Designing a Signed 4-bit Radix-4 Booth Multiplier

Task1: Multiplicand Multiple Generation (Outputs are 8-bit for an 8-bit product) (MMG Block)

- **Val Zero (8-bit 00000000);**

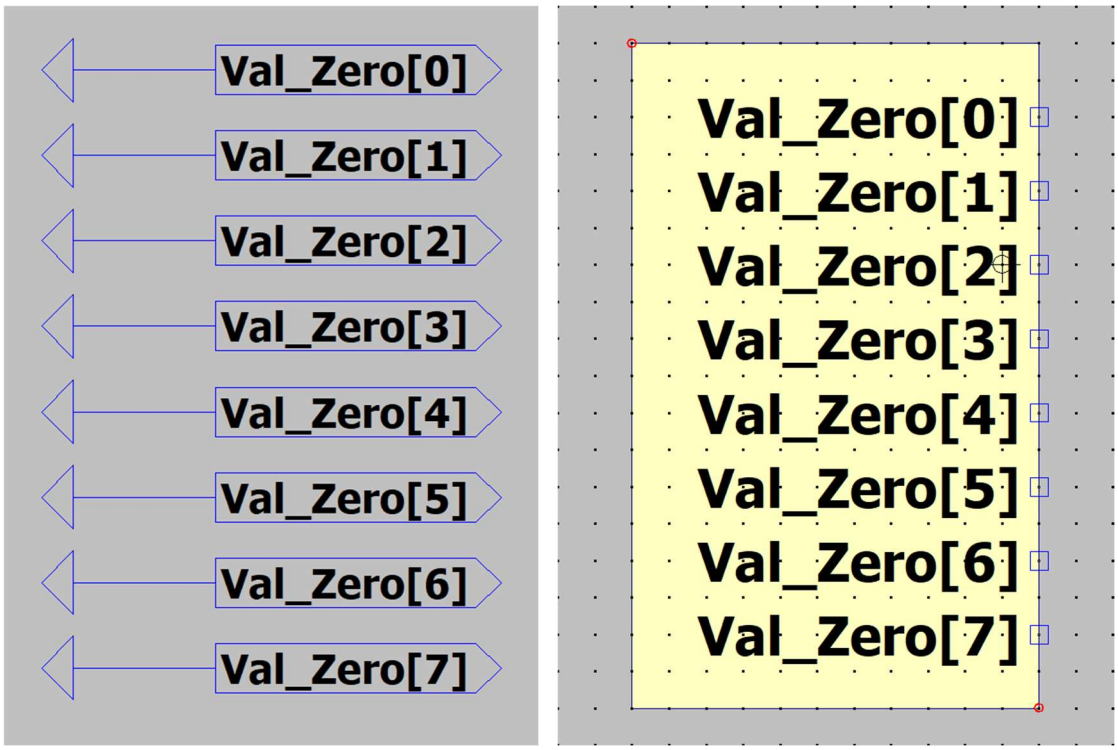


Figure 1: Val_Zero Block Circuit and Symbol

The Val_Zero block is implemented to generate an 8-bit zero value (00000000). In this block, all eight output lines are connected to logic LOW (ground). It is used as one of the selectable inputs in the partial product selector (multiplexer), and is chosen when the Booth encoder determines that a zero multiple of the multiplicand is required.

- **Val_Plus_A (8-bit $A_3A_3A_3A_3A_3A_2A_1A_0$);**

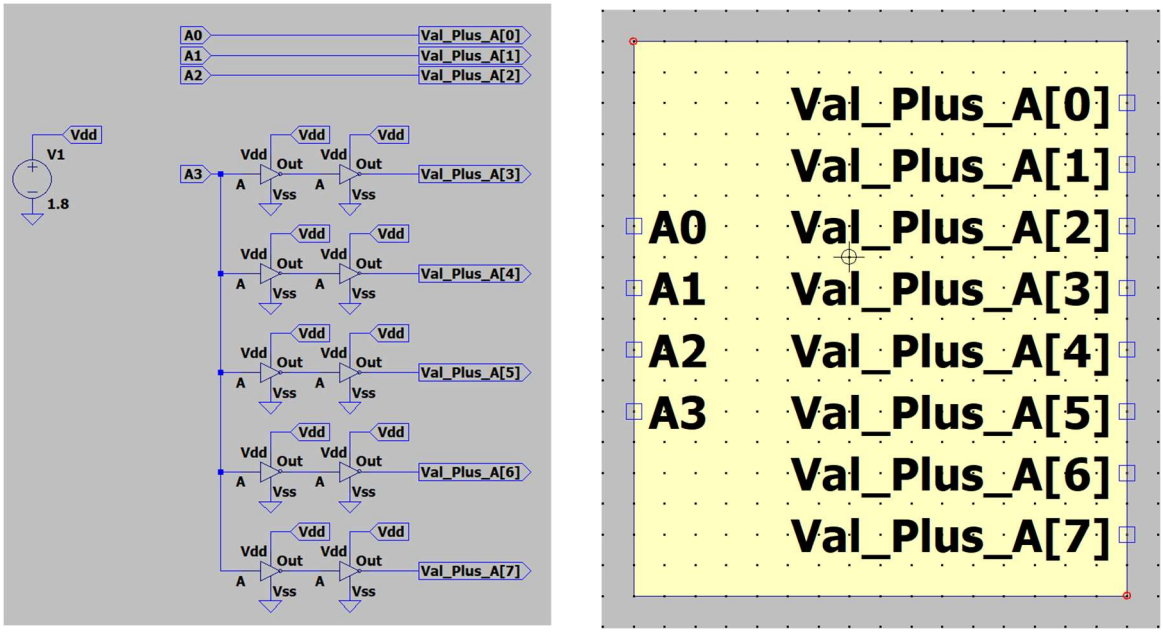


Figure 2: Val_Plus_A Block Circuit and Symbol

The Val_Plus_A block is designed to generate the sign-extended 8-bit representation of the 4-bit multiplicand A. In this implementation, the lower 4 bits are directly taken from A_0 to A_3, while the upper 4 bits are all connected to the sign bit A_3 to achieve proper sign extension. This block is selected when the Booth encoder indicates that the multiplicand should be added as +A during partial product generation.

- **Val_Plus_2A (8-bit $A_3A_3A_3A_3A_2A_1A_00$);**

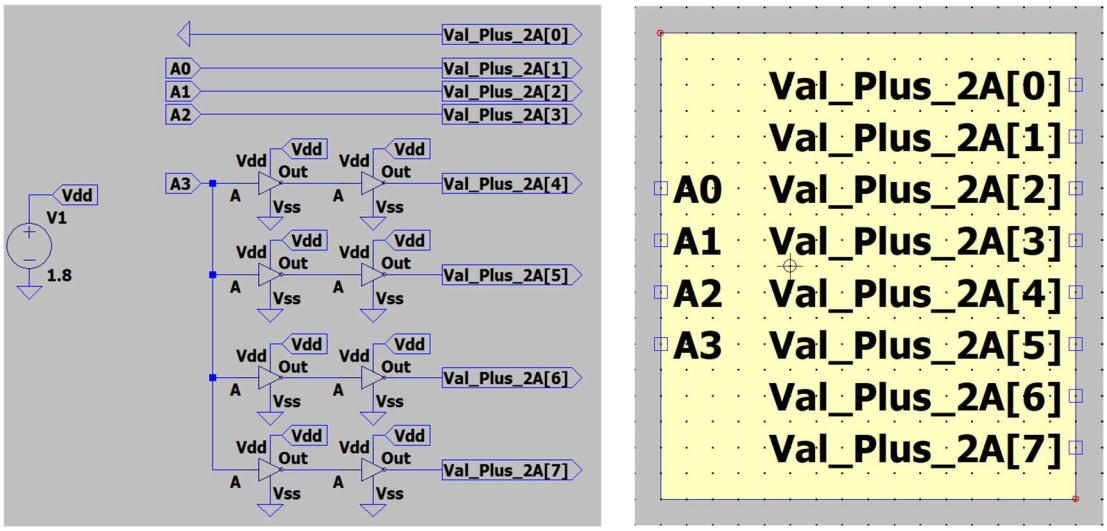


Figure 3: Val_Plus_2A Block Circuit and Symbol

The Val_Plus_2A block is implemented to generate the sign-extended 8-bit version of the multiplicand A multiplied by 2. This is achieved by shifting A one bit to the left and extending the sign bit A_3 to the higher-order bits. The least significant bit is set to logic LOW (0), while the upper bits are filled with A_3 to maintain the correct sign in 2's complement format. This block is used when the Booth encoder selects the +2A multiple during partial product generation.

- **Val_Minus_A (8-bit, 2's complement of Val_Plus_A);**

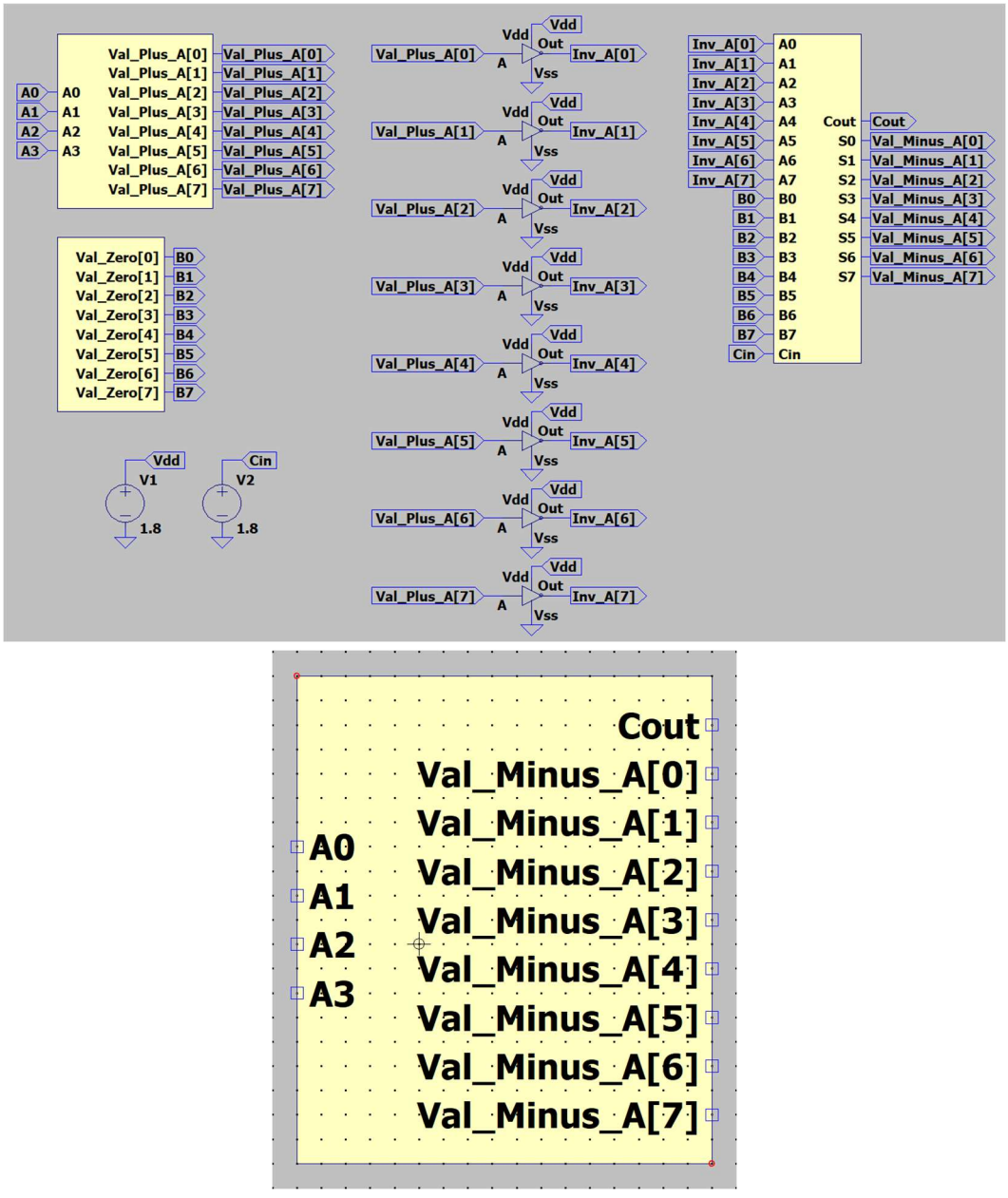


Figure 4: Val_Minus_A Block Circuit and Symbol

The Val_Minus_A block is used to produce the 8-bit two's complement representation of the multiplicand A, corresponding to the value –A. First, all bits of the Val_Plus_A output are inverted using NOT gates. Then, a binary 1 is added to the inverted result using an 8-bit incrementer. Through this process, the correct signed negative value of A is generated and sign-extended, ready to be selected by the partial product selector when the Booth encoder requests –A.

- **Val_Minus_2A (8-bit, 2's complement of Val_Plus_2A);**

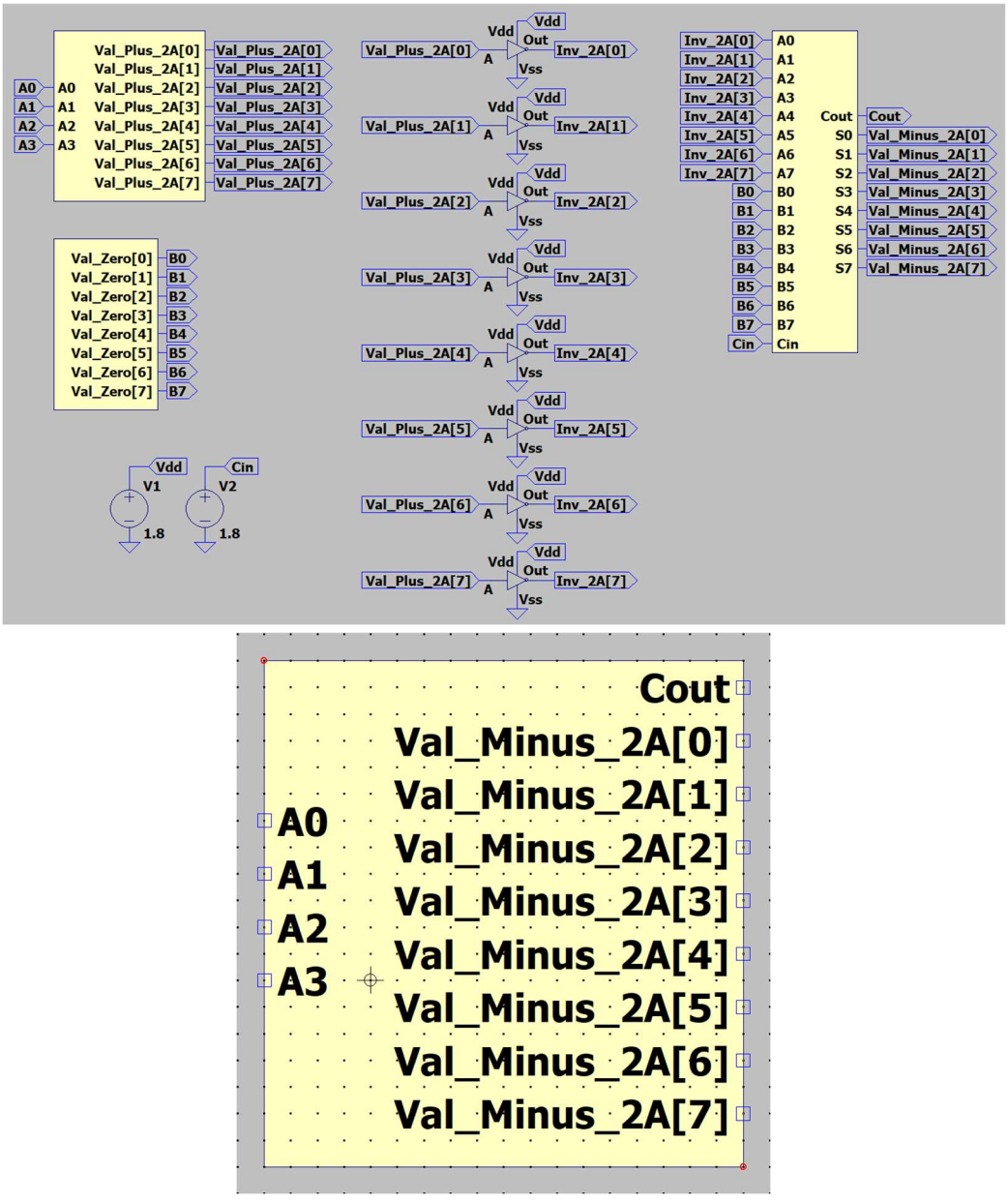


Figure 5: Val_Minus_2A Block Circuit and Symbol

- **MMG Block**

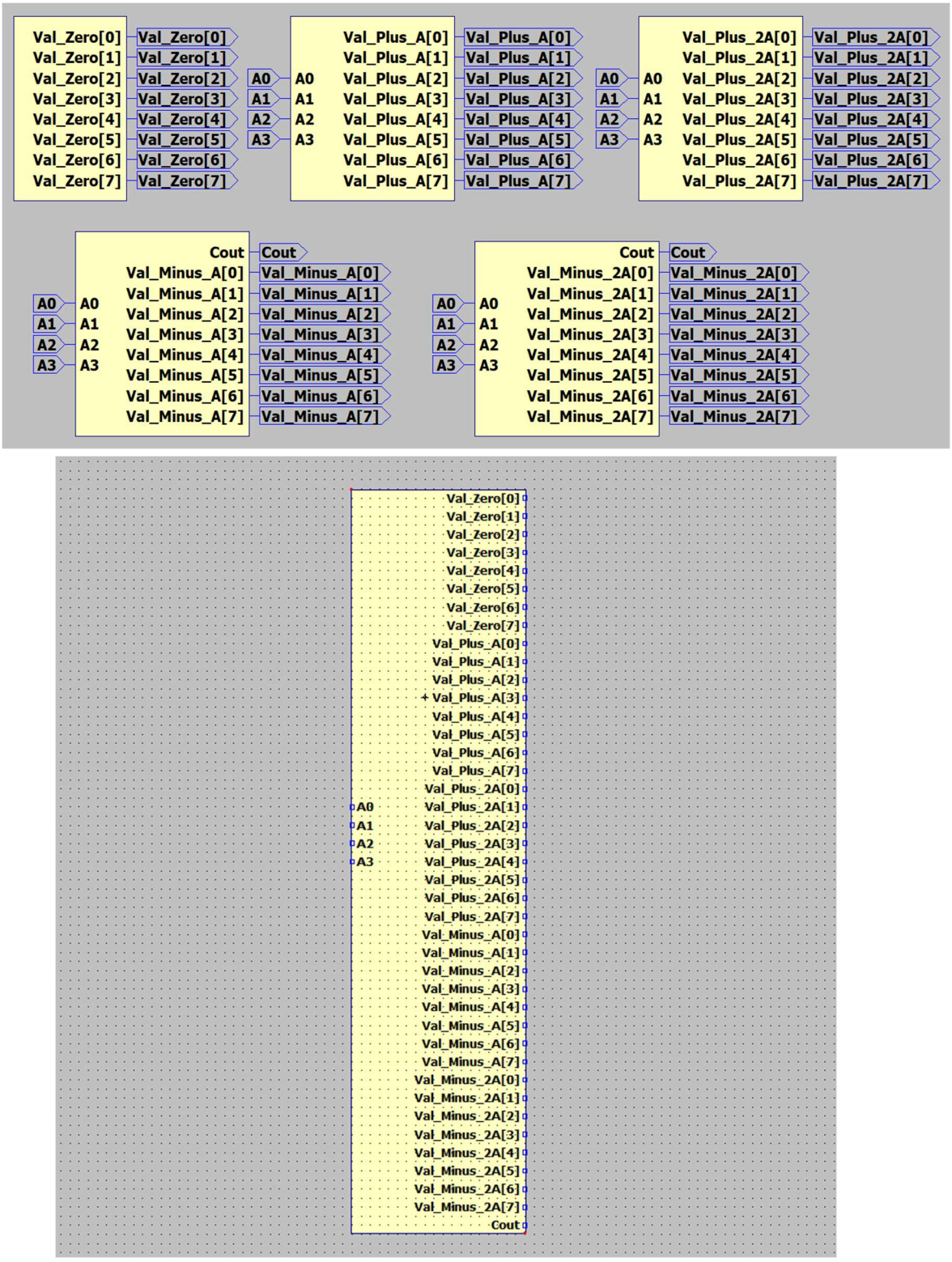


Figure 6: MMG Block Circuit and Symbol

The Multiplicand Multiple Generation (MMG) block is designed to generate five different 8-bit outputs from a 4-bit signed multiplicand A: 0, +A, –A, +2A, and –2A. These values are computed using combinations of direct wiring, left shifting, sign extension, and two's complement operations. Each output is implemented in a dedicated sub-block such as Val_Zero, Val_Plus_A, Val_Plus_2A, Val_Minus_A, and Val_Minus_2A. The MMG block enables partial product selection in the Booth multiplier by supplying these predefined multiplicand multiples based on control signals from the Booth encoder.

Task2: Booth Encoders (Radix-4) (BE Block)

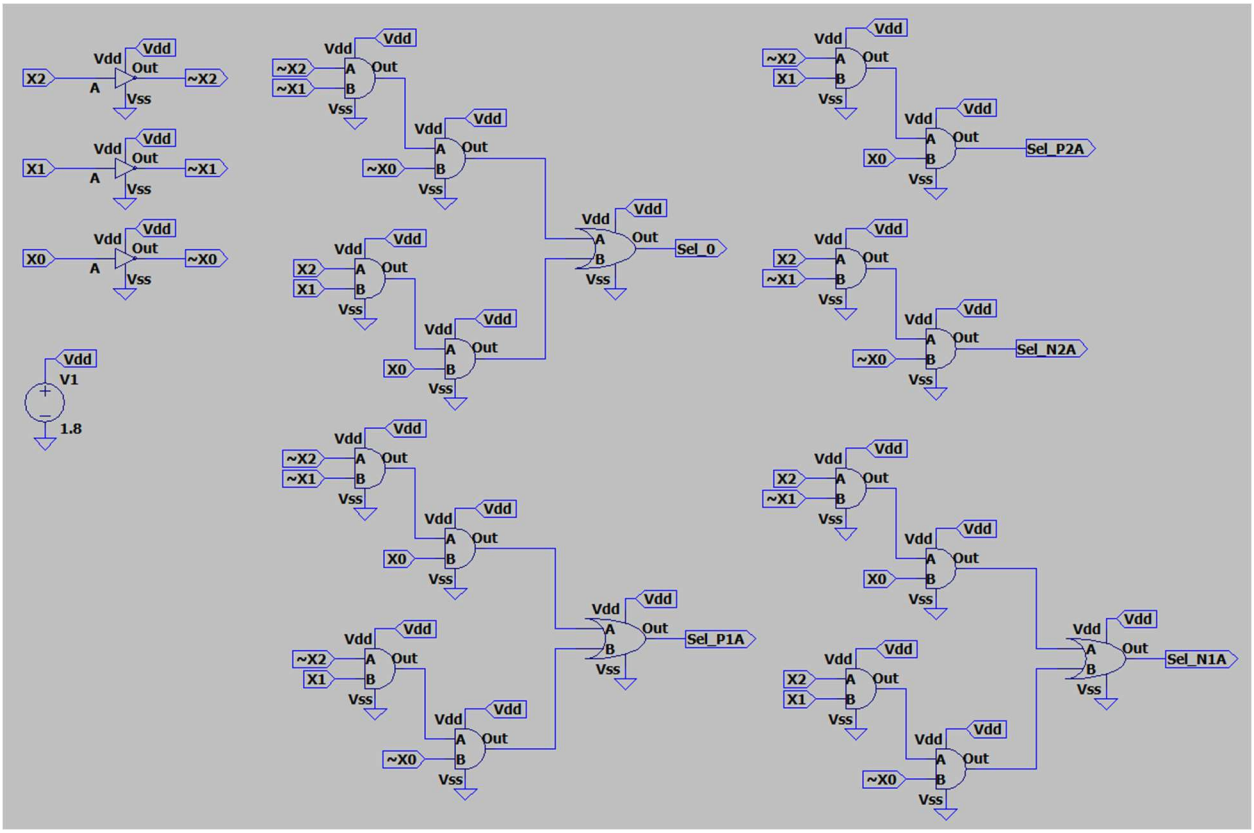


Figure 7: Booth Encoder Block Circuit

The Booth Encoder block is responsible for analyzing three consecutive bits of the multiplier (X_2, X_1, X_0) and generating five one-hot encoded control signals that determine which multiplicand multiple should be selected. The logic expressions shown in the truth table are used to minimize the number of partial products by mapping each 3-bit input combination to one of the operations: 0, +A, –A, +2A, or –2A. These control signals—Sel_0, Sel_P1A, Sel_N1A, Sel_P2A, and Sel_N2A—are derived using AND-OR logic based on the specific input pattern.

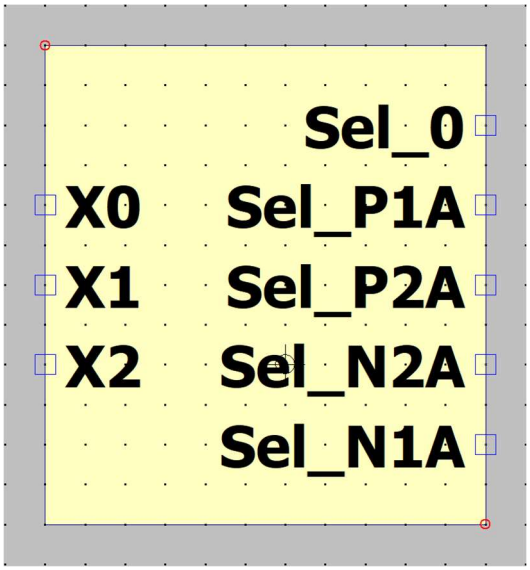
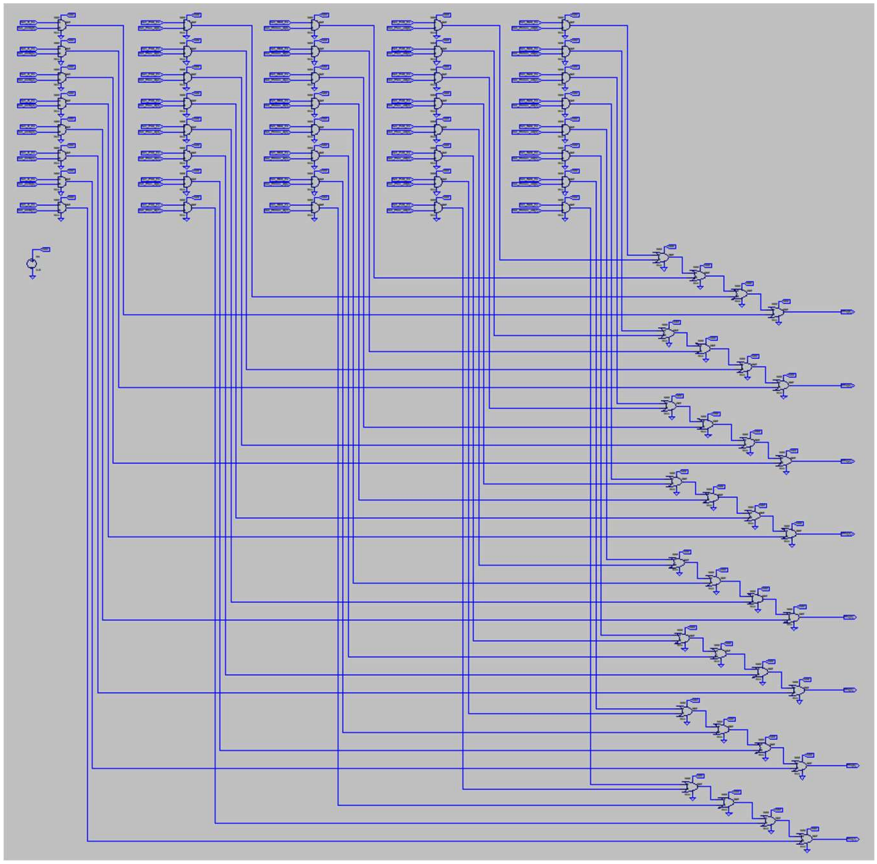
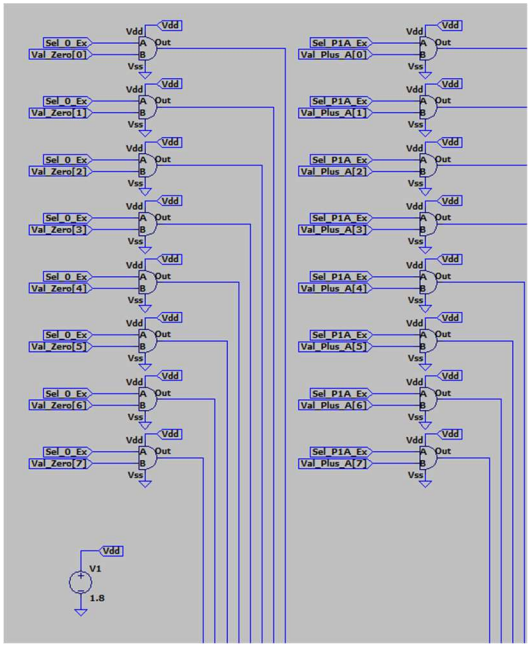
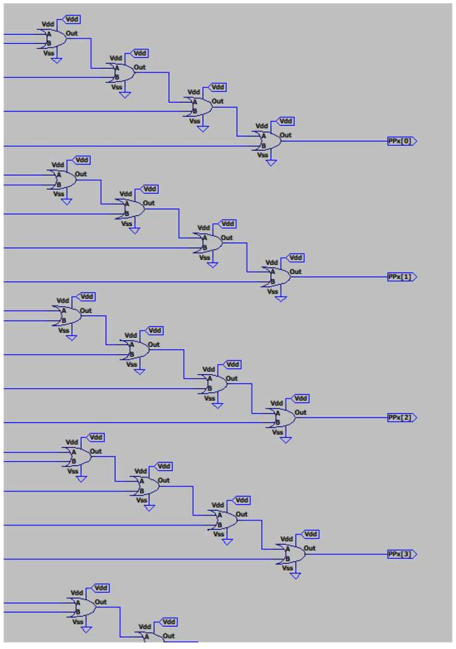


Figure 8: Booth Encoder Block Symbol

Task3: Partial Product Selectors (8-bit wide, 5-to-1 Multiplexers) (PPS Block)







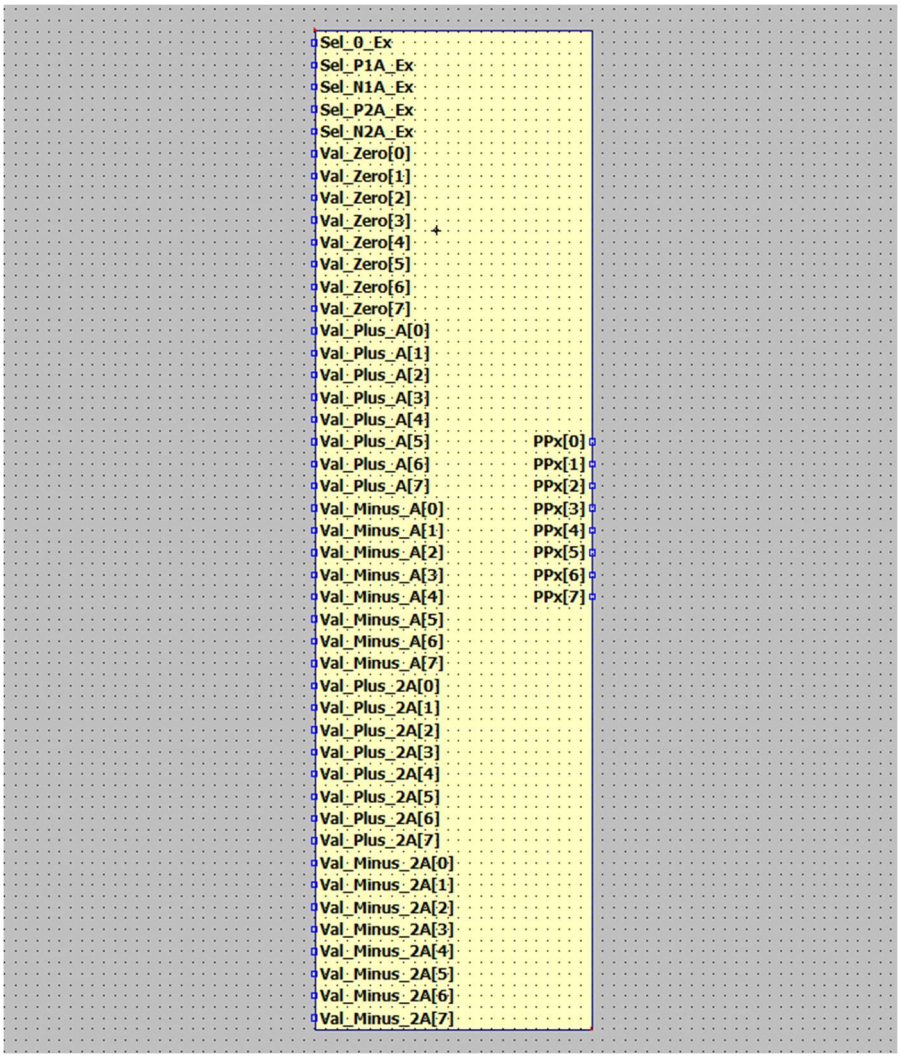


Figure 9: PPS Block Circuit Zoomed Out View and Zoomed In View with Symbol

The Partial Product Selector (PPS) block is implemented using an 8-bit wide 5-to-1 multiplexer. It is used to select one of the five precomputed multiplicand multiples—0, +A, –A, +2A, or –2A—based on the control signals generated by the Booth Encoder. For each output bit, the corresponding value from the selected input is routed through a logic structure composed of five AND gates and one OR gate. The truth table from the assignment is utilized to define which control line should be active for each possible 3-bit multiplier group, ensuring that only one partial product is selected for each encoding step.

Task4: Shifter for PP1_value (SHIFTER Block)

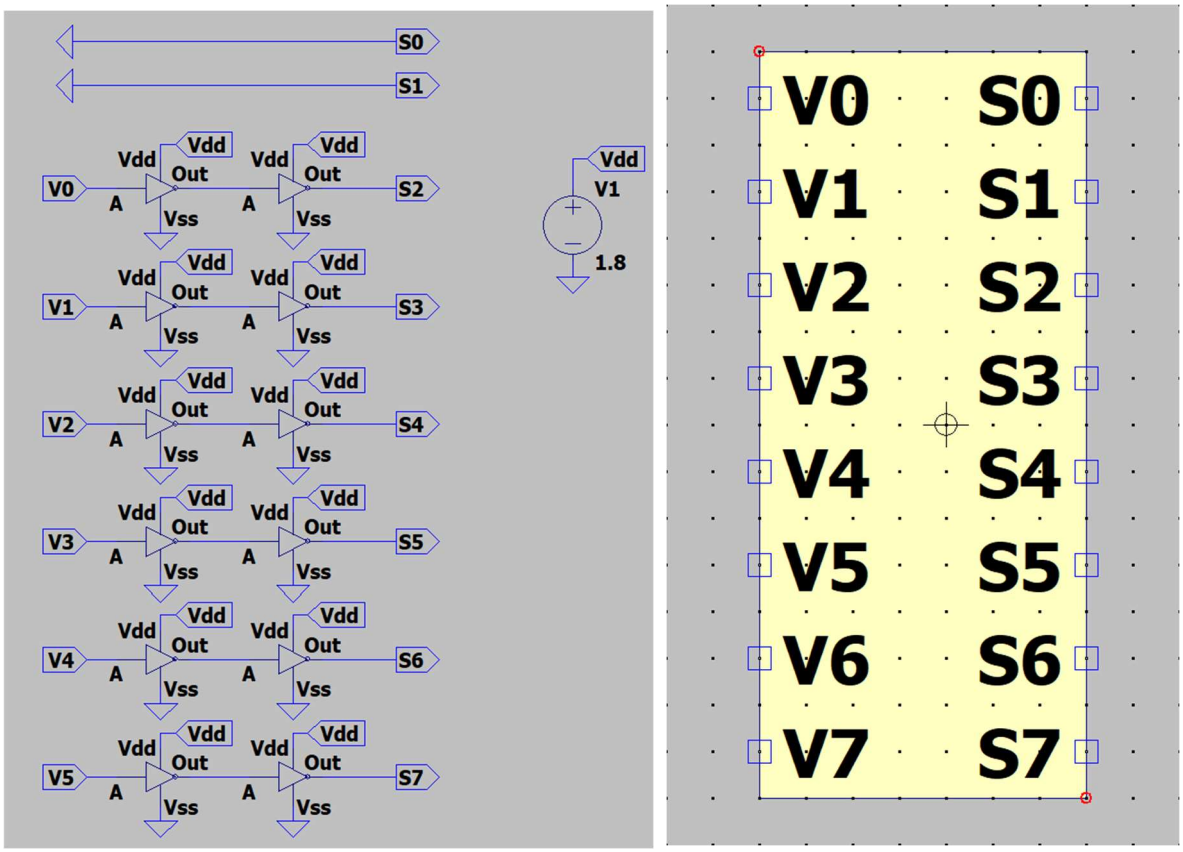


Figure 10: Shifter Block Circuit and Symbol

The Shifter block is used to perform a fixed left shift operation on the second partial product (PP1_value), shifting it by two bit positions. This is required because, in the Radix-4 Booth algorithm, each encoding step corresponds to two bits of significance in the final result. The shifting is implemented using direct wiring, where the lower two output bits are set to logic LOW (0), and the remaining bits are filled from the input with proper alignment.

Task4: Final Adder (8-bit Adder) (SUMMATION Block)

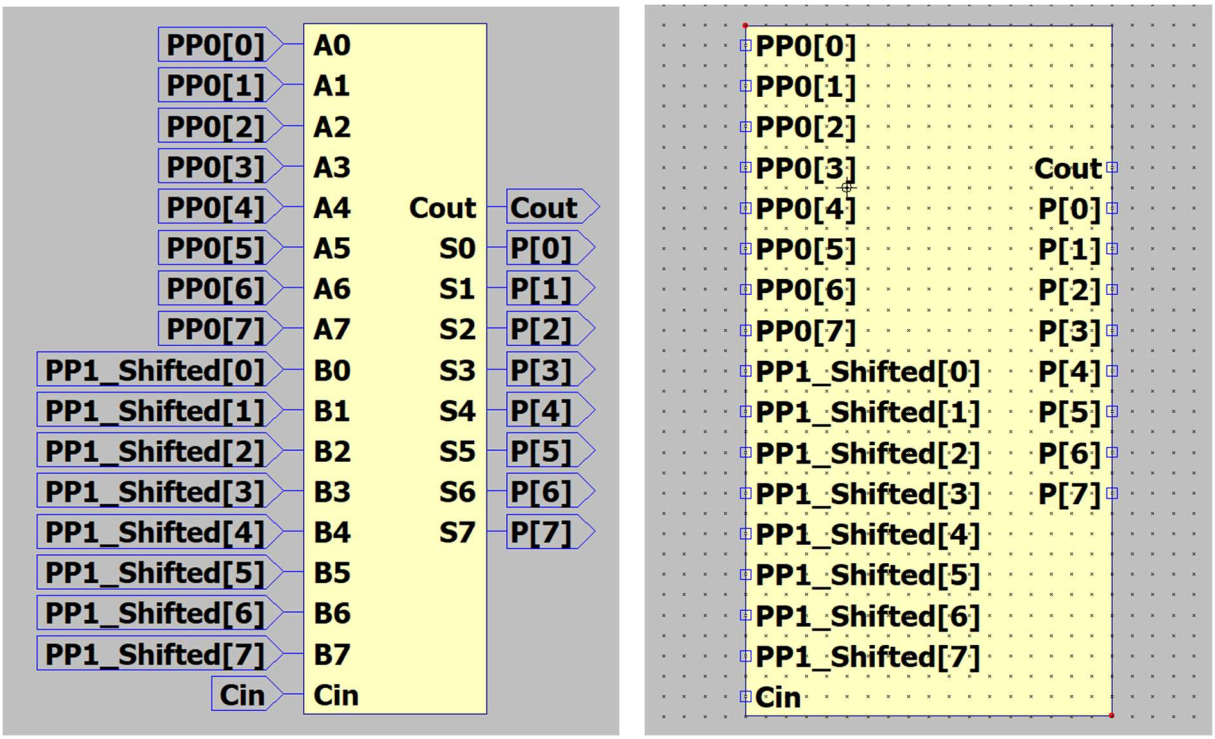


Figure 11: Summation Block Circuit and Symbol

The Summation block is used to compute the final 8-bit product by adding the outputs of the two partial product paths: PP0 and PP1_shifted. This addition is performed using an 8-bit adder, which constructed by 8 Bit Carry Lookahead Adder (8-bit CLA). The block ensures that the partial products are combined correctly to yield the final multiplication result.

FINAL DESIGN of SIGNED 4-BIT RADIX-4 BOOTH MULTIPLIER

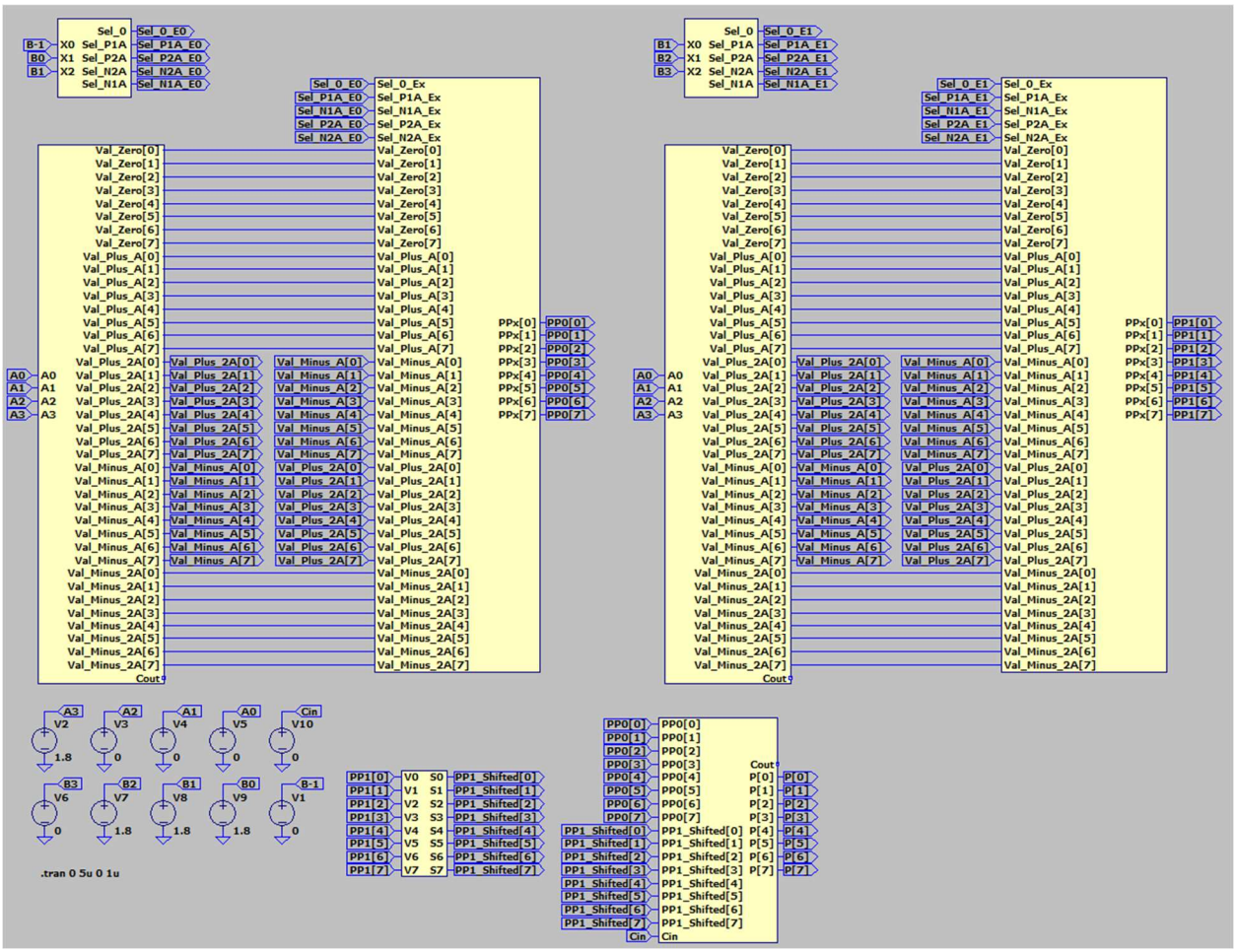


Figure 12: Final Design Circuit of Signed 4-bit Radix-4 Booth Multiplier

In the final design, all functional blocks required for a 4-bit signed Radix-4 Booth multiplier have been integrated. Two Booth Encoders are used to generate control signals from overlapping groups of the multiplier bits. These signals are fed into two Partial Product Selector (PPS) blocks, each of which selects one of five precomputed multiplicand multiples from the MMG block. The second partial product is then passed through a Shifter block, which performs a left shift by two bits. Finally, the two partial products (PP0 and PP1_shifted) are added using an 8-bit Summation block to produce the final 8-bit product output (P[7:0]). All modules are connected at the gate level and driven by test inputs for simulation and verification.

The calculated values for each step are presented in a table, and the simulation results are shown starting from the next page. You can find final design file in the zip file named as “FINAL_DESIGN_all_blocks_included_200207032_Nida_Mert”.

	A0	A1	A2	A3	B0	B1	B2	B3
Step 1	0V	1.8V	1.8V	0V	1.8V	1.8V	1.8V	0V
Step 2	0V	1.8V	1.8V	0V	1.8V	0V	0V	1.8V
Step 3	0V	0V	0V	1.8V	0V	0V	0V	1.8V
Step 4	0V	0V	0V	1.8V	1.8V	1.8V	1.8V	0V

	Encoder0	PP0	Encoder1	PP1	PP1_Shifted	P
Step 1	Sel_N1A_E0	11111010***(-6)***	Sel_P2A_E1	00001100***(+12)***	00110000***(+48)***	00101010***(+42)***
Step 2	Sel_P1A_E0	00000110***(+6)***	Sel_N2A_E1	11110100***(-12)***	11010000***(-48)***	11010110***(-42)***
Step 3	Sel_0_E0	00000000***(0)***	Sel_N2A_E1	00010000***(+16)***	01000000***(+64)***	01000000***(+64)***
Step 4	Sel_N1A_E0	00001000***(+8)***	Sel_P2A_E1	11110000***(-16)***	11000000***(-64)***	11001000***(-56)***

The tables are presented as expected value references for each simulation step, showing both the applied inputs and the anticipated outputs. In the first table, the input voltages applied to the multiplicand (A) and multiplier (B) bits are shown for each step. In the second table, the control signals selected by each Booth Encoder, the generated partial products (PP0 and PP1), the shifted result of PP1, and the final output product (P) are provided. All binary values are accompanied by their signed decimal equivalents to verify the correctness of the Booth multiplication at each step.

STEP 1 Simulation Results:

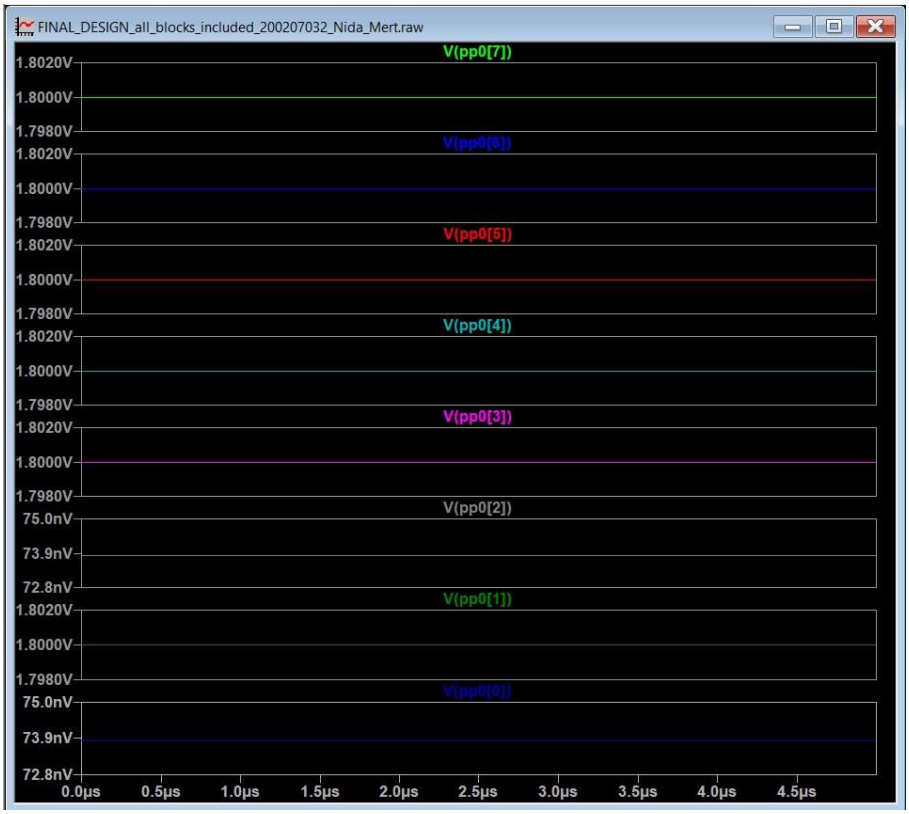


Figure 13: Step 1 PP0 Results

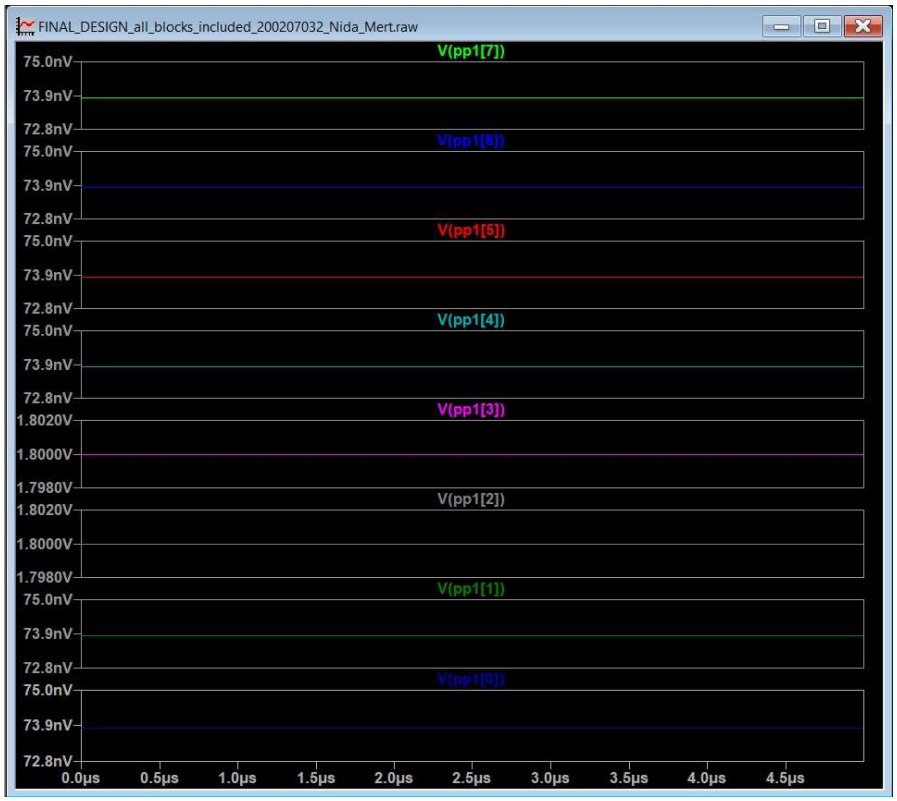


Figure 14: Step 1 PP1 Results

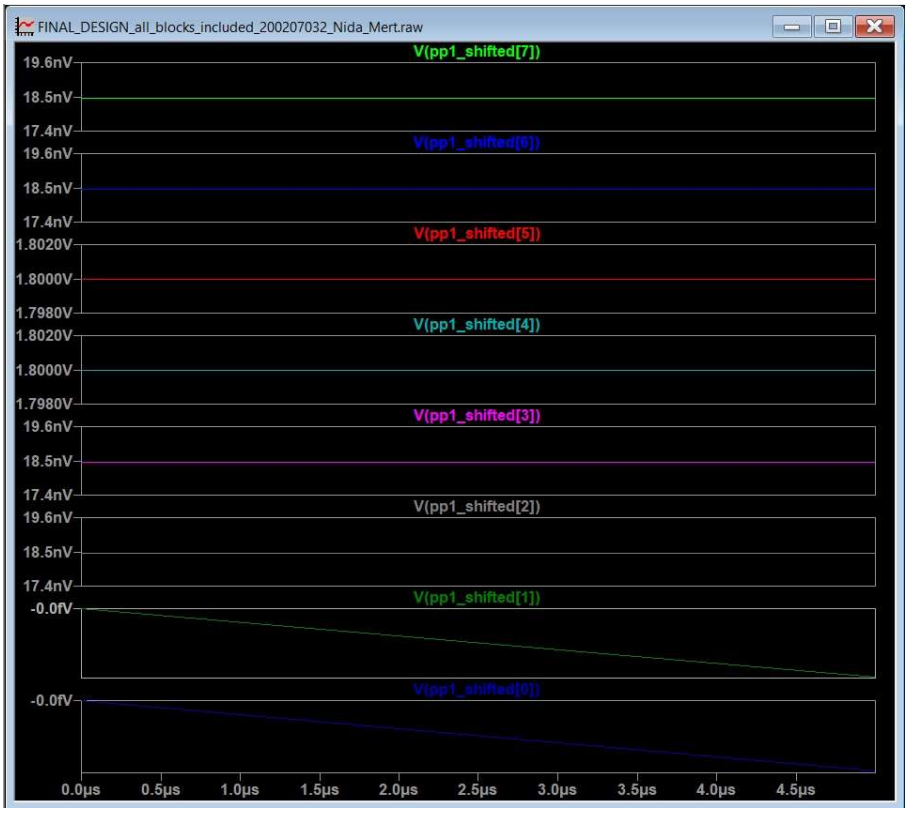


Figure 15: Step 1 PP1_Shifted Results

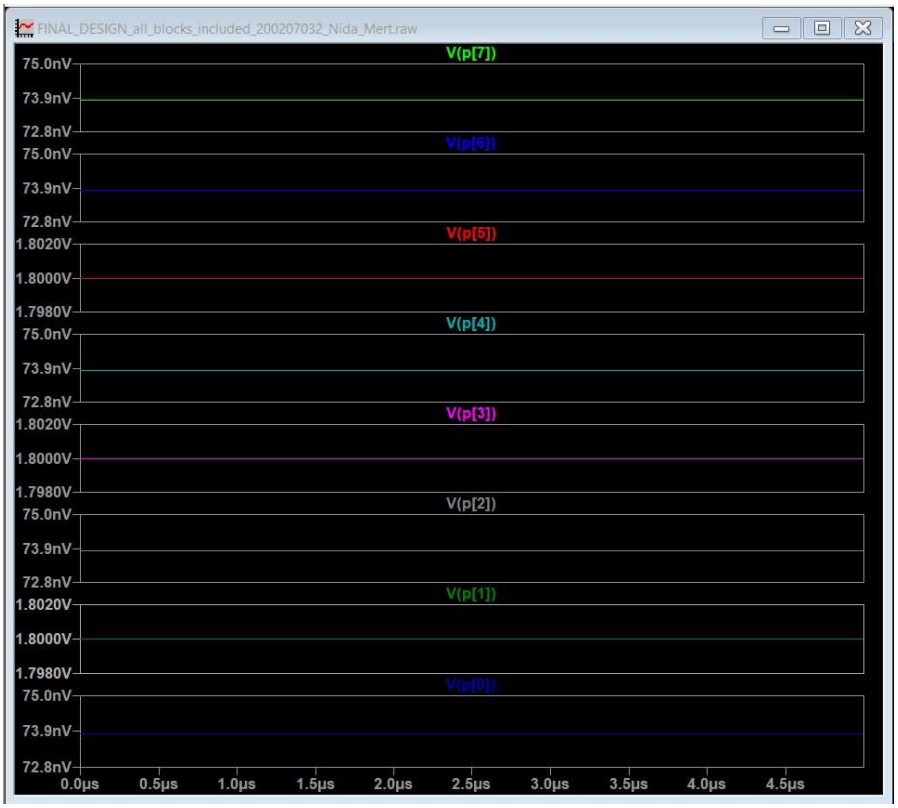


Figure 16: Step 1 P[i] Results

STEP 2 Simulation Results:

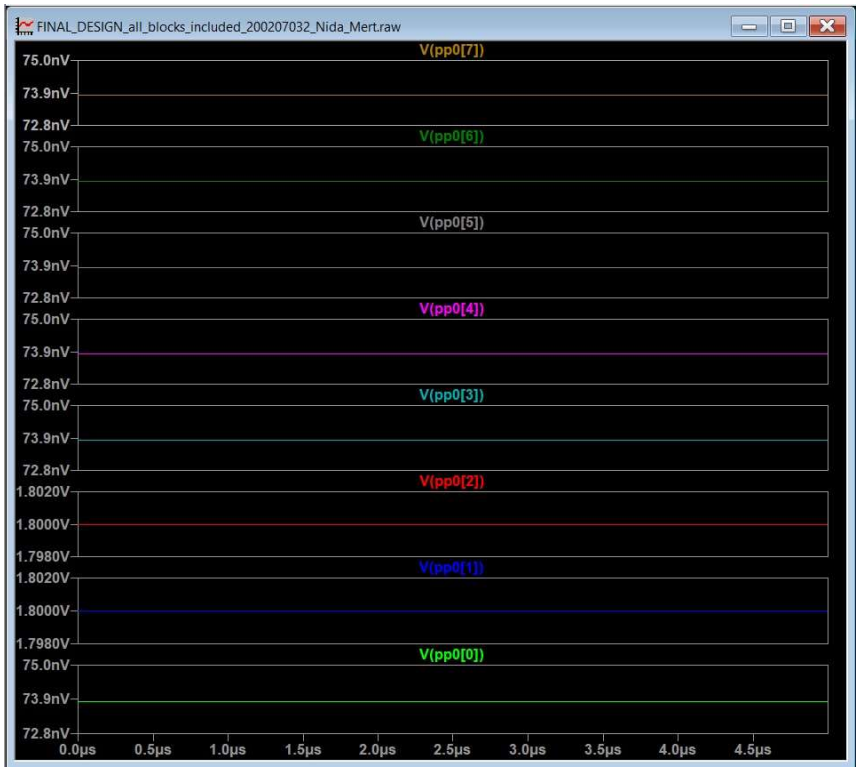


Figure 17: Step 2 PP0 Results

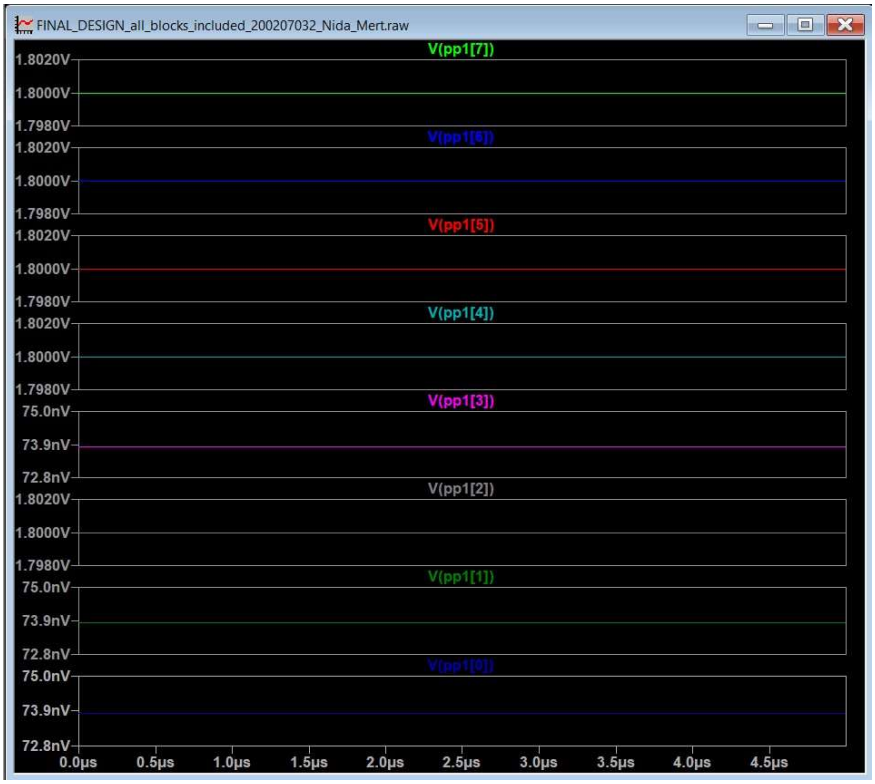


Figure 18: Step 2 PP1 Results

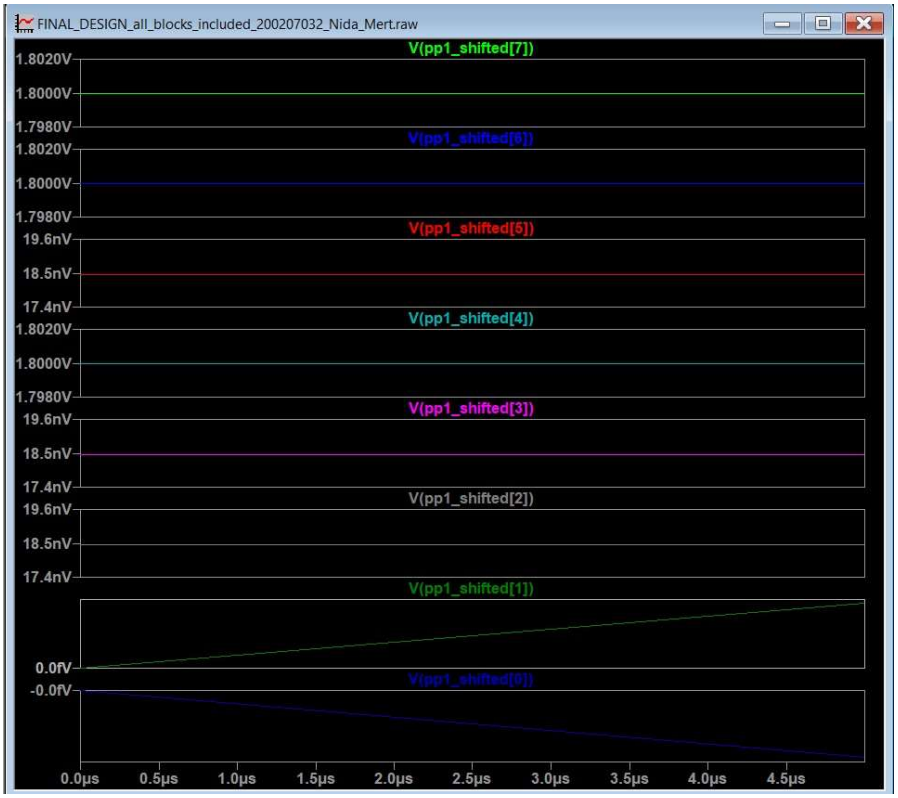


Figure 19: Step 2 PP1_Shifted Results

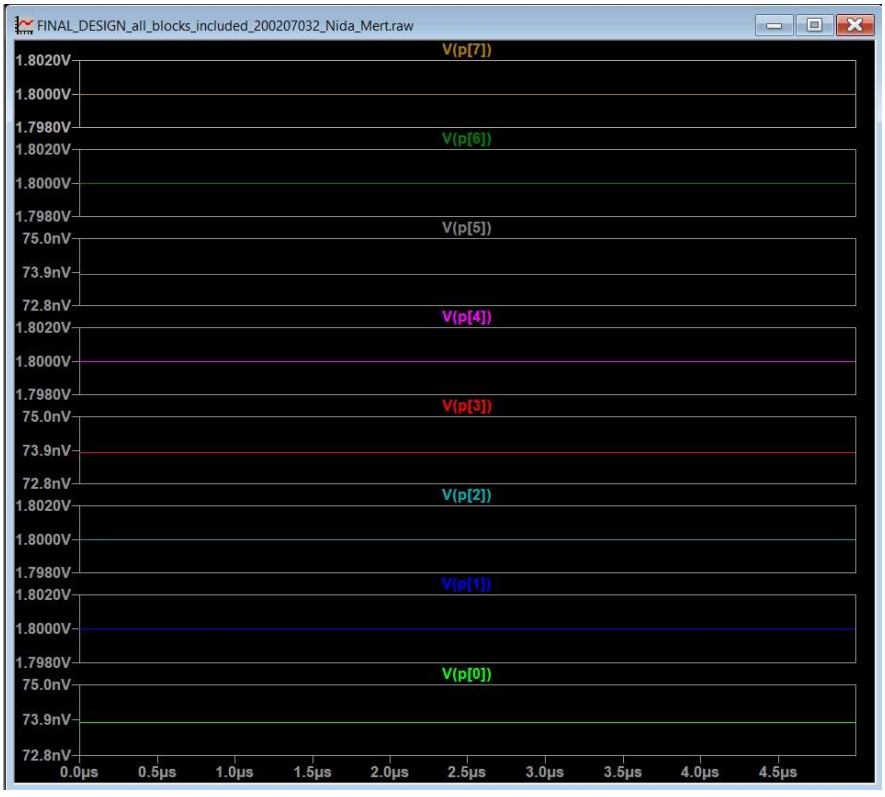


Figure 20: Step 2 P[i] Results

STEP 3 Simulation Results:

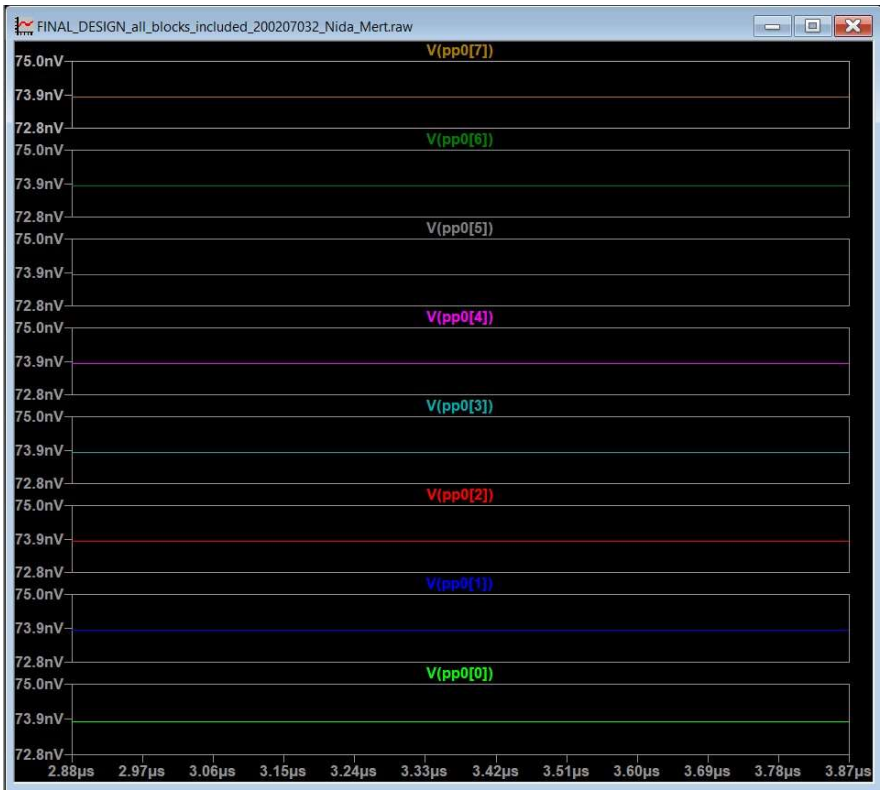


Figure 21: Step 3 PP0 Results

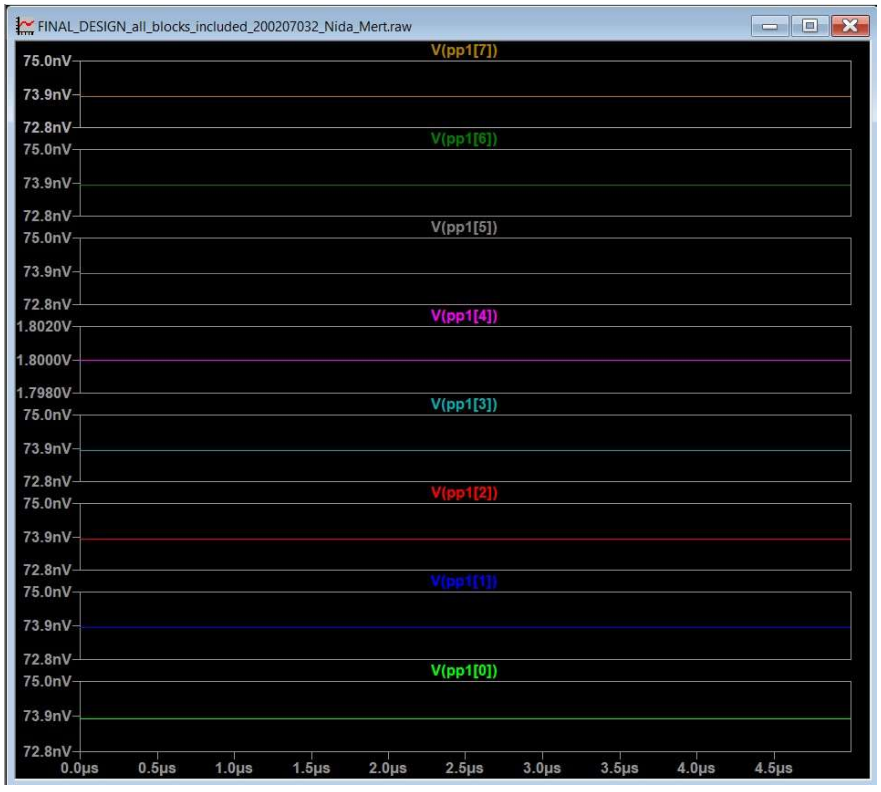


Figure 22: Step 3 PP1 Results

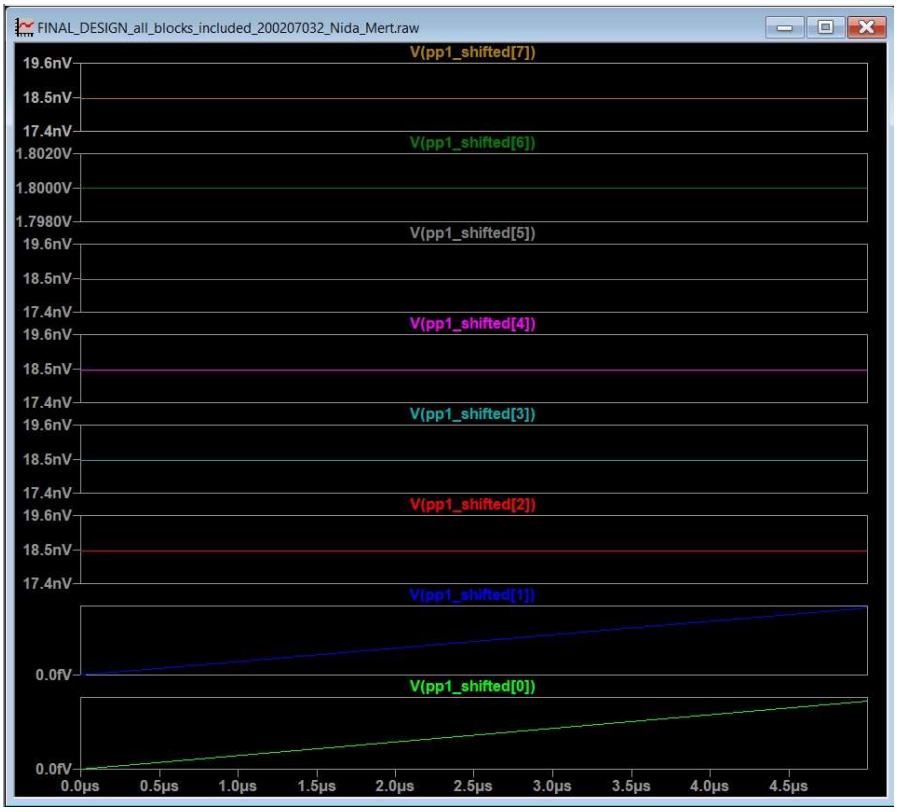


Figure 23: Step 3 PP1_Shifted Results

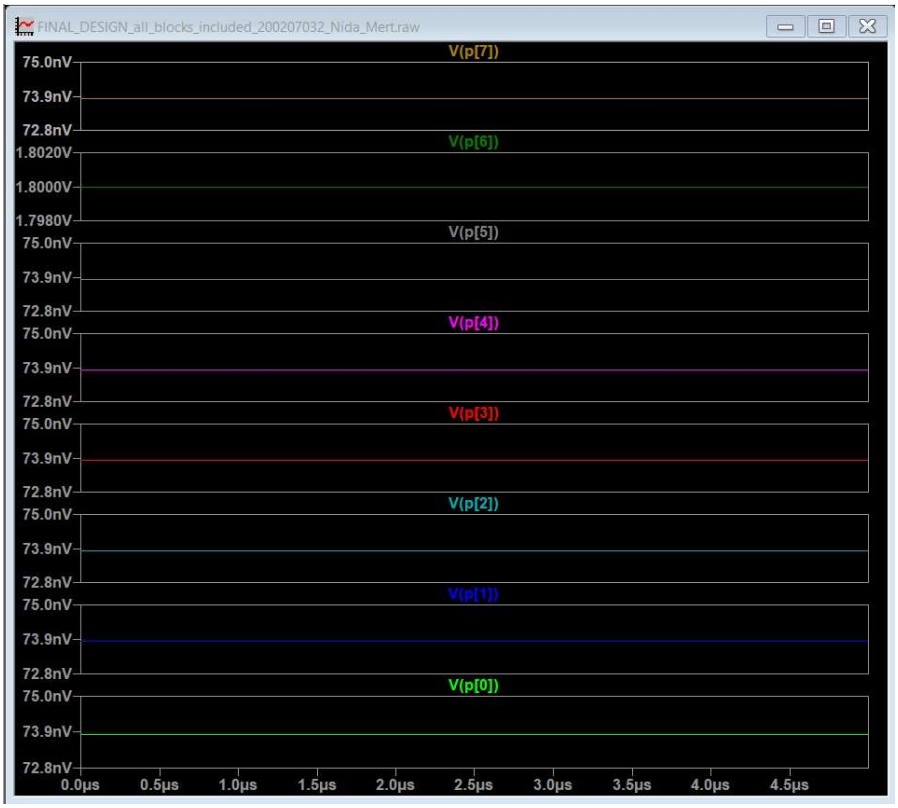


Figure 24: Step 3 P[i] Results

STEP 4 Simulation Results:

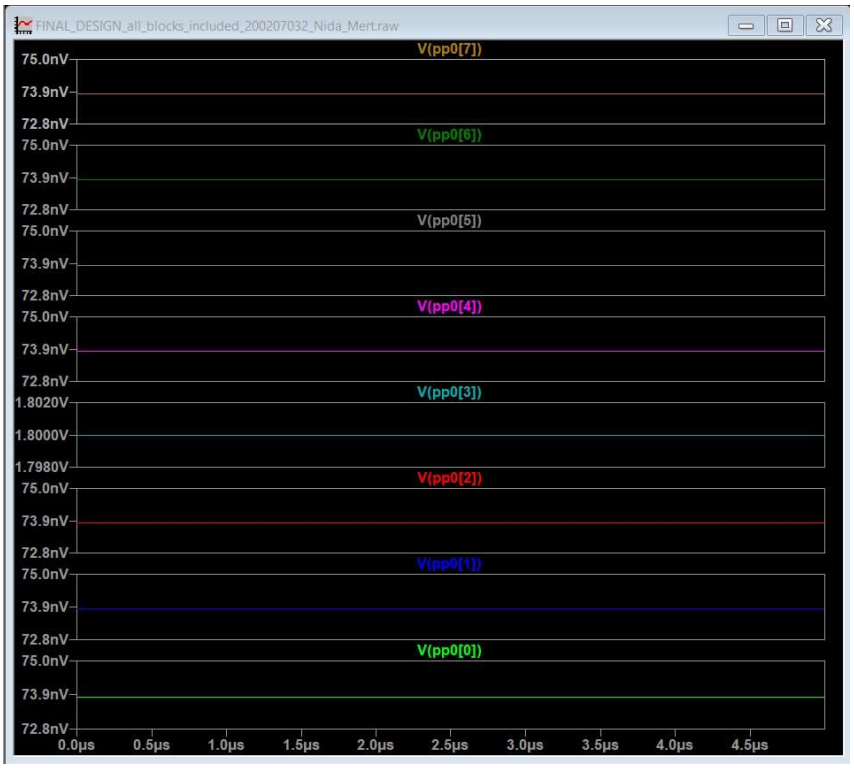


Figure 25: Step 4 PP0 Results

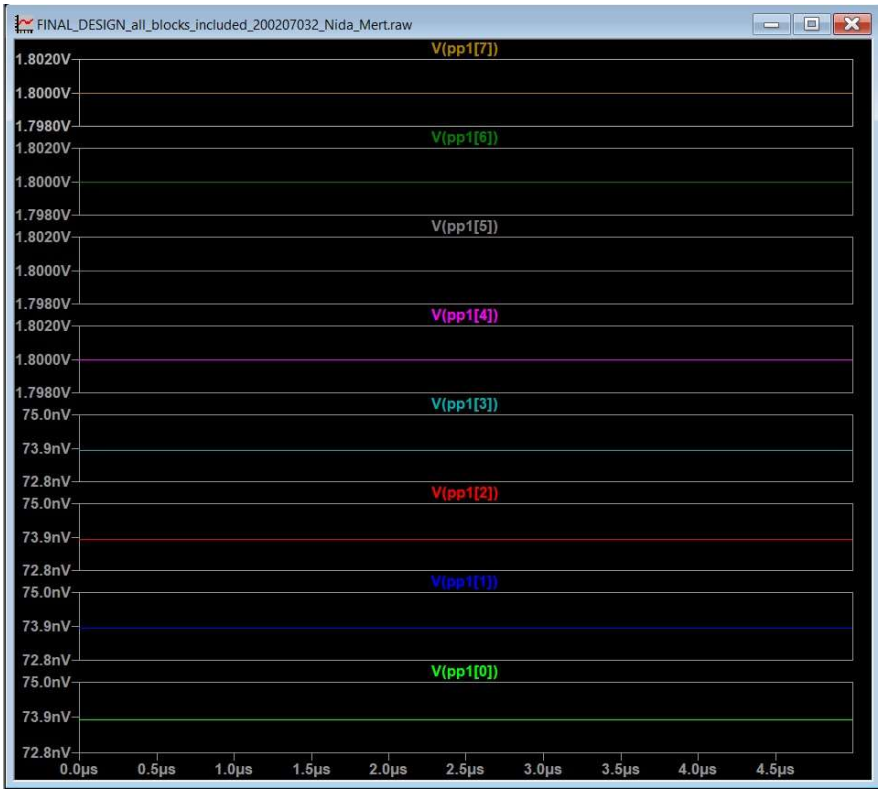


Figure 26: Step 4 PP1 Results

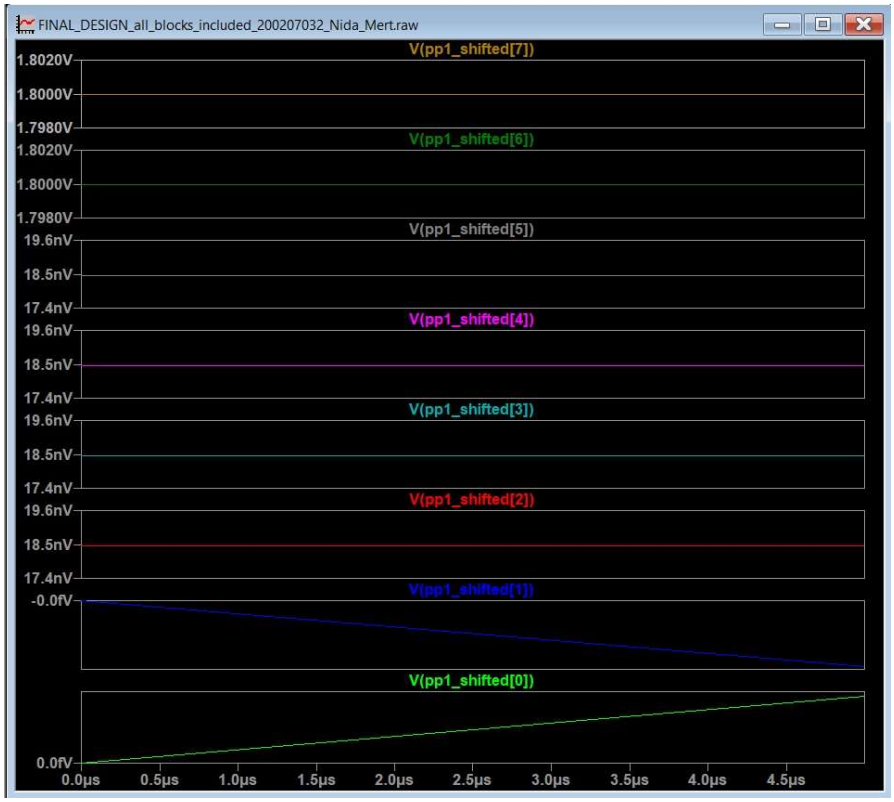


Figure 27: Step 4 PP1_Shifted Results

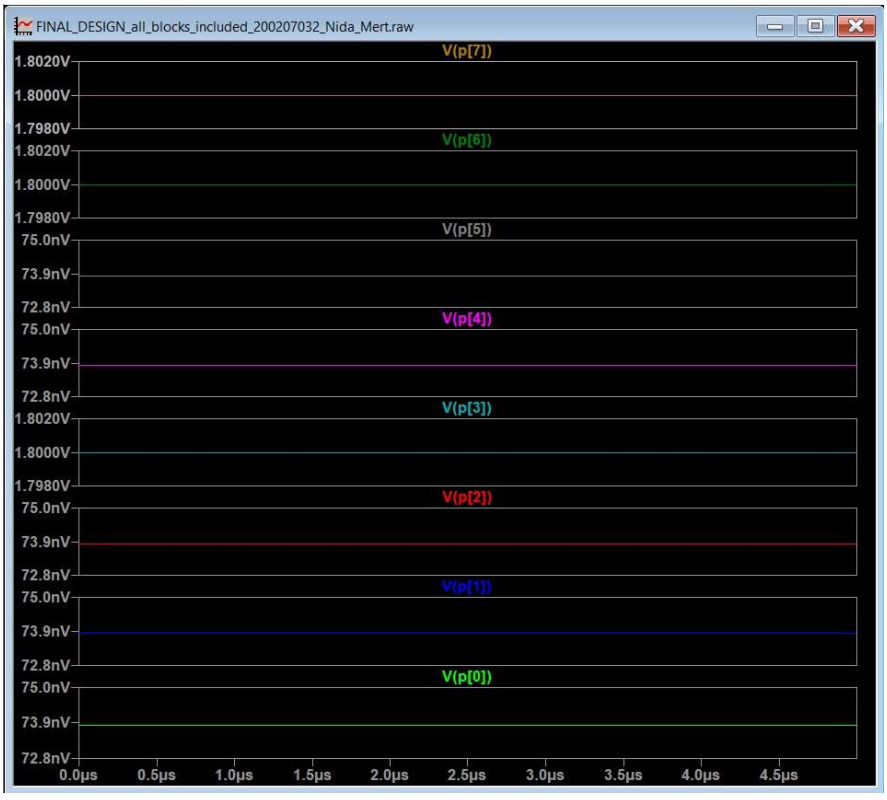


Figure 28: Step 4 P[i] Results

Conclusion:

The values of PP0, PP1, PP1_shifted, and the final product P were observed through simulation by applying the corresponding A and B input values for each step. As a result, the observed outputs matched the expected values provided in the reference table above. All blocks and the final design have been verified to function correctly.