IF Statement

Provides conditional control of sequential statements.

Condition in statement must evaluate to a Boolean value.

Statements execute if boolean evaluates to TRUE.

Formats:

```
IF condition THEN          --simple IF (latch)
-- sequential statements
END IF;
```

```
IF condition THEN          --IF-ELSE
-- sequential statements
ELSE
-- sequential statements
END IF;
```

```
IF condition THEN          --IF-ELSIF-ELSE
-- sequential statements
ELSIF condition THEN
-- sequential statements
ELSE
-- sequential statements
END IF;
```

IF Statement

Examples:

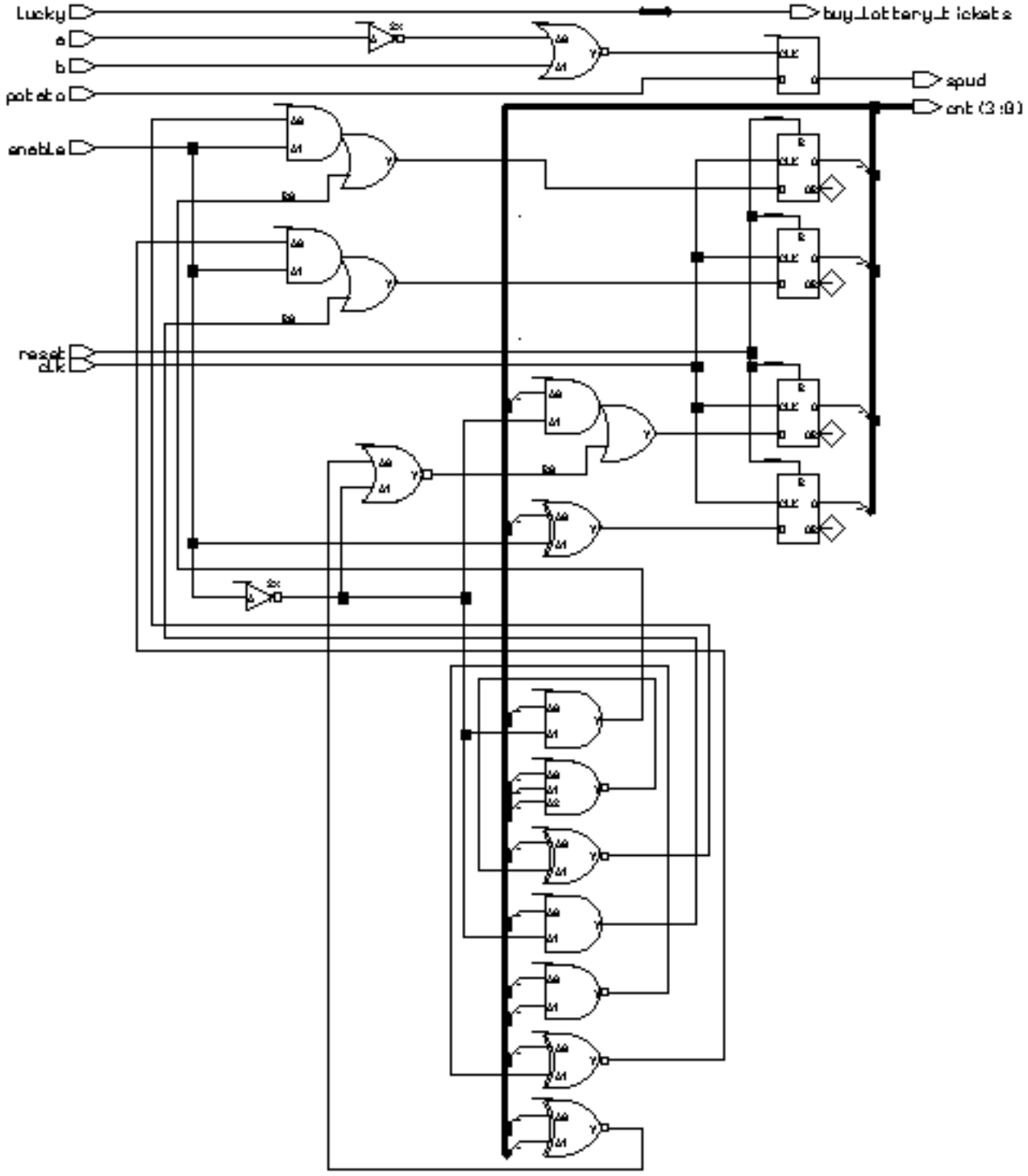
```
--enabled latch
IF (a = ‘1’ AND b = ‘0’) THEN
  spud <= potato;
END IF;
```

```
--a very simple “gate”
IF (lucky = ‘1’) THEN
  buy_lottery_tickets <= ‘1’;
ELSE
  buy_lottery_tickets <= ‘0’;
END IF;
```

```
--a edge triggered 4-bit counter with enable
--and asynchronous reset
IF (reset = ‘1’) THEN
  cnt <= “0000”;
  ELSIF (clk’EVENT AND clk = ‘1’) THEN
    IF enable = ‘1’ THEN
      cnt <= cnt + 1;
    END IF ;
END IF;
```

A Hint: Only `IF.....` needs `END IF`

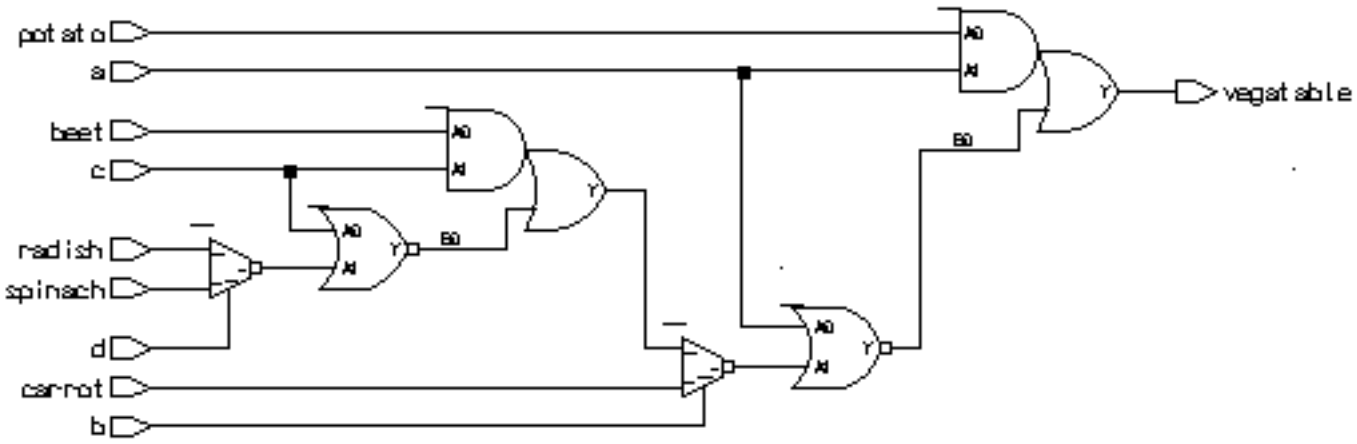
Synthesized example from previous page



IF Implies Priority

The if statement implies a priority in how signals are assigned to the logic synthesized. See the code segment below and the synthesized gates.

```
ARCHITECTURE tuesday OF example IS
  BEGIN
   wow: PROCESS (a, b, c, d, potato, carrot, beet, spinach, radish)
   BEGIN
   IF (a = '1') THEN
     vegatable  <= potato;
   ELSIF (b = '1') THEN
     vegatable <= carrot;
   ELSIF (c = '1') THEN
     vegatable <= beet;
   ELSIF (d = '1') THEN
     vegatable <= spinach;
   ELSE
       vegatable <= radish;
   END IF;
  END PROCESS wow;
END ARCHITECTURE tuesday;
```



what are the delays for each path?

Note how signal with the smallest gate delay through the logic was the first one listed. You can use such behavior to your advantage. Note that use of excessively nested **IF** statements can yield logic with lots of gate delay.

Beyond about four levels of **IF** statement, the **CASE** statement will typically yield a faster implementation of the circuit.

Area and delay of nested IF statement

We can put reporting statements in our synthesis script to tell us the number of gate equivalents and the delays through all the paths in the circuit. For this example, we included the two statements:

```
report_area -cell area_report.txt
report_delay  -show_nets  delay_report.txt
```

In *area_report.txt*, we see:

```
****************************************************
Cell: example    View: tuesday    Library: work
****************************************************
  Cell        Library  References        Total Area
  ao21       ami05_typ     2 x      1      2 gates
  mux21      ami05_typ     2 x      2      4 gates
  nor02      ami05_typ     2 x      1      2 gates

 Number of gates :                        8
```

The *delay_report.txt* has the delay information:

```
             Critical Path Report
Critical path #1   spinach to   vegatable  3.42ns
Critical path #2   radish  to   vegatable  3.41ns
Critical path #3   d       to   vegatable  3.31ns
Critical path #4   c       to   vegatable  2.88ns
Critical path #5   c       to   vegatable  2.57ns
Critical path #6   beet    to   vegatable  2.48ns
Critical path #7   carrot  to   vegatable  1.63ns
Critical path #8   b       to   vegatable  1.52ns
Critical path #9   a       to   vegatable  1.08ns
Critical path #10  a       to   vegatable  1.46ns
```

If implies priority (cont.)

The order in which the IF's conditional statement are evaluated also makes a difference in how the outputs value is assigned. For example, the first check is for (a = '1'). If this statement evaluates true, the output vegetable is assigned "potato" for any input combination where a= '1'.

If the first check fails, the possibilities narrow. If the second check (b= '1') is true, then any combination where a is '0' an b is '1' will assign carrot to vegetable.

If all prior checks fail, an ending ELSE catches all other possibilities.

Relational Operators

The IF statement uses relational operators extensively.

Relational operators return Boolean values (true, false) as their result.

OperatorOperation

```
=                   equal
/=                  not equal
<                   less than
<=                  less than or equal
>                   greater than
>=                  greater than or equal
```

The expression for signal assignment and less than or equal are the same. They are distinguished by the usage context.