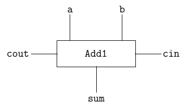
A 1-bit CMOS full adder

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



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

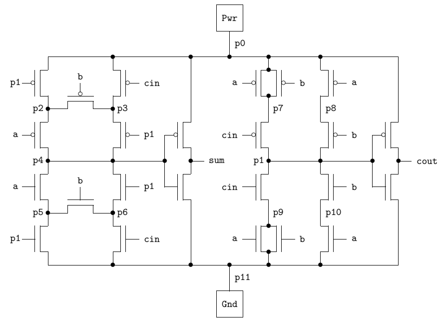
- **Specification of the adder:**

  $\texttt{Add1}(a, b, cin, sum, cout) \quad \equiv$
  $(2{\times}\texttt{Bv}(cout) + \texttt{Bv}(sum) = \texttt{Bv}(a) + \texttt{Bv}(b) + \texttt{Bv}(cin))$

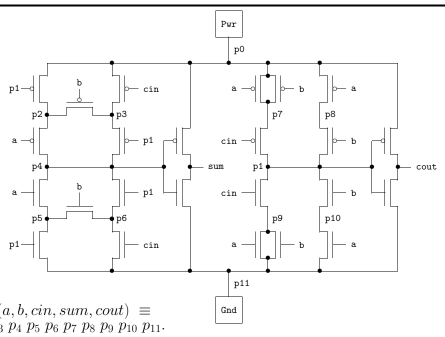
- **A correct implementation has:**
  - **lines** `a`, `b`, `cin`, `sum` **and** `cout`
  - **constrains** a, b, cin, sum **and,** $cout$ **so** $\texttt{Add1}(a, b, cin, sum, cout)$

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



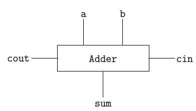
$\texttt{Add1\_Imp}(a, b, cin, sum, cout) \ \equiv$
$\exists p_0\ p_1\ p_2\ p_3\ p_4\ p_5\ p_6\ p_7\ p_8\ p_9\ p_{10}\ p_{11}.$
$\texttt{Ptran}(p_1, p_0, p_2) \wedge \texttt{Ptran}(cin, p_0, p_3) \wedge \texttt{Ptran}(b, p_2, p_3) \wedge \texttt{Ptran}(a, p_2, p_4) \wedge$
$\texttt{Ptran}(p_1, p_3, p_4) \wedge \texttt{Ntran}(a, p_4, p_5) \wedge \texttt{Ntran}(p_1, p_4, p_6) \wedge \texttt{Ntran}(b, p_5, p_6) \wedge$
$\texttt{Ntran}(p_1, p_5, p_{11}) \wedge \texttt{Ntran}(cin, p_6, p_{11}) \wedge \texttt{Ptran}(a, p_0, p_7) \wedge \texttt{Ptran}(b, p_0, p_7) \wedge$
$\texttt{Ptran}(a, p_0, p_8) \wedge \texttt{Ptran}(cin, p_7, p_1) \wedge \texttt{Ptran}(b, p_8, p_1) \wedge \texttt{Ntran}(cin, p_1, p_9) \wedge$
$\texttt{Ntran}(b, p_1, p_{10}) \wedge \texttt{Ntran}(a, p_9, p_{11}) \wedge \texttt{Ntran}(b, p_9, p_{11}) \wedge \texttt{Ntran}(a, p_{10}, p_{11}) \wedge$
$\texttt{Pwr}(p_0) \wedge \texttt{Ptran}(p_4, p_0, sum) \wedge \texttt{Ntran}(p_4, sum, p_{11}) \wedge$
$\texttt{Gnd}(p_{11}) \wedge \texttt{Ptran}(p_1, p_0, cout) \wedge \texttt{Ntran}(p_1, cout, p_{11})$

- 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**

- `Adder` n **specifies an** $n{+}1$**-bit adder !!!**

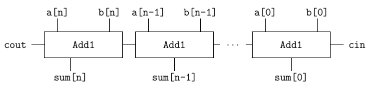
- **Example:** $\texttt{Adder}(3)$ **specifies a** 4**-bit adder**

Specification

- **The definition of `Adder` is:**

  $\texttt{Adder}(n)(a, b, cin, sum, cout) \;\equiv$
  $\quad (2^{n+1} \times \texttt{Bv}(cout) + \texttt{V}(sum\,\texttt{[}n:0\texttt{]}) =$
  $\quad\quad \texttt{V}(a\,\texttt{[}n:0\texttt{]}) + \texttt{V}(b\,\texttt{[}n:0\texttt{]}) + \texttt{Bv}(cin))$

- **Diagram of implementation:**



- **By primitive recursion:**

  $\texttt{Adder\_Imp}(0)(a, b, cin, sum, cout) \;\equiv$
  $\quad \texttt{Add1}(a\,\texttt{[0]}, b\,\texttt{[0]}, cin, sum\,\texttt{[0]}, cout)$

  $\texttt{Adder\_Imp}(n{+}1)(a, b, cin, sum, cout) \;\equiv$
  $\quad \exists c.\ \texttt{Adder\_Imp}(n)(a, b, cin, sum, c) \wedge$
  $\quad\quad \texttt{Add1}(a\,\texttt{[}n{+}1\texttt{]}, b\,\texttt{[}n{+}1\texttt{]}, c, sum\,\texttt{[}n{+}1\texttt{]}, cout)$

Verification:

- **Prove by induction on n that for all n:**

  $\texttt{Adder\_Imp}(n)(a, b, cin, sum, cout)$
  $\Rightarrow$
  $\texttt{Adder}(n)(a, b, cin, sum, cout)$

- **Basis:**

  $\texttt{Adder\_Imp}(0)(a, b, cin, sum, cout)$
  $\Rightarrow$
  $\texttt{Adder}(0)(a, b, cin, sum, cout)$

- **Expanding definitions of `Adder_Imp` and `Adder`:**

  $\texttt{Add1}(a\,\texttt{[0]}, b\,\texttt{[0]}, cin, sum\,\texttt{[0]}, cout)$
  $\Rightarrow$
  $(2^{0+1} \times \texttt{Bv}(cout) + \texttt{V}(sum\,\texttt{[0:0]}) = \texttt{V}(a\,\texttt{[0:0]}) + \texttt{V}(b\,\texttt{[0:0]}) + \texttt{Bv}(cin))$

- **Expanding definition of `Add1` and simplifying:**

  $(2 \times \texttt{Bv}(cout) + \texttt{Bv}(sum\,\texttt{[0]}) = \texttt{Bv}(a\,\texttt{[0]}) + \texttt{Bv}(b\,\texttt{[0]}) + \texttt{Bv}(cin))$
  $\Rightarrow$
  $(2 \times \texttt{Bv}(cout) + \texttt{V}(sum\,\texttt{[0:0]}) = \texttt{V}(a\,\texttt{[0:0]}) + \texttt{V}(b\,\texttt{[0:0]}) + \texttt{Bv}(cin))$

- **Follows by** $\texttt{V}(w\,\texttt{[0:0]}) = \texttt{Bv}(w\,\texttt{[0]})$

Induction step

- **Step:**

  $(\texttt{Adder\_Imp}(n)(a, b, cin, sum, cout) \Rightarrow$
  $\quad \texttt{Adder}(n)(a, b, cin, sum, cout))$
  $\Rightarrow$
  $(\texttt{Adder\_Imp}(n{+}1)(a, b, cin, sum, cout) \Rightarrow$
  $\quad \texttt{Adder}(n{+}1)(a, b, cin, sum, cout))$

- **Assume:**

  $(\texttt{Adder\_Imp}(n)(a, b, cin, sum, cout) \Rightarrow$
  $\quad \texttt{Adder}(n)(a, b, cin, sum, cout))$

- **Then show:**

  $\texttt{Adder\_Imp}(n{+}1)(a, b, cin, sum, cout)$
  $= \exists c.\ \texttt{Adder\_Imp}(n)(a, b, cin, sum, c) \wedge$
  $\quad\quad \texttt{Add1}(a\,\texttt{[}n{+}1\texttt{]}, b\,\texttt{[}n{+}1\texttt{]}, c, sum\,\texttt{[}n{+}1\texttt{]}, cout)$
  $\Rightarrow \exists c.\ \texttt{Adder}(n)(a, b, cin, sum, c) \wedge$
  $\quad\quad \texttt{Add1}(a\,\texttt{[}n{+}1\texttt{]}, b\,\texttt{[}n{+}1\texttt{]}, c, sum\,\texttt{[}n{+}1\texttt{]}, cout)$
  $= \exists c.\ (2^{n+1}\texttt{Bv}(c) + \texttt{V}(sum\,\texttt{[}n:0\texttt{]}) = \texttt{V}(a\,\texttt{[}n:0\texttt{]}) + \texttt{V}(b\,\texttt{[}n:0\texttt{]}) + \texttt{Bv}(cin))$
  $\quad\quad \wedge$
  $\quad\quad (2\texttt{Bv}(cout) + \texttt{Bv}(sum\,\texttt{[}n{+}1\texttt{]}) = \texttt{Bv}(a\,\texttt{[}n{+}1\texttt{]}) + \texttt{Bv}(b\,\texttt{[}n{+}1\texttt{]}) + \texttt{Bv}(c))$

Step continued

If:

$(A = B) \wedge (C = D)$

then it follows that ($\Rightarrow$)

$(A + 2^{n+1}C) = (B + 2^{n+1}D)$

hence:

$$\exists c.\ \overbrace{(2^{n+1}\texttt{Bv}(c) + \texttt{V}(sum\,\texttt{[}n:0\texttt{]})}^{A} = \overbrace{\texttt{V}(a\,\texttt{[}n:0\texttt{]}) + \texttt{V}(b\,\texttt{[}n:0\texttt{]}) + \texttt{Bv}(cin)}^{B}$$
$$\wedge$$
$$\underbrace{(2\texttt{Bv}(cout) + \texttt{Bv}(sum\,\texttt{[}n{+}1\texttt{]})}_{C} = \underbrace{\texttt{Bv}(a\,\texttt{[}n{+}1\texttt{]}) + \texttt{Bv}(b\,\texttt{[}n{+}1\texttt{]}) + \texttt{Bv}(c)}_{D}$$

$$\Rightarrow \exists c.\ \overbrace{2^{n+1}\texttt{Bv}(c) + \texttt{V}(sum\,\texttt{[}n:0\texttt{]})}^{A} + \overbrace{2^{n+1}2\texttt{Bv}(cout) + 2^{n+1}\texttt{Bv}(sum\,\texttt{[}n{+}1\texttt{]})}^{2^{n+1}C}$$
$$= \overbrace{\texttt{V}(a\,\texttt{[}n:0\texttt{]}) + \texttt{V}(b\,\texttt{[}n:0\texttt{]}) + \texttt{Bv}(cin)}^{B}$$
$$+ \underbrace{2^{n+1}\texttt{Bv}(a\,\texttt{[}n{+}1\texttt{]}) + 2^{n+1}\texttt{Bv}(b\,\texttt{[}n{+}1\texttt{]}) + 2^{n+1}\texttt{Bv}(c)}_{2^{n+1}D}$$

$= \exists c.\ (\texttt{V}(sum\,\texttt{[}n{+}1:0\texttt{]}) + 2^{n+2}\texttt{Bv}(cout) = \texttt{V}(a\,\texttt{[}n{+}1:0\texttt{]}) + \texttt{V}(b\,\texttt{[}n{+}1:0\texttt{]}) + \texttt{Bv}(cin))$
$= (\texttt{V}(sum\,\texttt{[}n{+}1:0\texttt{]}) + 2^{n+2}\texttt{Bv}(cout) = \texttt{V}(a\,\texttt{[}n{+}1:0\texttt{]}) + \texttt{V}(b\,\texttt{[}n{+}1:0\texttt{]}) + \texttt{Bv}(cin))$
$= \texttt{Adder}(n{+}1)(a, b, cin, sum, cout))$

Sequential Devices

- **Pure combinational adder:**

  $\texttt{Adder}(n)(a, b, cin, sum, cout) \equiv$
  $(2^{n+1} \times \texttt{Bv}(cout) + \texttt{V}(sum\texttt{[}n:0\texttt{]}) =$
  $\texttt{V}(a\texttt{[}n:0\texttt{]}) + \texttt{V}(b\texttt{[}n:0\texttt{]}) + \texttt{Bv}(cin))$

- **a, b and sum range over words**

- **cin and $cout$ range over bits (Booleans)**

- **Zero-delay adder:**

  $\texttt{Combinational\_Adder}(n)(a, b, cin, sum, cout) \equiv$
  $\forall t.\ \texttt{Adder}(n)(a(t), b(t), cin(t), sum(t), cout(t))$

- **a, b and sum range over functions from time to words**

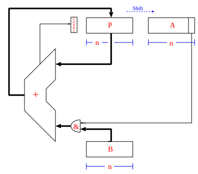
- **cin and $cout$ range over functions from time to bits**

- **Unit-delay adder:**

  $\texttt{Unit\_Delay\_Adder}(n)(a, b, cin, sum, cout) \equiv$
  $\forall t.\ \texttt{Adder}(n)(a(t), b(t), cin(t), sum(t{+}1), cout(t{+}1))$

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 the of proofs (the invariant) are the same**
  - **compare sections 1.8 and 2.7 of notes (only if you enjoy messy details)**