CC are circuits *without* memory where the outputs are obtained from the inputs only. A $n$-input $m$-output combinational circuit is of the form

$$\text{Combinational Circuit}$$

where, $o_i = f(i_1, \ldots, i_n), \ 1 \leq i \leq m$.

- Half/Full adder/subtractor
- Binary ripple adder/subtractor
- Look-ahead carry adder
- Code converter, Encoder/Decoder
- Magnitude comparator
- Multiplexer/Demultiplexer
- Read-only memory
- Programmable logic array
- Programmable array logic
- Arithmetic logic unit
ADDERS AND SUBTRACTORS

**Half Adder** Adds 2 bits and outputs carry and sum

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$c$</th>
<th>$s$</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>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>


**Full Adder** Adds previous carry and 2 bits and outputs new carry and sum

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$z$</th>
<th>$c$</th>
<th>$s$</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>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>


**K-map for $c$**

<table>
<thead>
<tr>
<th>$x$ \ $yz$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>


**K-map for $s$**

<table>
<thead>
<tr>
<th>$x$ \ $yz$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Binary Ripple Carry Adder

Adds 2 \( n \)-bit numbers and outputs carry-out and sum

\[ \begin{align*}
&\text{Input carry of } FA_i = \text{output carry of } FA_{i-1} \\
&\text{Simple, easy and } cheap \text{ design} \\
&\text{Reusable} \\
&\text{Very slow} \\
&\text{Can you design a faster } n \text{-bit adder circuit?} \\
&\text{Can you design a smaller } n \text{-bit adder circuit?}
\end{align*} \]
Carry Lookahead Adder

Faster than BRCA but more complex

\[ FA = \text{Partial Full Adder} + \text{Ripple Carry Path} \]

So for a 4-bit BRCA we have 8 gate delays in the RCP

A solution to BRCA problem is to replace the RCP by a two-level circuit (2 gate delays)

**How:** Each \( C_i \) is a function of \( G_{i-1} \), \( P_{i-1} \) and \( C_{i-1} \), that is \( C_i = f(G_{i-1}, P_{i-1}, C_{i-1}) = G_{i-1} + P_{i-1}C_{i-1} \).

Generate a SOP expression for each \( C_i \) and obtain its two-level circuit implementation.
Carry Lookahead Adder (continued)

\[ C_i = G_{i-1} + P_{i-1}C_{i-1} \text{ for } 1 \leq i \leq n \]

\[
\begin{align*}
C_1 &= G_0 + P_0C_0 \\
C_2 &= G_1 + P_1C_1 \\
&= G_1 + P_1(G_0 + P_0C_0) \\
&= G_1 + P_1G_0 + P_1P_0C_0 \\
C_3 &= G_2 + P_2C_2 \\
&= G_2 + P_2(G_1 + P_1G_0 + P_1P_0C_0) \\
&= G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0 \\
C_4 &= G_3 + P_3C_3 \\
&= G_3 + P_3(G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0) \\
&= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0 \\
&= G_0\ldots3 + P_0\ldots3C_0 \\
G_0\ldots3 &= G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 \\
P_0\ldots3 &= P_3P_2P_1P_0
\end{align*}
\]

- \( P_i = \) Propagate function: propagate a carry from position \( i \) to \( i + 1 \)
- \( G_i = \) Generate function: generate a carry in position \( i \)
- \( P_0\ldots3 = \) Group propagate function
- \( G_0\ldots3 = \) Group generate function
Four-Bit Carry Lookahead Circuit
4-bit FAs with carry lookahead circuit

More expensive than RCP but much more faster

4-bit CLAs can be used to implement circuits for adding 8, 12, 16, ..., $4n$-bit numbers
Subtractors

Half Subtractor, Full Subtractor, Binary Ripple Borrow Subtractor and Borrow Lookahead Subtractor can be defined in the same way as done for the adders. Instead of carry and sum we have borrow and difference, and different truth table and circuits.

Exercise to submit in next lab Define and implement a Half Subtractor, a Full Subtractor and a Binary Ripple Borrow Subtractor. You’ll have a bonus if you implement a Borrow Lookahead Subtractor.

Note: it’ll be marked

Do we really need subtractor circuits?

If not then how can we implement subtraction operation?

Hint: Remember the relationship between binary addition and binary subtraction operations
Circuit is a

**Adder** When $S = 0$:
\[ A + B \]

**Subtractor** When $S = 1$:
\[ A + (2\text{CF of } B) = A - B \]