A ripple carry adder allows you to add two k-bit numbers. We use the half adders and full adders and add them a column at a time.

Before Adding

- Adding column 0
  - add $x_0$ to $y_0$, to produce $z_0$

- Adding column 1
  - In column 1, we add $x_1$ to $y_1$ and $c_i$ and, to produce $z_1$, and $c_i+1$,
  - where $c_i$ is the carry-in for column $i$ and $c_{i+1}$ is the carry out of column $i$. 

**Ripple Carry Adders**
• Adding column 2
  – add $x_2$ to $y_2$ and $c_2$ to produce $z_2$, and $c_3$.

• We can use a full adder
  – for column 0 with $C_{in}=0$

---

**Delay with Ripple Adder**

• combinational logic circuits can't compute the outputs instantaneously.
• some delay between the time the inputs are sent to the circuit, and the time the output is computed.
• Assume the delay is $T$ units of time for one adder, what is the delay for a n-bit ripple carry adder?
  – the adders are working in parallel
  – the carrys must "ripple" their way from the least significant bit and work their way to the most significant bit.
• The delay is not a big problem, usually, because hardware adders are fixed in size.
Summary

• Half adders add two bits.
• Full adders add three bits.
• Ripple carry adders use half adders and full adders to add two n-bit numbers represented in either UB or 2C.
• Ripple carry adders have $O(n)$ delay.
• There are adders that can add faster than $O(n)$, by using more hardware (Carry Lookahead Adders).

Mux and deMux
Multiplexers

- A n-1 multiplexer, or MUX, for short, is a device that allows you to pick one of n inputs and direct it to an output
- Examples:
  - TV: you select one of many channels to be displayed on your screen
  - Radio:
- An n-1 MUX consists of the following:
  - Data inputs: n
  - Control inputs: ceil( log₂ n ) (~ what is this?)
  - Outputs: 1
- Diagram of 2-1 MUX
  - Data inputs: n=2 (x₁, x₀)
  - Control inputs: ceil( log₂ 2 )=1; C: 1 bit
  - Outputs: 1

2-1 MUX cont’d

- Behavior of 2-1 MUX
  - If c=0, then x₀ is directed to the output z
  - If c=1, then x₁ is directed to the output z
- Truth table

<table>
<thead>
<tr>
<th>Row</th>
<th>C</th>
<th>X₁</th>
<th>X₀</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Boolean Expression: \( Z = \overline{C}X₀ + CX₁ \)
- Implementation?
2-bit MUX

- 2-bit MUX
  - a MUX where you select 2 bits at a time, instead of 1 bit.
- For example, you might want to select either \(x_1x_0\) or \(y_1y_0\)
  - Data inputs: 4 \((x_1, x_0, y_1, y_0)\)
  - Control inputs: 1 \((c)\) since we are selecting only one of two inputs
  - Outputs: 2 \((z_1, z_0)\)
- Constructing a 2-1 2-bit MUX from 2-1 1 bit MUXes

Let us "trace" to see what happens to the inputs when \(c == 0\) and when \(c == 1\).

4-1 MUX

- A 4-1 MUX has the following attributes:
  - Data inputs: 4 \((x_3, x_2, x_1, x_0)\)
  - Control inputs: 2 \((c_1, c_0)\)
  - Outputs: 1 \((z)\)
- Truth table
- Boolean expression
  \[ Z = C_1C_0X_0 + \overline{C_1}C_0X_1 + C_1\overline{C_0}X_2 + C_1C_0X_3 \]
- Implementation
  - Two-level representation
  - Sum of products: 4 AND gates and 1 OR gate
- How do we build a 4-1 Mux using 2-1 Muxes?
DeMUX

• Demultiplexers (or DeMUX) are basically multiplexers where the inputs and outputs have been switched.

• An 1-n DeMUX consists of the following:
  – Data inputs: 1
  – Control inputs: ceil( log₂ n )
  – Outputs: n

• A 1-2 DeMUX (n=2)
  – Data inputs: 1
  – Control inputs: ceil( log₂ 2 )=1
    \( C \): a 1 bit number that specifies the output which the input is directed to.
  – Outputs: 2 (\( Z_1, Z_0 \))

1-2 DeMUX cont’d

• Behavior of 1-2 DeMUX
  – When \( c == 0 \), the input, \( x \), is directed to the output \( z_0 \).
  – When \( c == 1 \), the input, \( x \), is directed to the output \( z_1 \).

• Truth table

<table>
<thead>
<tr>
<th>Row</th>
<th>C</th>
<th>X</th>
<th>Z1</th>
<th>Z0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</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>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

• Boolean Expression:

\[
Z_0 = \overline{C}X \\
Z_1 = CX
\]

• Implementation? 1-4 DeMUX?
Summary

• MUX (n-1)
  – How many bits do we need for the control input?
  – How is input selected depending on the control?

• DeMUX
  – How many bits do we need for the control input?
  – How is input directed depending on the control?