

### Exercise 3.29.

Another way to construct an or-gate is as a compound digital logic device, built from and-gates and inverters. Define a procedure `or-gate` that accomplishes this. What is the delay time of the or-gate in terms of `and-gate-delay` and `inverter-delay`?

### Answer.

Consider the law of De Morgan, which shows different ways for expressing the same logical function:

$$\begin{aligned} A_1 \vee A_2 &= \overline{\overline{A_1} \wedge \overline{A_2}} \\ &= \overline{\overline{A_1} \wedge \overline{A_2}} \end{aligned}$$

To erecting an or-gate of two input signal  $A_1$  and  $A_2$  from and-gates and inverters, we first invert each of them respectively, then compute the **logical-and** of their output and obtain the designated **logical-or** by one more inverting. Figure 1 shows such an or-gate build from and-gates and inverters.

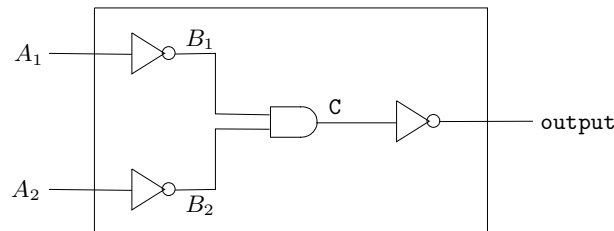


Figure 1. An or-gate build from and-gates and inverters.


We can also express this idea in Lisp:

```
(define (or-gate a1 a2 output)
  (let ((b1 (make-wire))
        (b2 (make-wire))
        (c (make-wire)))
    (inverter a1 b1)
    (inverter a2 b2)
    (and-gate b1 b2 c)
    (inverter c output)
    'ok))
```

We see that in this implementation of or-gate, the and-gate was invoked for once and the inverter was invoked for three times. Hence, the delay time of the or-gate here is

$$\text{and-gate-delay} + 3 \times \text{inverter-delay}$$

---

\*. Creative Commons  2013, Lawrence X. Amlord (颜世敏, aka 颜序).