So far

Levels of transformation:

- Problem statement
- Algorithm
- Program
- Instruction Set Architecture
- Microarchitecture
- Logic circuits
- Devices

Function representation
Binary representation
DESIGN OF ADDERS AND ARITHMETIC UNITS

Binary addition

Let's consider the addition of two $N$-bit numbers:

\[ \begin{array}{cccccc}
C_n & C_{n-1} & \ldots & C_2 & C_1 \\
A_{n-1} & \ldots & A_2 & A_1 & A_0 & + \\
& b_{n-1} & \ldots & b_2 & b_1 & b_0 \\
\hline
& S_{n-1} & \ldots & S_2 & S_1 & S_0
\end{array} \]

Carry (C)
First operand (A)
Second operand (B)
Summation (S)

Remember: fixed-width calculations
1-bit half-adder (HA)

Truth table:

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>S</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Implementation:

Symbol:
1-bit full adder (FA)

Truth table:

<table>
<thead>
<tr>
<th>Cin</th>
<th>a</th>
<th>b</th>
<th>S</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Symbol:

Using 3-variable K-maps:

<table>
<thead>
<tr>
<th>S</th>
<th>a</th>
<th>b</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cin</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ S = \]

<table>
<thead>
<tr>
<th>Cout</th>
<th>a</th>
<th>b</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cin</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ Cout = \]
Implementation:

![Adder Circuit Diagram]

\[ \text{Cin} \quad \text{a} \quad \text{b} \quad \text{S} \quad \text{Cout} \]
Ripple carry adder

Consider how to add $n$-bit numbers, e.g., 3-bit numbers:

\[
\begin{array}{ccc}
\alpha_2 & b_2 & C_3 \\
\alpha_1 & b_1 & C_2 \\
\alpha_0 & b_0 & C_1 \\
\end{array}
\]

\[
\begin{array}{ccc}
C_{out} & FA & C_{in} \\
S & S & S \\
S_2 & S_1 & S_0 \\
\end{array}
\]

Question: how do we subtract in 2's complement?

Subtraction using ripple carry adder:

\[
\begin{array}{ccc}
\alpha_2 & b_2 & C_3 \\
\alpha_1 & b_1 & C_2 \\
\alpha_0 & b_0 & C_1 \\
\end{array}
\]

\[
\begin{array}{ccc}
C_{out} & FA & C_{in} \\
S & S & S \\
S_2 & S_1 & S_0 \\
\end{array}
\]

© 2017-2018 Juan Jose Jaramillo. All rights reserved.
Question: can we combine both designs in a single circuit?

Combined adder/subtractor:

```
\[ \begin{array}{c}
C_3 \quad \text{FA} \quad C_2 \quad \text{FA} \quad C_1 \quad \text{FA} \\
\text{S} \quad \text{S} \quad \text{S}
\end{array} \]
```

Question: how do we detect overflow in 2's complement addition/subtraction?
Final circuit:
We can extend the 2’s complement adder/subtractor to perform more functions by adding more control inputs.

**Example:**

<table>
<thead>
<tr>
<th>$K_1$</th>
<th>$K_0$</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>$A + B$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>$A - B = A + B' + 1$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$A$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$B$</td>
</tr>
</tbody>
</table>
General structure of modified network: