To add or subtract:

Full Adder

To multiply or divide, use same ripple carry adder for

\[ OV \to \text{test for overflow} \]

\[ OV = c_{in-1} \]

\[ c_{in} \text{ is a control bit which says to add or subtract} \]

\[ c_{in} = 1 \text{ - subtract/divide} \]

\[ c_{in} = 0 \text{ - add} \]

* \( c_5 \) chooses whether it is

\[ c_5 = 1 \text{ - add/subtract} \]

\[ c_5 = 0 \text{ - multiply/divide} \]

* \( c_4 = 1 \text{, } c_{in} = 1 \text{ divide} \)

* \( c_4 = 1 \text{, } c_{in} = 0 \text{ multiply} \)

* \( c_3 = 0 \text{, } c_{in} = 1 \text{ subtract} \)

* \( c_3 = 0 \text{, } c_{in} = 0 \text{ add} \)

Outputs: \( OV \Rightarrow \) overflow

\( (z_2, z_1, z_0) \)
In Addition PLA comparator

\[ S_1 = x_1'y_1 \quad E_1 = (x_1 \oplus y_1) \quad G_1 = x_2'y_2 \]

\[ \begin{align*}
S_2 &= E_1'y_1 + E_1y_1 + E_2'E_0'G_0' + S_2 = b \quad \text{for the sign bit} \\
E_2' &= E_2E_0G_0 \\
S_2 &= E_2'S_2 + E_2E_0G_0 + G_2
\end{align*} \]

\[ z_E = (x_2 \oplus y_2') (y_1 \oplus y_1') (x_0 \oplus y_0) \text{cin} \]

\[ z_E = (x_2y_2 + x_2'y_2) (x_1y_1 + y_1x_1') (x_0y_0 + y_0x_0) \text{cin} \]

\[ \begin{align*}
z_S &= (y_2'y_2 + x_2'y_2) (x_1'y_1) + \\
z_G &= (x_0y_0 + y_0x_0) (x_1'y_1) + (x_2'y_2x_2')
\end{align*} \]

- Simplify the equations until you have 1 AND gate level and 1 OR gate level (SOP)

OR A

 PLA

Ram comparator

* Blow circuit on both levels
\[ \begin{array}{ccc}
A \text{ code} & a_2, a_1, a_0 & B \text{ code} & b_3, b_2, b_1, b_0 \\
0 & 0000 & 0 & 0000 \\
1 & 0011 & 7 & 0111 \\
2 & 0110 & 14 & 1110 \\
3 & 1001 & 5 & 0101 \\
4 & 1100 & 12 & 1100 \\
5 & 1111 & 3 & 0011 \\
6 & 0010 & 10 & 1010 \\
7 & 0101 & 1 & 0001 \\
8 & 1000 & 8 & 1000 \\
9 & 1011 & 15 & 1111 \\
10 & 1110 & 6 & 0110 \\
11 & 0001 & 13 & 1101 \\
12 & 0100 & 4 & 0100 \\
13 & 0111 & 11 & 1011 \\
14 & 1010 & 2 & 0010 \\
15 & 1101 & 9 & 1001 \\
\end{array} \]

(a)

\[ a_1 \rightarrow b_1 \]
\[ a_0 \rightarrow b_0 \]
\[ a_2 \rightarrow b_3 \]

(b)

Use truth table to connect a code directly to \( b_3 \) code
\[ b_2 \]
\[ b_1 \]
\[ b_0 \]
I have learned how to analyze and build circuits using combinational and sequential logic and standard modules. Some combinational logic, which is time independent, meaning there's no memory, we used were boolean algebra, K-maps, and truth tables to build circuits. Then we analyzed the time delays of circuits, time complexities, and space, and the optimization of gates to minimize both. Real world apps include, full adders, ALUs, and more.

Sequential logic, which depends on past events requires the use of flip-flops and/or registers to "remember." D-FF, SR, JK, and T Flip-Flops are some of the memory cells we used. Using FF we can build registers.

We use state tables, diagrams, to program combinational logic with registers.

Some applications we apply to sequential circuits are like how soda machines, door openers, just about anything that has time or frequency dependence. We can build standard modules like counters, shift registers, and combinational modules like muxes, demuxes, decoders, and their tree/coincident networks.

I also learned how to research other sources since textbook is seriously lacking and how to teach at University level.