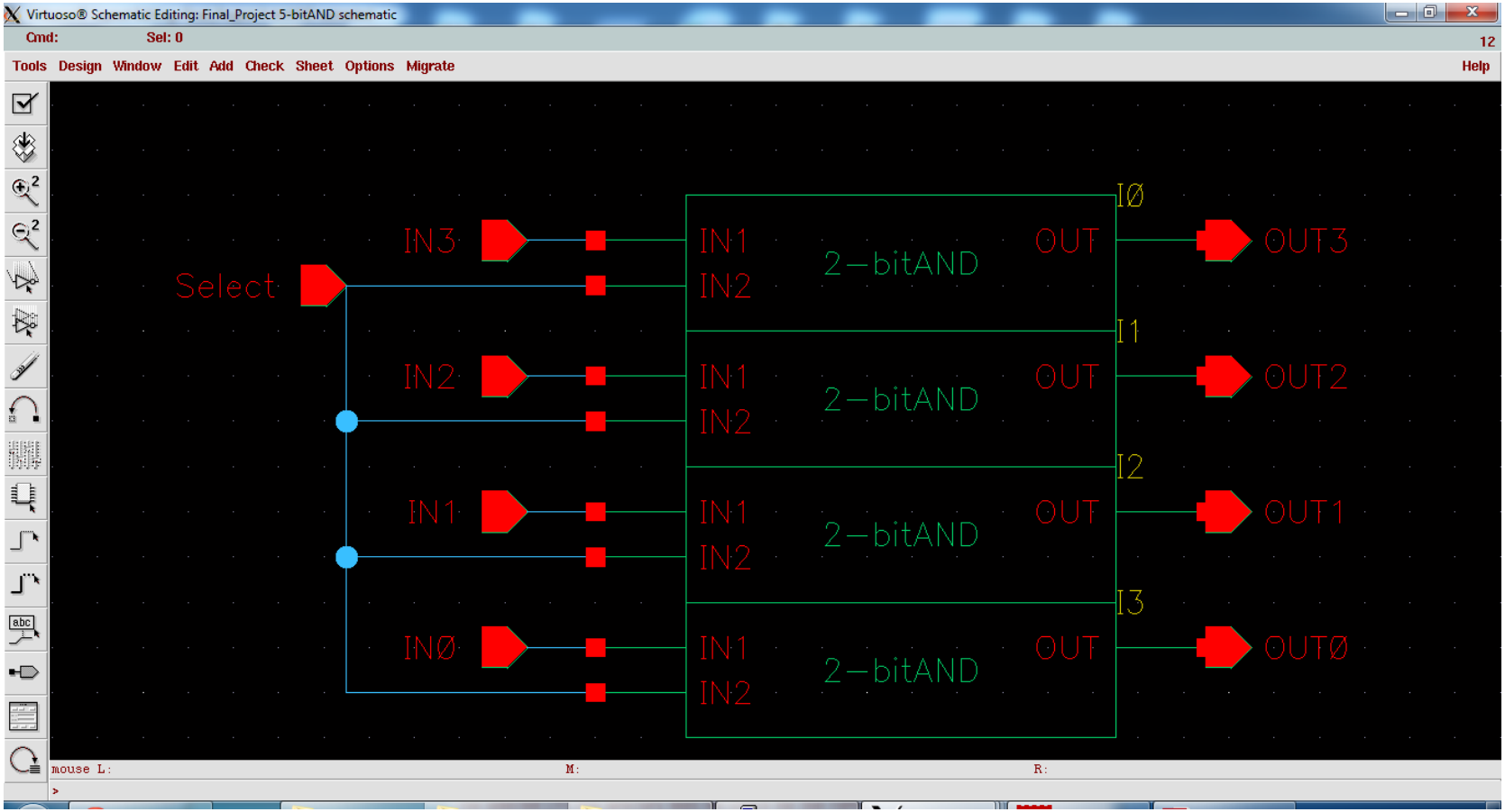

Figure 32: Schematic of the 4x2-bit AND

Next figure shows the simulation result of this block. When the controlling signal is one, the outputs are equal to the inputs, otherwise the outputs are all zero.

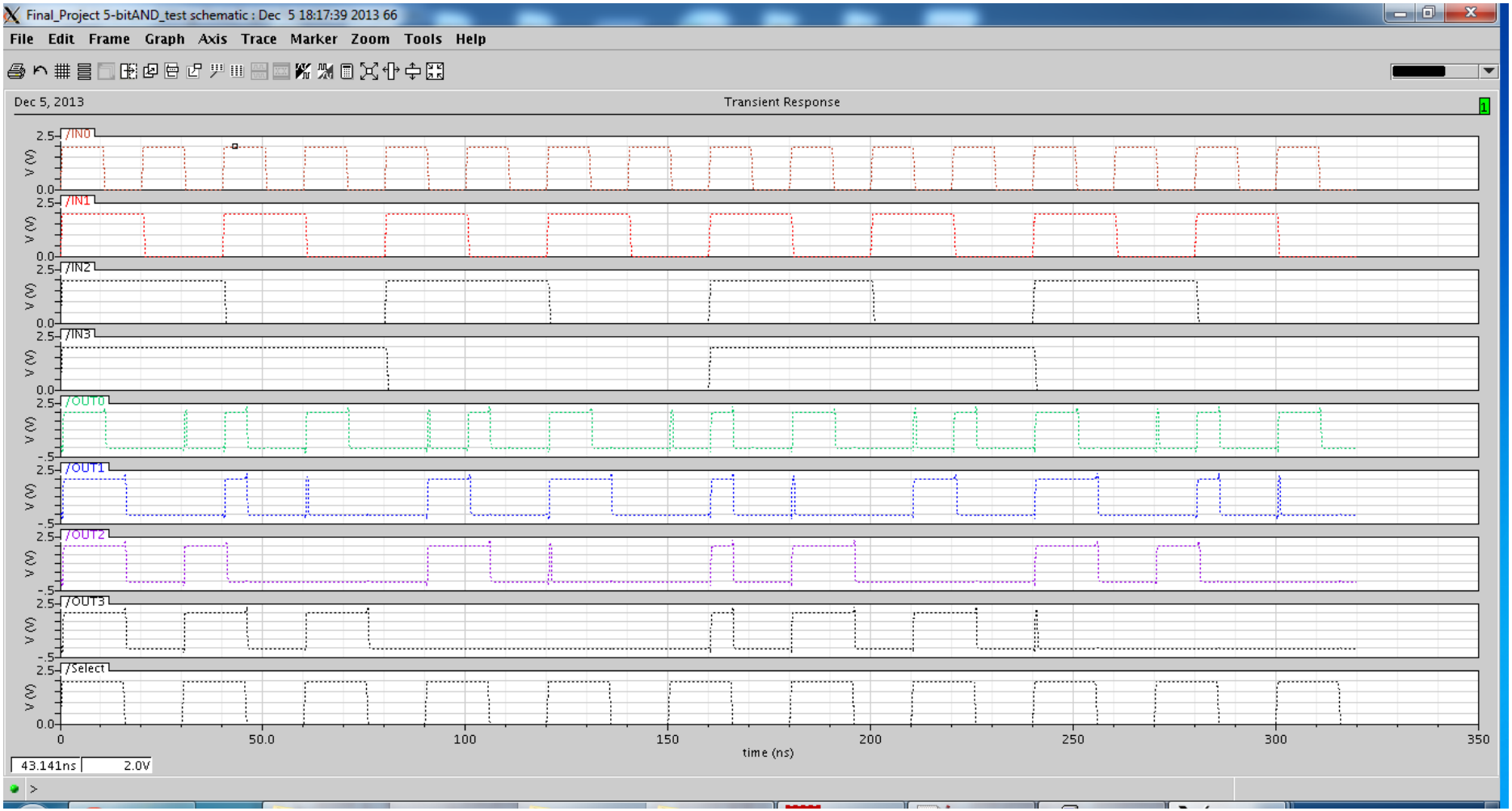

Figure 33: Simulation result for the 4x2-bit AND

Next figure shows the layout of this block.

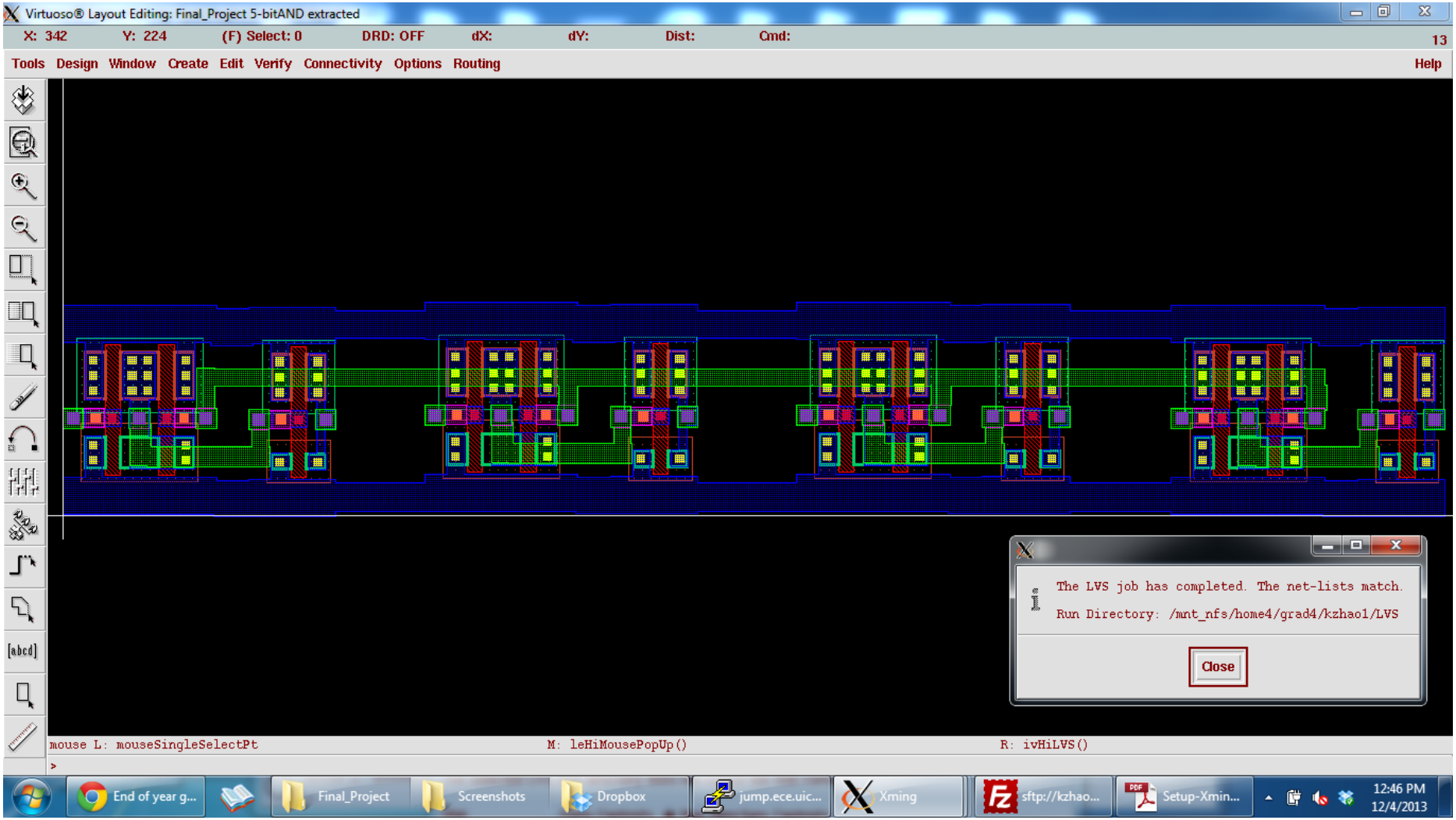

Figure 34: Layout of the 4 * 2-bit AND gates block

3.1) 6-bit OR:

This block is built using 2-input NAND gates. Next figure shows the schematic of this block.

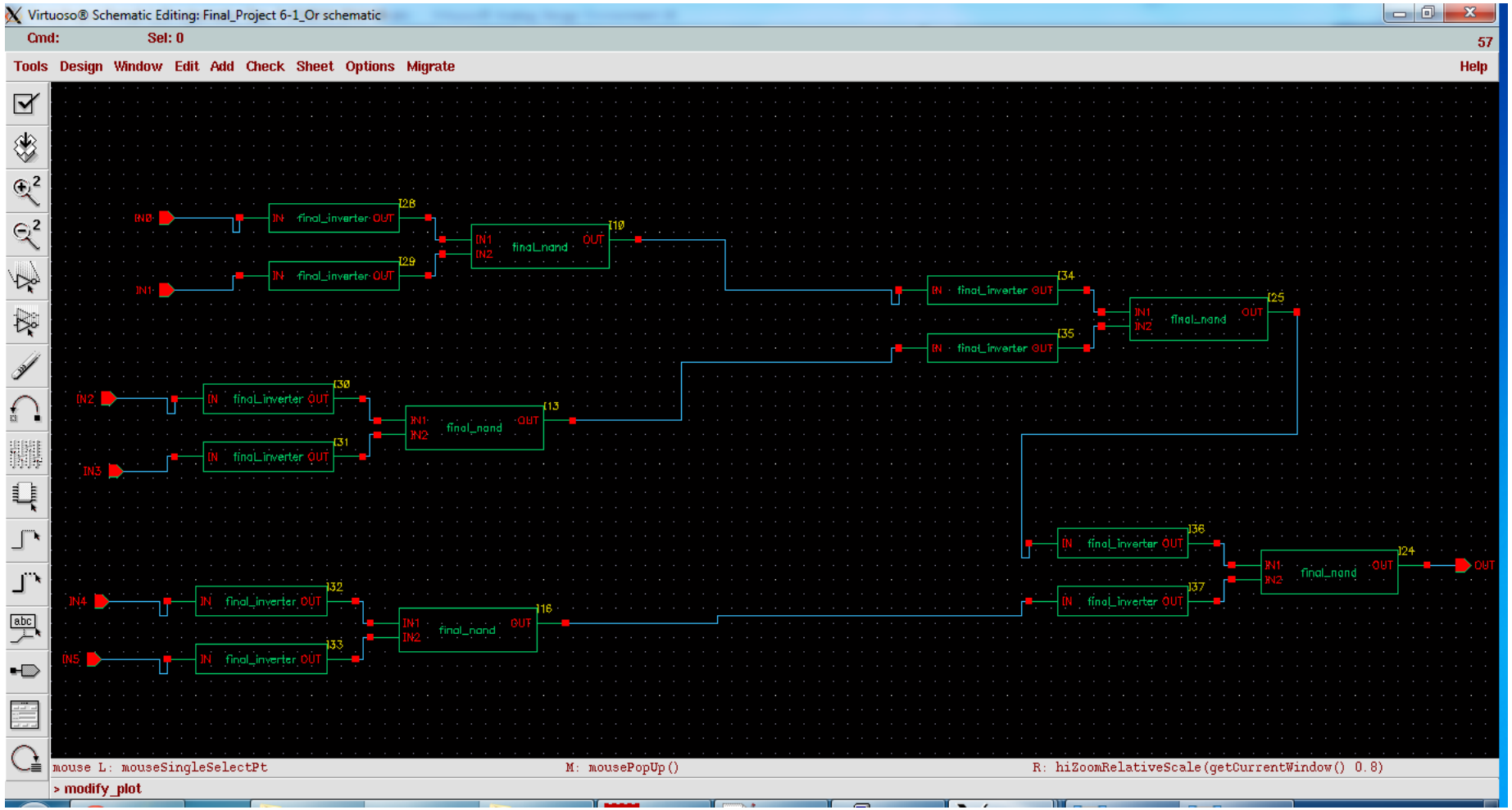

Figure 35: Schematic of the 6-bit OR gate

Next figure shows the simulation result of the 6-bit OR. As it can be seen, the output is high if any input is high.

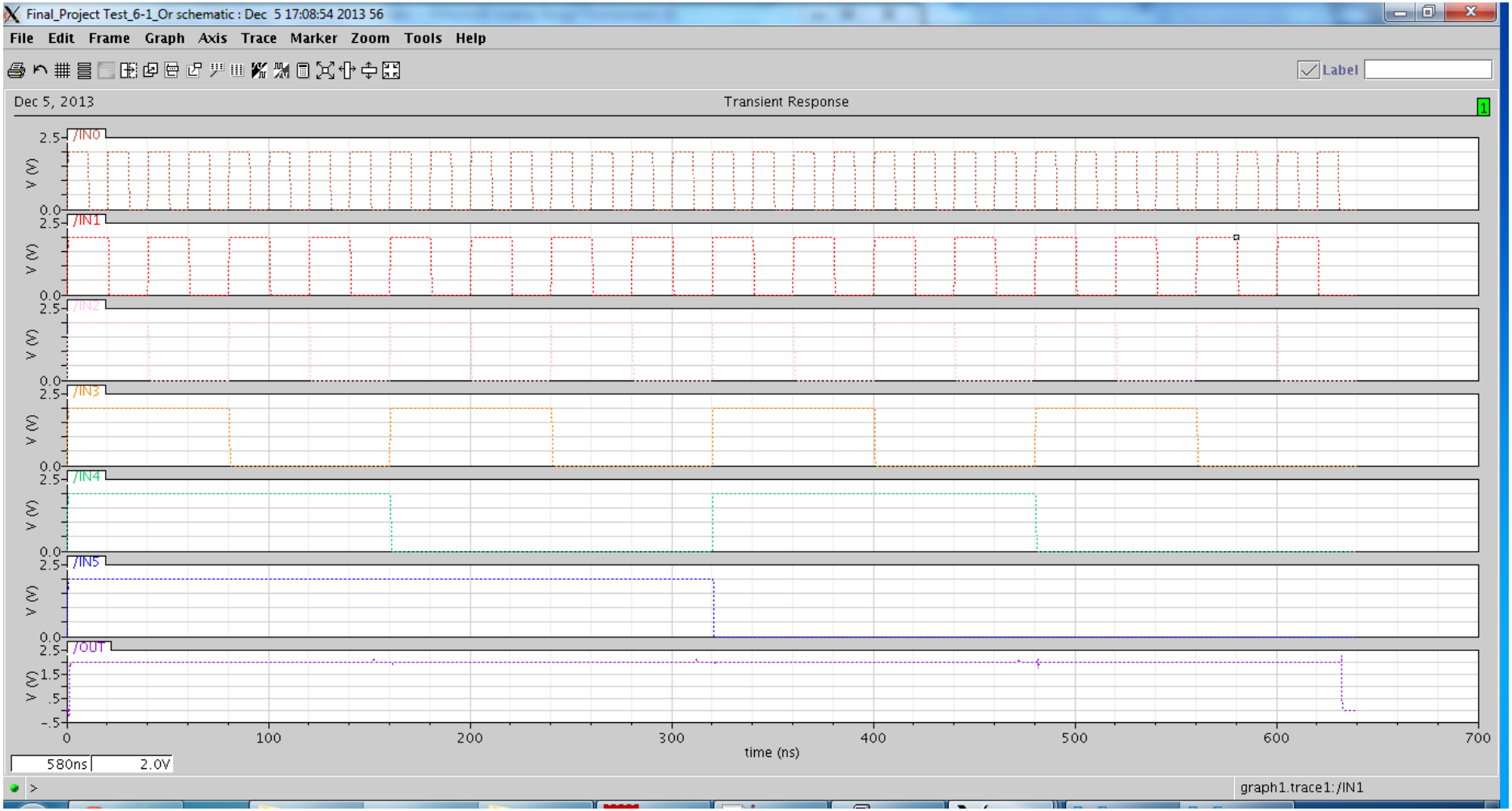

Figure 36: Simulation result for the 6-bit OR gate

Next figure shows the layout of the 6-bit OR gate.

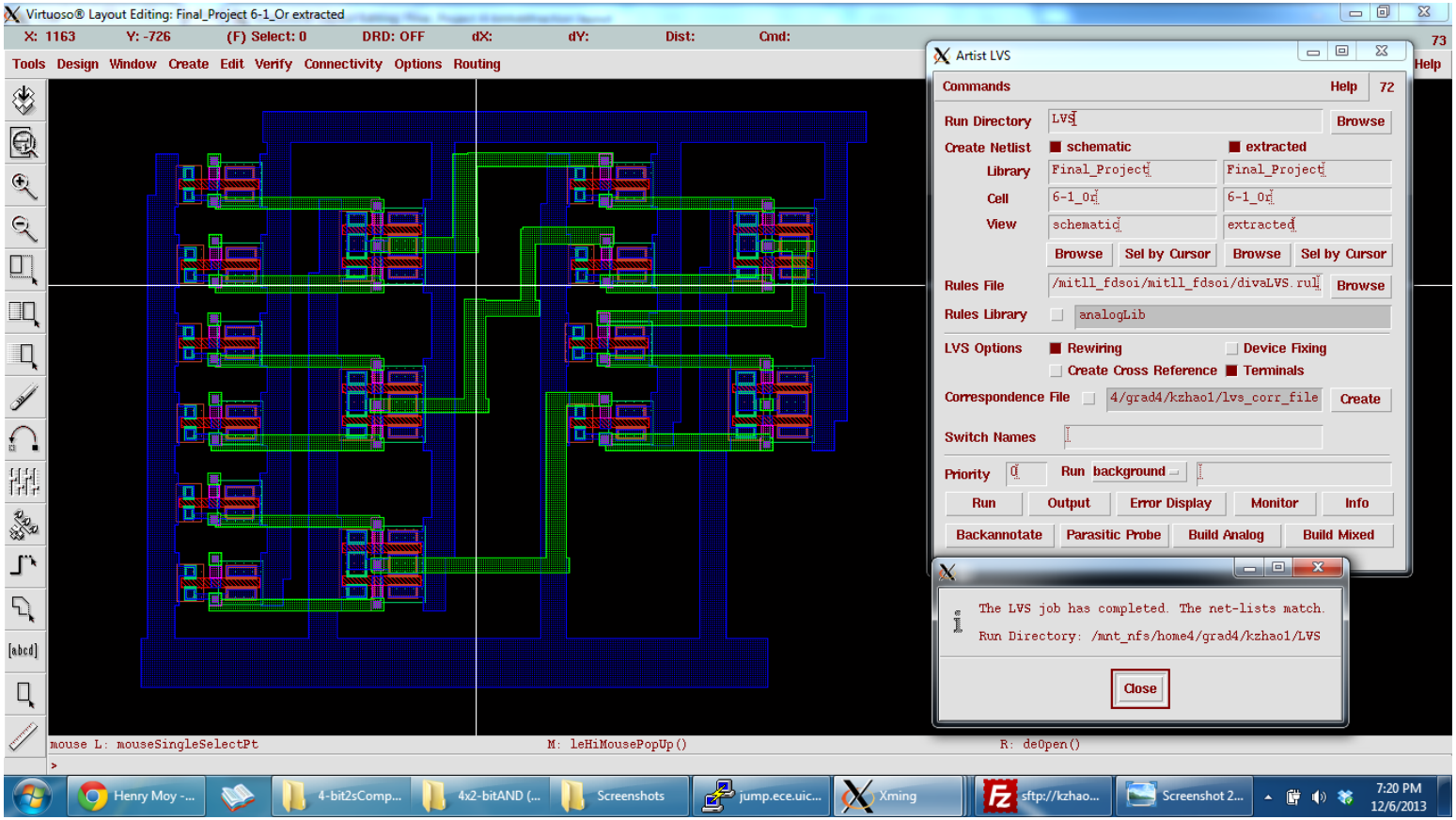

Figure 37: Extracted view of the 6-bit OR gate

Part 4) (Additional) Requirements:

Clock frequency of 100 MHz or more:

A frequency of 100 MHz corresponds to a clock period of 10 ns. The post layout simulation (Figures 4, 6, and 8) shows that the output is correctly within 10 ns after the inputs change.

Load of 10 fF in addition to internal parasitic elements present in the circuit:

Figures 3, 5, and 7 shows that the ALU does drive a load of 10 fF. The post layout simulaton (Figures 4, 6, and 8) is done with the analog_extracted view of the ALU, which takes parasitic capacitance into account.

Schematics:

Shown first of every component.

Simulation:

Shown last of every component.

DRC Pass:

Showing LVS pass is enough to show DRC pass because DRC pass is required for etracting, which is required for LVS pass.

LVS Pass:

Shown middle of every component along with the layout.

Post-layout Simulation:

Shown for the ALU, which demonstrates the functionality of various functions while meeting time constraints.

Part 5) Final Project Naming Conventions

There is a lack of naming conventions because multiple people worked on designing an building this ALU. Listed below is the component's cell name for every component of the ALU in the order it appears in this report so spectators will know the reference of each cell.

Compoenent Name is this Report	**Cell Name in Final_Project Library**
ALU	final_ALU
4-bit NAND	4-1_Nand
4-bit NOR	4-1_Nor
1's Complement	4-4_Ones_Complement
4-bit Adder	4-bitRippleCarryAdder
2's Complement	4-bit2sComplement
Addtraction	4-bitAddtraction
3to6 Decoder	3to6DECODER3
4x2-bit AND	5-bitAND
6-bit OR	6-1_Or
Half Adder	2-bitHalfAdder
Full Ader	3-bitFullAdder
2-bit AND	2-bitAND
2-bit OR	2_OR
2-bit XOR	2-bitXOR