John Gangemi
Alexander Holst
Raj Patel
TEAM RAJ
Friday, October 24, 2014
CMOS VLSI Lab # 5

Introduction & Background

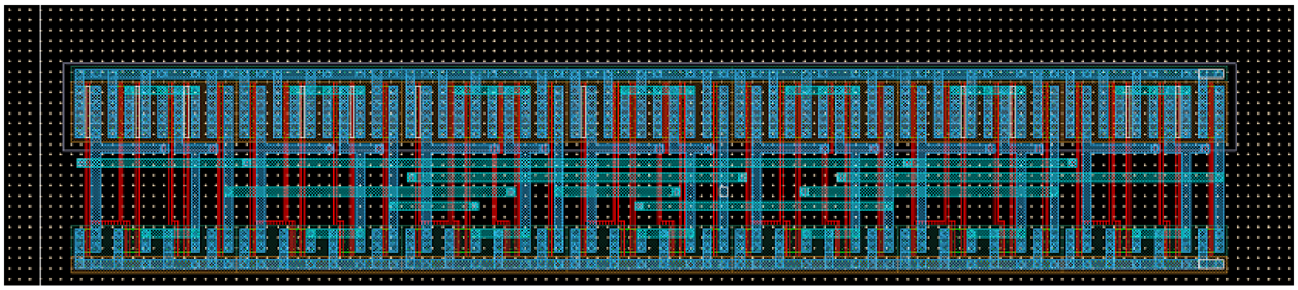
In this lab, our task was to create the CMOS layouts for a single-bit 8-to-1 multiplexer, an 8-bit adder-subtractor, and an 8-bit magnitude comparator. The idea behind the lab was to expand from one or two-bit circuits, which are easily exhaustively tested, to eight-bit circuits and more complicated combinational circuits, which cannot be exhaustively tested in any reasonable length of time. This meant applying and refining circuit designs made previously taking into consideration suitability for linking them together. Each circuit is described below.

Design & Layout

8-to-1 Multiplexer

A multiplexer is a circuit that takes some number of data lines and maps a single output line to one of those data lines using some input data to select which to map. An 8-to-1 multiplexer takes eight data lines and produces one output, using three select bits to choose the mapping. This can be easily accomplished by cascading seven instances of a simpler 2-to-1 multiplexer, four into two into one, and using one select bit for each layer. However, reductions in bounding area and propagation delay can be had by building the circuit from the ground up using basic gates.

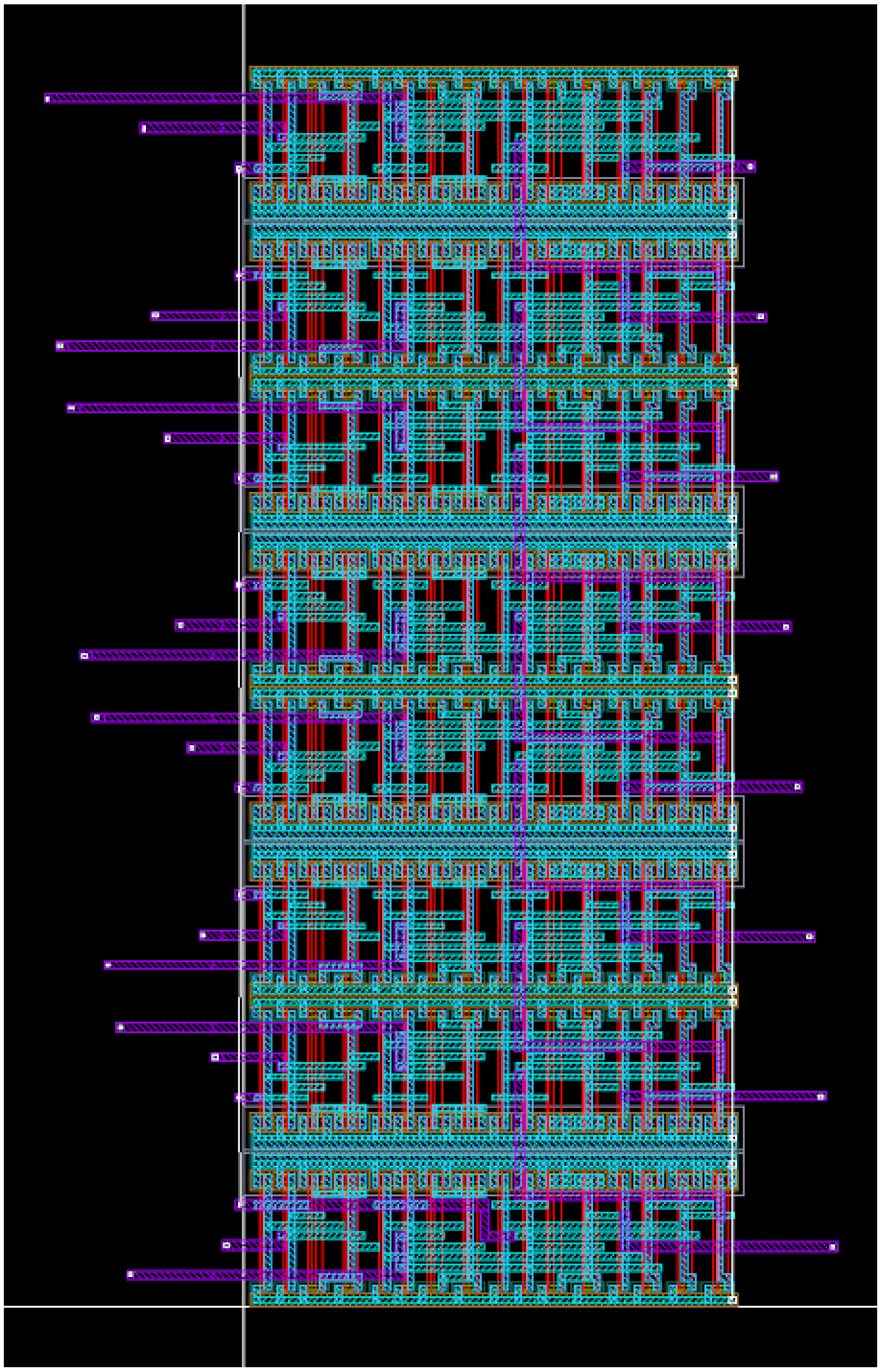
We chose to create a custom design in order to emphasize speed. Our bounding area for this design was 170.4 x 30.3 microns. The layout is as follows:



8-bit Adder/Subtractor

An adder is a circuit that takes two binary numbers, each of size (in bits) *n*, and calculates a sum of size *n + 1*, which depending on implementation may be truncated to size *n*. A subtractor is a similar circuit that takes two size *n* binary numbers and calculates the difference of the two numbers. An adder/subtractor is a circuit that includes both adder and subtractor and can switch between the two functions using a one-bit select line. Creating an 8-bit adder/subtractor required cascading eight instances of the bit-slice adder/subtractor we created before so that the carry/borrow line of each lower significance slice is fed into the next highest slice. We decided to ignore the final carry out, giving us an output of size *n*, same as the inputs. Our implementation requires that, in addition to supplying the same select line to each bit slice, we also supply the select line as the first carry in so that addition carries in a one and subtraction carries in a zero, as we are using two's complement to implement subtraction.

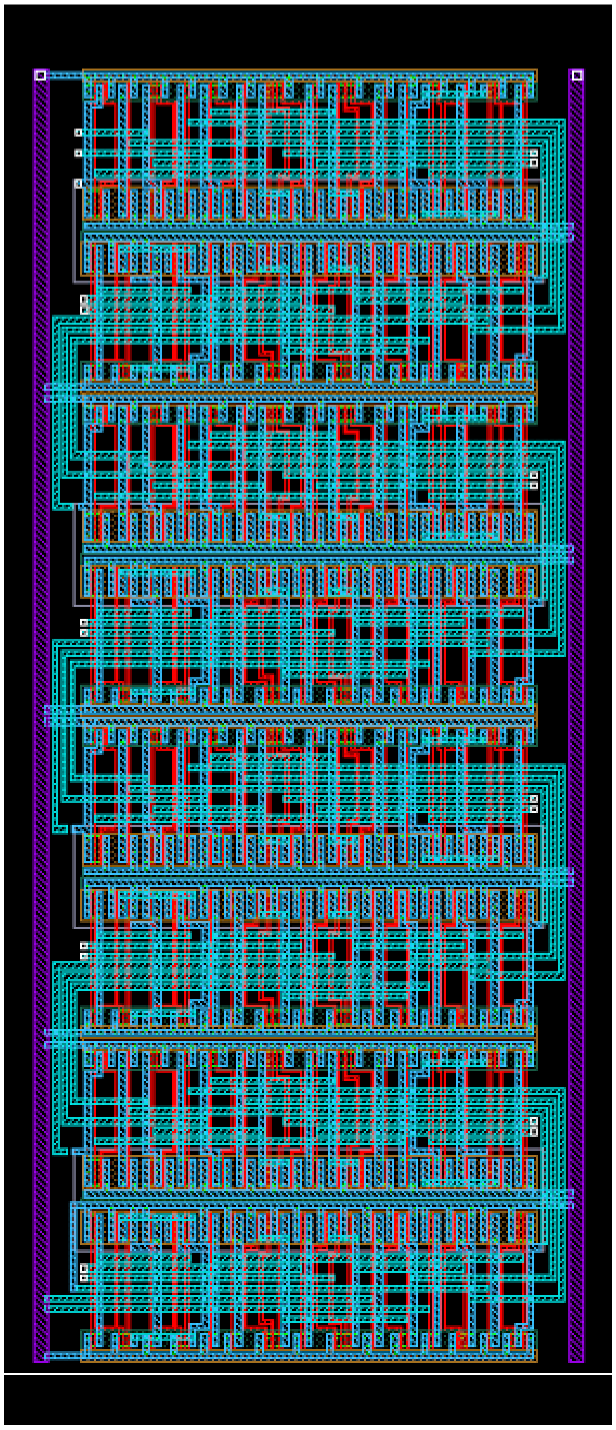
We routed the inputs and outputs from one bit-slice to the next using the top metal layer. Our bounding area for the design was roughly 250 microns in width by 100 microns in length. The layout is as follows:



8-bit Magnitude Comparator

A magnitude comparator is a circuit that takes two binary numbers *A* and *B* and produces some output that calculates whether *A* is greater than, less than, or equal to *B*. Creating an 8-bit magnitude comparator required cascading eight instances of the bit-slice magnitude comparator we created before so that the output of higher-significance bit-slices is fed as input to the next lowest slice. This is because calculation of comparison for any one bit is dependent on the result for the next highest bit, as higher-significance bits overrule lower-significance bits. The output of the lowest-significant bit serves as the output of the circuit. The circuit will have three output bits, one to mark $A < B$, one to mark $A > B$, and one to mark $A = B$. One and only one of these output bits must be asserted at any time.

Like with the adder/subtractor, we chained the inputs and outputs for bit-slices of the magnitude comparator and observed the output at the lowest significant bit-slice. Our bounding area for the design was 113.4 X 266.4 . The layout is as follows:



Testing & Results

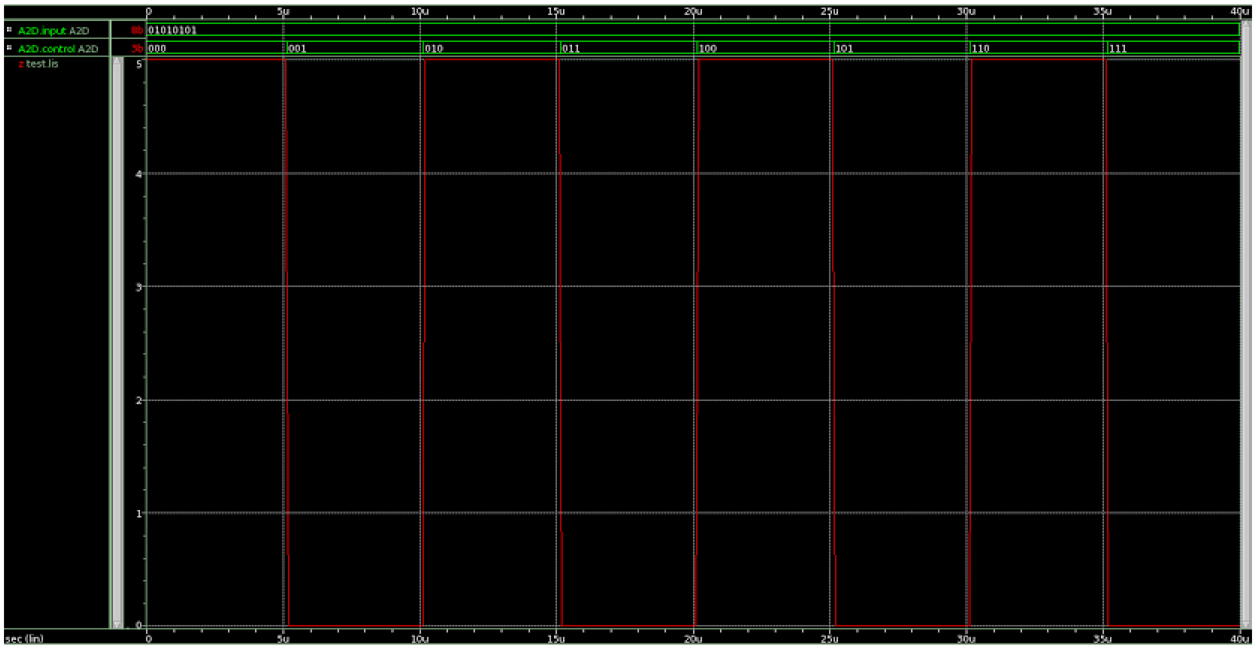
With simpler one, two, and three-bit combinational circuits we were able to perform exhaustive testing, as there were only a few inputs to worry about. We could conceivably exhaustively test maybe up to six-input circuits, but beyond is a lot of time and data invested unnecessarily. Now, with seventeen inputs to the adder/subtractor, a total of 131072 combinations, we cannot test every combination, so we limit ourselves to demonstrating correctness using some edge cases where overflow and underflow manifest, and using typical operating cases with solid numbers.

For uniformity, we use the same suite of test inputs for all of our circuits. For the adder/subtractor, we repeated the test for each function. For the multiplexer, we set the data inputs to one of our A tests and ran through all select inputs. We also decided that, since we are working with larger ripple delays, we would simulate in microseconds rather than nanoseconds. Outputs are, as always, capacitively loaded. Our test suite is shown below:

A	B	Expected add/sub	Expected mag comp
11111111	11111111	11111110/00000000	A=B
11110000	11110111	11100111/11111001	A<B
00000000	11111111	11111111/00000001	A<B
00000000	00000000	00000000/00000000	A=B
01010101	11111111	01010100/01010110	A<B
11111111	00000000	11111111/11111111	A>B
11001100	11001100	10011000/00000000	A=B
00001111	11110000	11111111/00011111	A<B

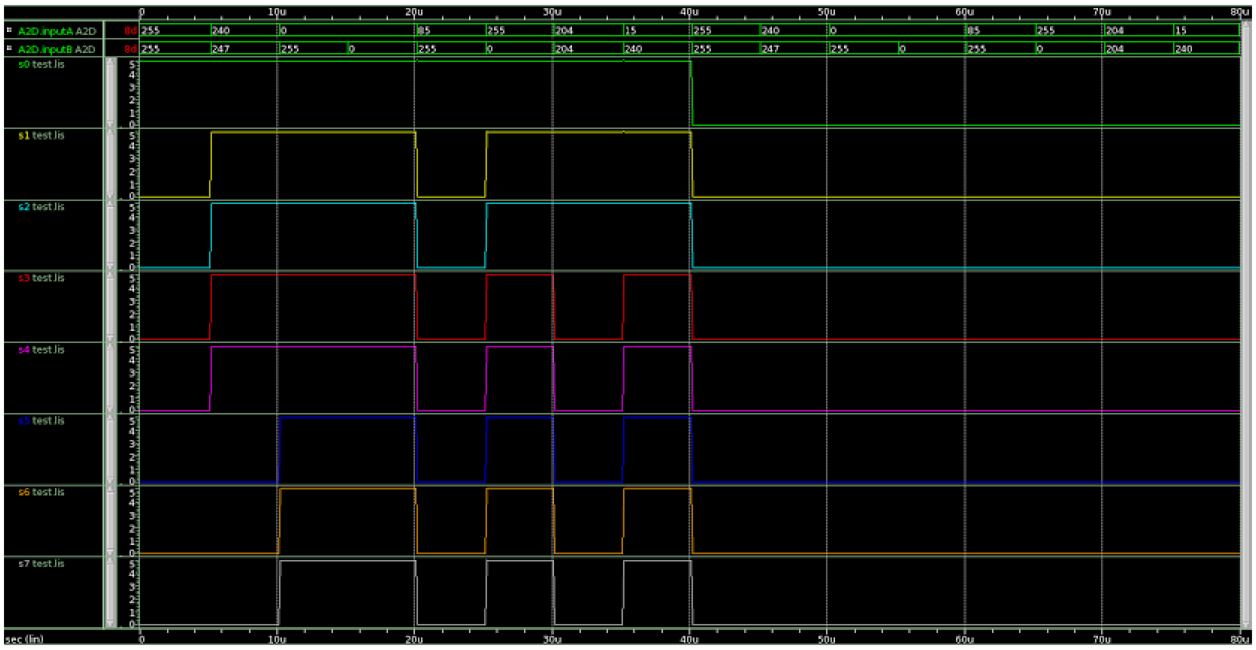
8-to-1 Multiplexer

For the multiplexer, we just used one of our A inputs (01010101) from the table and ran through all the select combinations. If we see the same A input reflected in the output, we can be pretty sure our multiplexer is working. Output is shown below. As we can see, this is exactly what happens:



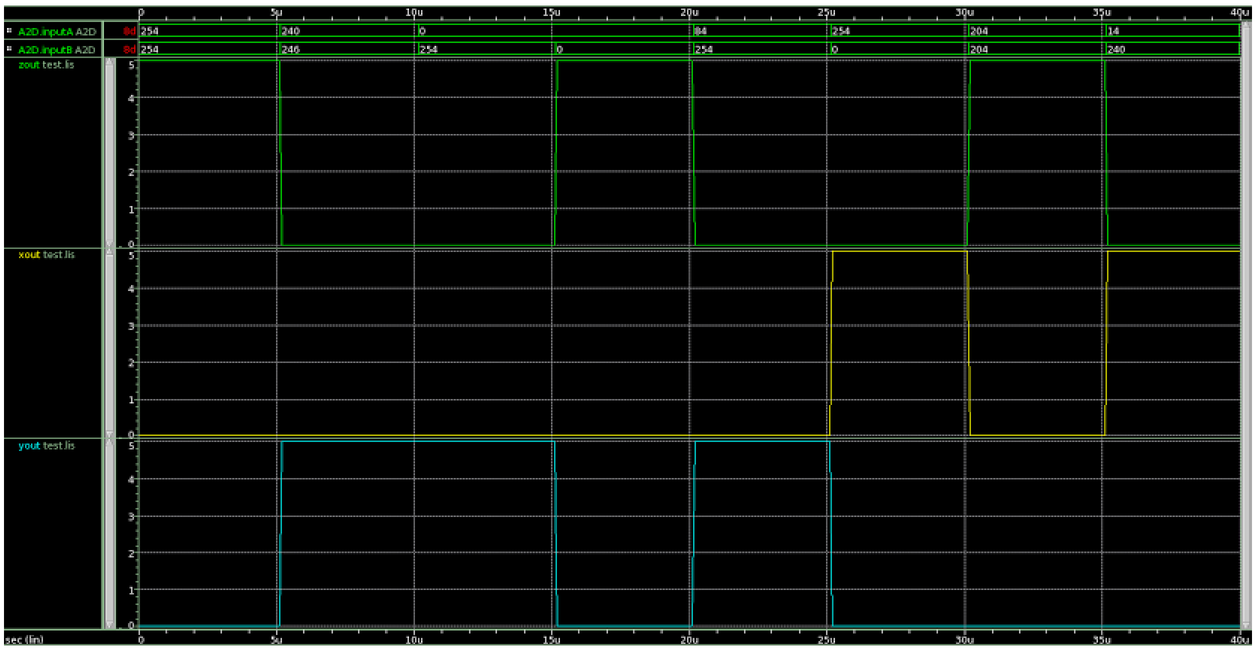
8-bit Adder/Subtractor

For the adder/subtractor, we used vectored input to run through eight combinations for each function, a total of sixteen inputs. We chose edge cases and typical cases to show that, in general, we can be confident our system works. Output is shown below:



8-bit Magnitude Comparator

For the magnitude comparator, we used the same vectored input to run through a test suite to show a general working of our comparator. Output is shown below:



For the magnitude comparator the values are as showcased.
X is when a >b
Y is when a<b
Z is when a=b

Conclusion & Feedback

For the 8-to-1 multiplexer, we were pleasantly surprised by the result. We could implement the additional functionality for only a very, very slight addition in propagation delay. We consider the 8-to-1 multiplexer a success.

For the 8-bit adder/subtractor, we were successful in stacking our bit-slice and getting a solid output. We did notice that as we are working with the full ripple propagation delay of eight bits in the carry, we had some large rise and fall times when using nanoseconds timing, as well as some glitches. Switching to microsecond timing made the outputs more smooth. However, for reasons as yet unknown, the output does not respond as we intended. We know for certain through exhaustive testing that each bit-slice for the adder/subtractor that the implementation and layout is correct and responds to inputs as we expect. We have also checked that carry inputs and outputs are being tied to the correct pins and have made sure that global voltage and ground pins are properly established. This is evidenced by the strength of the outputs, which are at solid 5.0V and 0.0V. We think one possibility is that glitches at lower significant bits are being buffered and sent up the chain where outputs are being prevented from switching as normal. Otherwise, we plan to continue investigating where the problem lies. We consider this part a successful failure in that we showed some switching behavior, but unexplained incorrect behavior.

For the 8-bit magnitude comparator, we were likewise successful in stacking our bit-slice and getting a solid output. The same delays we were noticing with full ripple propagation at nanosecond timing were eliminated with microsecond timing. Our outputs performed as expected, and we consider the comparator a success.

Otherwise, the lab generally went as smoothly as the previous labs.