This was the final project of my VLSI class during my senior year. Our group was tasked with designing and simulating a simplified ALU using a 45nm process (FreePDK45) in Cadence. The ALU required a 4-bit multiplier, logical operations (AND, OR, XOR, NOT), and arithmetic operations (add, subtract, multiply), and was to be optimized for power, area, and delay. I was tasked with creating the multiplier and overall ALU schematics and layouts.

For the multiplier, I chose to use combinational logic which contained 3 4-bit adders and 16 AND gates. The 4-bit adders were ripple-carry adders, and we chose this over a faster architecture like a carry look-ahead adder because the trade-off between the reduced delay and increased area and power consumption was not worth it. The inputs to the adders are delayed by AND gates and other adders, so additional logic that reduces the time of the addition using the inputs would be inefficient.

A computer screen shot of a game

Description automatically generated

*Multiplier Schematic*

I designed the layout to be as space efficient as possible and to have the inputs and outputs be easily accessible. I also tried to keep the overall shape as square as possible, which is common practice in layout design to reduce area and help it fit in higher level layouts. The layout ended up being 13.98 um long and 13.16 um wide.

A computer circuit board with many different colored lines

Description automatically generated

*Multiplier Layout*

The ALU contained 4-bit AND/OR/XOR/NOT logic, my 4-bit multiplier, a 4-bit adder/subtractor, select logic which determines which operation the add/subtract unit will perform, and a control unit. The control unit is made up of four 8x1 multiplexers that determine which output to select based on 3 given select bits. Each of the 4-bit logic/arithmetic units have a 4-bit output except for the multiplier, which has an 8-bit output. To account for this, the first four bits of the ALU output were fed from the control unit output, and the last four bits were taken directly from the multiplier. The user would need to decide if the last four bits of the output were useful based on whether or not the multiply operation was chosen.

A computer screen shot of a diagram

Description automatically generated

*ALU Schematic*

The ALU layout contained many sub-units and required careful routing and positioning to keep the design compact. One way I did this was by splitting up the 4-bit logic units and instead grouping them by each bit. This meant that, for example, the 0th bit output logic of the AND, OR, NOT, and XOR gates were grouped together and fed into the first multiplexer of the control unit. The layout ended up being 40.33 um long by 33.3 um wide, and from the post simulation results we determined the power consumption to be 140 uW and the critical-path propagation delay to be 750.02 ps.

A computer screen shot of a computer scheme

Description automatically generated

*ALU Layout*

Overall, I was happy with the design of our team’s ALU, but there were some improvements I would’ve liked to make if we had more time. One of those would’ve been adding flip-flops to the inputs and outputs to allow for state management through a clock. It also would’ve been nice to make the control unit and adder/subtractor layouts more square to reduce the area of the ALU layout. Depending on the use case, we also could’ve added additional logic that would set the 4 most significant bits to 0 if any operation other than multiply was selected, since multiply is the only operation with an 8-bit output.