A 1-bit CMOS full adder

- Here is a diagram of a 1-bit full adder:

- Lines a, b, cin, sum and cout carry the boolean values T or F.

- Specification of the adder:

\[
\text{Add1}(a, b, \text{cin}, \text{sum}, \text{cout}) \equiv (2 \times \text{Bv}(\text{cout}) + \text{Bv}(\text{sum}) - \text{Bv}(a) + \text{Bv}(b) + \text{Bv}(\text{cin}))
\]

- A correct implementation has:
  - lines a, b, cin, sum and cout
  - constrains \(a, b, \text{cin}, \text{sum} \text{ and cout as Add1}(a, b, \text{cin}, \text{sum}, \text{cout})\)


### Implementation

- A CMOS implementation of the adder:
  - lines with the same name are connected
  - lines p0, ..., p11 are internal
  - horizontal transistors are bidirectional

### Specification in logic

\[
\text{Add1}_{\text{logic}}(a, b, \text{cin}, \text{sum}, \text{cout}) \equiv \\
\text{PTran}(p0, p0, p3) \land \text{PTran}(\text{cin}, p0, p1) \land \text{PTran}(b, p0, p1) \land \text{PTran}(a, p0, p3) \\
\text{NTran}(p0, p0, p3) \land \text{NTran}(\text{cin}, p0, p1) \land \text{NTran}(b, p0, p1) \land \text{NTran}(a, p0, p3) \\
\text{PTran}(p0, p0, p3) \land \text{PTran}(\text{cin}, p0, p1) \land \text{PTran}(b, p0, p1) \land \text{NTran}(a, p0, p3) \\
\text{NTran}(p0, p0, p3) \land \text{NTran}(\text{cin}, p0, p1) \land \text{NTran}(b, p0, p1) \land \text{NTran}(a, p0, p3) \\
\text{PTran}(p0, p0, \text{sum}) \land \text{NTran}(p0, \text{sum}, p0) \land \text{PTran}(p0, \text{cout}, p0) \\
\text{NTran}(p0, p0, \text{cout}) \land \text{NTran}(p0, \text{cout}, p0) \\
\text{Verify by Boolean algebra (tedious) or exhaustive enumeration}
\]

### An n-bit adder

- n-bit adder computes an n-bit sum and 1-bit carry-out from two n-bit inputs and a 1-bit carry-in

- Diagram:

- cin and cout carry single bits, i.e. Booleans

- \(a, b\) and sum carry n-bit words

- \(\text{adder}\ n\) specifies an \(n+1\)-bit adder !!!

- Example: \(\text{adder}(3)\) specifies a 4-bit adder
Adder

### Specification

- The definition of Adder is:
  \[
  \text{Adder}(n, a, b, \text{cin}, \text{sum}, \text{cout}) =
  \begin{cases} 
  \text{Add1} \times V\text{Br}(\text{cout}) + V\text{sum}[n : 0] = \\
  V(a[n : 0]) + V(b[n : 0]) + V\text{cin} 
  \end{cases}
  \]

- Diagram of implementation:

- By primitive recursion:

  Adder

  \[
  \text{Addl}(0, a, b, \text{cin}, \text{sum}, \text{cout}) =
  \text{Addl}(a[0], b[0], \text{cin}, \text{sum}, \text{cout})
  \]

  \[
  \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, \text{cout}) =
  \exists c. \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, c) 
  \]

  \[
  \begin{align*}
  \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, c) & = \\
  \text{Addl}(a[n+1], b[n+1], c, \text{sum}[n+1], \text{cout}) 
  \end{align*}
  \]

### Verification:

- Prove by induction on \( n \) that for all \( n \):

  Adder

  \[
  \text{Addl}(0, a, b, \text{cin}, \text{sum}, \text{cout}) =
  \text{Addl}(a, b, \text{cin}, \text{sum}, \text{cout})
  \]

- Basis:

  Adder

  \[
  \text{Addl}(0, a, b, \text{cin}, \text{sum}, \text{cout}) =
  \text{Addl}(a, b, \text{cin}, \text{sum}, \text{cout})
  \]

- Expanding definitions of Adder and Adder:

  Adder

  \[
  \text{Addl}(0, a, b, \text{cin}, \text{sum}, \text{cout}) =
  \text{Addl}(a[0], b[0], \text{cin}, \text{sum}[0], \text{cout})
  \]

  \[
  \begin{align*}
  \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, \text{cout}) & = \\
  \exists c. \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, c) \land \\
  \text{Addl}(a[n+1], b[n+1], c, \text{sum}[n+1], \text{cout})
  \end{align*}
  \]

  \[
  \begin{align*}
  \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, c) & = \\
  \exists c. \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, c) 
  \end{align*}
  \]

### Induction step

#### Step

\[
\begin{align*}
\text{Addl}(n, a, b, \text{cin}, \text{sum}, \text{cout}) & \Rightarrow \\
\text{Addl}(a, b, \text{cin}, \text{sum}, \text{cout}) 
\end{align*}
\]

#### Assume:

\[
\begin{align*}
\text{Addl}(n, a, b, \text{cin}, \text{sum}, \text{cout}) & \Rightarrow \\
\text{Addl}(a, b, \text{cin}, \text{sum}, \text{cout})
\end{align*}
\]

Then show:

\[
\begin{align*}
\text{Addl}(n+1, a, b, \text{cin}, \text{sum}, \text{cout}) & = \\
\exists c. \text{Addl}(n+1, a, b, \text{cin}, \text{sum}, c) \land \\
\text{Addl}(a[n+1], b[n+1], c, \text{sum}[n+1], \text{cout}) 
\end{align*}
\]

Step continued

If:

\[
(A = B) \land (C = D)
\]

then it follows that \((\Rightarrow)\)

\[
(A + 2\times C) = (B + 2\times D)
\]

hence:

\[
\begin{align*}
\exists c. \left(2^{n+1} V\text{Br}(c) + V\text{sum}[n : 0] = V(a[n : 0]) + V(b[n : 0]) + V\text{cin}\right) & \\
\land \\
\left(2V\text{cout} + V\text{sum}[n+1] = V(a[n+1]) + V(b[n+1]) + V\text{cin}\right) 
\end{align*}
\]

\[
\begin{align*}
\exists c. \left(2^{n+1} V\text{Br}(c) + V\text{sum}[n : 0] = V(a[n : 0]) + V(b[n : 0]) + V\text{cin}\right) & \\
\land \\
\left(2V\text{cout} + V\text{sum}[n+1] = V(a[n+1]) + V(b[n+1]) + V\text{cin}\right)
\end{align*}
\]

\[
\begin{align*}
\exists c. (V\text{sum}[n+1 : 0] + 2^{n+1} V\text{Br}(\text{cout})) = V(a[n+1 : 0]) + V(b[n+1 : 0]) + V\text{cin} & \\
(\Rightarrow) & \\
(V\text{sum}[n+1 : 0] + 2^{n+1} V\text{Br}(\text{cout})) = V(a[n+1 : 0]) + V(b[n+1 : 0]) + V\text{cin} & \\
\Rightarrow & \\
\text{Addl}(n+1, a, b, \text{cin}, \text{sum}, \text{cout})
\end{align*}
\]
Sequential Devices

- Pure combinational adder:
  \[
  \text{Adder}(n)(a, b, \text{cin}, \text{sum}, \text{cout}) \equiv \\
  (2^{n+1} + Bv(\text{cout}) + V(\text{sum}[n : 0])) = \\
  V(a[n : 0]) + V(b[n : 0]) + Bv(\text{cin})
  \]

- \(a, b\) and \(\text{sum}\) range over words
- \(\text{cin}\) and \(\text{cout}\) range over bits (Booleans)

- Zero-delay adder:
  \[
  \text{Combinational Adder}(n)(a, b, \text{cin}, \text{sum}, \text{cout}) \equiv \\
  \forall t. \text{Adder}(n)(a(t), b(t), \text{cin}(t), \text{sum}(t), \text{cout}(t))
  \]

- \(a, b\) and \(\text{sum}\) range over functions from time to words
- \(\text{cin}\) and \(\text{cout}\) range over functions from time to bits

- Unit-delay adder:
  \[
  \text{Unit Delay Adder}(n)(a, b, \text{cin}, \text{sum}, \text{cout}) \equiv \\
  \forall t. \text{Adder}(n)(a(t), b(t), \text{cin}(t), \text{sum}(t+1), \text{cout}(t+1))
  \]

Textbook add-shift multiplier

- A standard add-shift multiplier:

  This can be verified directly

- Verification can be done directly in HOL or using Hoare Logic

- HOL proof by induction on word size
  - essence of proofs (the invariant) are the same
  - compare sections 1.8 and 2.7 of notes (only if you enjoy messy details)