Magic Tutorial #S-3: Transistor stacks

Rajit Manohar

Department of Computer Science
California Institute of Technology
Pasadena, CA 91125

This tutorial corresponds to Magic version 7.

Tutorials to read first:

Magic Tutorial #S-1: The scheme command-line interpreter

Commands introduced in this tutorial:

:stack.p, :stack.n, :stack.tallp, :stack.talln, :prs.draw, :prs.mgn,
:prs.talldraw, :prs.tallmgn

Macros introduced in this tutorial:

(None)

1 Stacks

The first step in laying out a gate/operator using magic tends to involve drawing the transistor stacks without any wiring, labelling all the important nodes in the circuit. Since the extractor pretends that nodes that have the same label are electrically connected, the extracted circuit can be simulated using SPICE to obtain some indication of the power/speed of the circuit.

stack.tallp and **stack.talln** can be used to draw such transistor stacks and place contacts where required. These two functions take a transistor width and a list of strings that represent the stack as their arguments, and draw the stack vertically (gates run horizontally) at the current box. For example,

(stack.tallp 40 '(("Vdd") "a" "b" ("Inode") "d" ("out")))

draws a vertical stack of p-transistors with the diffusion being 40 lambda wide. The stack begins with a contact labelled “Vdd”, followed by two gates labelled “a” and “b”, followed by a contact labelled “Inode”, followed by a gate labelled “d”, followed by a contact labelled “out”. Contacts are indicated by placing the string for the label in parenthesis. Note the presence of the quote that prevents the interpreter from attempting to evaluate the list.

The contact width and contact-gate spacing together amount to more than the spacing between adjacent gates. Often it is desired to eliminate this extra space by jogging the poly wires so that the amount of internal diffusion capacitance is minimized. The functions **stack.p** and **stack.n** can be used to do so. Typing

(stack.tallp 40 '(("Vdd") "a" "b" ("Inode") "d" ("out")))

will draw the same stack and jog the poly wires corresponding to the gate for “d”.

2 Production rules

The functions **prs.draw** and **prs.mgn** can be used to draw the transistor stacks that implement a production rule. For instance,

(prs.draw 40 "x & y − > z-")

will draw the stack required to implement the specified production rule with width 20. The function takes a gate width and a single production rule as its arguments. Note that the production rules must be in negation normal form, i.e., all negations must be on variables.

Contacts can be shared between operators by providing a list of production rules as input to **prs.mgn**, as follows:

(prs.mgn 40 20 "x & y − > z-" "u & v − > w-" "˜x & ˜y − > z+")

The contact to GND will be shared between the pull-down stacks for z and w.

Both production-rule drawing function ensure that the variable order in the production rule (from left to right) corresponds to the gate order in the transistor stacks (from power supply to output).

It is not always possible to directly draw a production rule in a single stack with no additional internal contacts. In this case, the function creates an internal node name of the form “_” followed by a number. You can search for these using **(label.search "_*")**, followed by **(label.find-next)**. To complete the implementation, you will have to wire up these internal contacts as well.

Both **prs.mgn** and **prs.draw** draw the transistor stacks using **stack.p** and **stack.n**. The variants **prs.tallmgn** and **prs.talldraw** use **stack.tallp** and **stack.talln** instead.