EXCLUSION CONSTRAINTS FOR DIGITAL MOS CIRCUITS: A NEW SET OF ELECTRICAL DESIGN RULES

Kevin Karplus

Cornell University, School of Electrical Engineering
(also in Computer Science Department)

Abstract: Exclusion constraints are boolean equations that must always be satisfied for an MOS circuit to be adequately modeled by simple switch models. The constraints are generated by a new set of electrical design rules, which are simple enough to be checked automatically. Violating an exclusion constraint does not necessarily mean a circuit is unusable, but careful analysis or analog simulation is needed to ensure digital operation. The rules attempt to formalize common rule-of-thumb design practices, summarized as follows:

Rule 1: Avoid shorting power.
Rule 2: Avoid changing the inputs.
Rule 3: Avoid parallel pullups.
Rule 4: Avoid gates in the middle of pulldown chains.
Rule 5: Avoid charge-sharing.
Rule 6: Avoid odd inverter cycles.

Rule 1 is the primary well-formedness criterion for static cMOS and pre-charged circuits. Rule 2 prevents sneak paths that can cause modules to behave incorrectly when connected. Rule 3 makes inverter ratio checking more accurate, and points out where special ratio computations are needed. Rule 4 detects un-intentional bi-directional pass transistors. Rule 5 detects some situations in which circuit behavior is not digital. Rule 6 detects oscillation and some non-digital circuit behavior.

Introduction

This paper presents a new set of design rules for switch circuits. No information is needed about switch sizes or parasitics, so the rules can be applied before layout. Problems are spotted early in the design, so corrections can be made quickly and cheaply. The six rules in this paper are simple enough to be applied automatically.

The rules apply to both nMOS and cMOS circuits, and use a simple switch model of the circuits. Some useful circuits violate the rules, but these circuits are often not handled correctly by current CAD tools. By pointing out the places where the implicit assumptions of the tools are violated, the rules can focus attention on those parts of the circuit that need more detailed modeling.

The first five rules are checked on a *switch graph*, in which each signal is a vertex, and each switch is an edge connecting the source and drain. Edges are labeled with the signal on the gate of the switch and with the type of switch (nMOS or pMOS). Static load devices (pullups) are not included in the switch graph, but pulled up vertices are marked. A path in the switch graph represents a possible DC connection between two signals.

Boolean expressions are generated to summarize all paths from any node n to V_{dd} (V_n), ground (G_n), a pullup (P_n), a high input ($In1_n$), or a low input ($In0_n$). The expressions can be quickly generated using sparse matrix techniques on the adjacency matrix for the switch graph[1,2].

The sixth rule is checked on an *inverter graph*, in which vertices correspond to signals, and edges to subcircuits that act as static inverters. Each edge is labeled with the conditions needed to make the subcircuit act as an inverter. The label on an edge from x to y can be computed form the paths needed to check the first four rules:

$$\begin{aligned} label(x,y) = &(G_y \vee In0_y)\Big|_{x=1} \\ &\wedge (\neg G_y)\Big|_{x=0} \wedge (\neg In0_y)\Big|_{x=0} \\ &\wedge (V_y \vee P_y \vee In1_y)\Big|_{x=0} \\ &\wedge (\neg V_y)\Big|_{x=1} \wedge (\neg In1_y)\Big|_{x=1} . \end{aligned}$$

Figure 1 shows inverter graphs for some simple subcircuits.

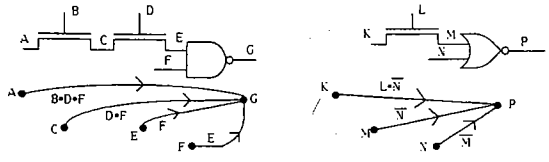

Figure 1: Inverter graphs.

The Rules

Rule 1: Avoid shorting power.

Every path in the switch graph from V_{dd} to ground must have at least one open switch on it (that is, $G_{Vdd} = 0$). Rule 1 generates constraints mainly for cMOS circuits and pre-charged circuits. For cMOS, the power-short constraints are the primary criteria for the well-formedness of logic gates. For pre-charged-circuits (nMOS or cMOS), the power-short constraints are usually satisfied by using a clocking discipline[3,4].

Figure 2 shows a cMOS NAND-gate and its switch graph. Rule 1 requires that $(\neg A \vee \neg B) \wedge A \wedge B = 0$.

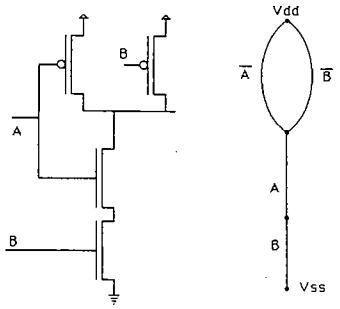

Figure 2: cMOS NAND-gate and its switch graph.

Rule 2: Avoid changing the inputs.

Active paths are prohibited from a low input of a circuit to V_{dd} or a pullup, from a high input to ground, or from an input to an input with a different value, that is:

$$\forall \text{ inputs } n : n \wedge (G_n \vee In0_n) = 0$$
$$\neg n \wedge (V_n \vee P_n \vee In1_n) = 0.$$

This rule tries to prevent "sneak paths" back through the inputs of a circuit or subcircuit. If such sneak paths are allowed, the behavior of a system can not be reliably computed from the behavior of individual subcircuits.

Not all circuits that violate rule 2 are wrong. For example, datapath modules commonly use buses for both input and output.

Rule 3: Avoid parallel pullups.

No signal vertex may be simultaneously connected to two different pulled-up nodes and to ground. This rule is intended to make the usual simple pullup over pulldown ratio calculations accurate. The constraints generated by rule 3 for an array of the two-port dynamic RAM cells of Figure 3 can be satisfied by putting some restrictions on RAM usage:

1) Simultaneous reads and writes on the same bus are prohibited.
2) Writing into a cell from both buses simultaneously is prohibited.
3) Reading the same cell from both buses simultaneously is prohibited when the storage transistor is on.

Although the last seems an unnatural restriction, the storage pulldown must be wider than usual if both buses read from it at the same time, since the saturation currents of the two pullups are added.

Rule 4: Avoid gates in the middle of pulldown chains.

If the gate of a transistor is connected to a pulldown tree, the connection should be through the pulled-up node, not through a lower node in the tree. For cMOS, a gate connected to either an nMOS pulldown tree or a pMOS pullup tree should be connected through the node where the two trees meet.

The circuit shown in Figure 4 will be interpreted by some programs as having a uni-directional information flow through the pass transistor. This is usually intended,

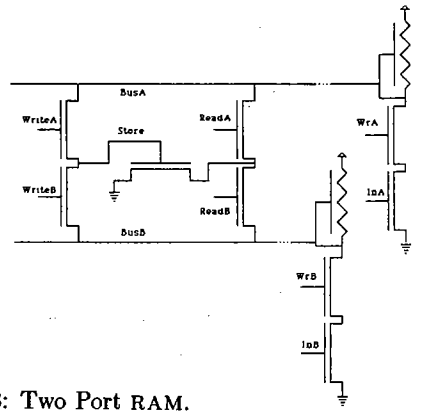

Figure 3: Two Port RAM.

but is correct only if **PASS** and **CLEAR** are mutually exclusive.

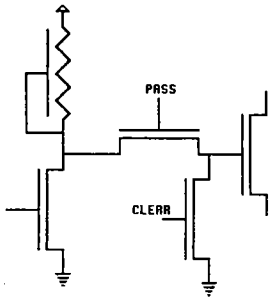

Figure 4. Gate in middle of pulldown chain.

Rule 5: Avoid charge-sharing.

A signal that is used on the gate of some transistor must be either isolated from all other nodes (storing charge) or connected to V_{dd}, ground, or a pulled-up node. When two nodes isolated from power become connected to each other, the charge on the nodes is shared between them. If the nodes initially have different values, the result may be an illegal intermediate voltage. Static storage nodes (nodes on even cycles in the inverter graph) also must be isolated to avoid charge-sharing.

The circuit in figure 5 illustrates the charge-sharing rule. The input is stored at node STORE when LOAD is high during phase 1. LOAD may go high before ϕ_1, sharing the charge between STORE and X. If the LOAD and READ signals are simultaneously high, the illegal value can be propagated to the output.

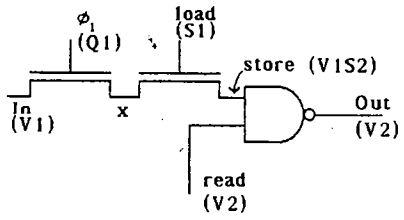

Figure 5: Charge-sharing violation.

Rule 6: Avoid odd inverter cycles.

In the inverter graph, at least one edge in each odd cycle must be inactive. This rule prohibits circuits that oscillate or present intermediate voltage outputs. Such circuits are particularly troublesome for switch simulators, which can get stuck in infinite loops looking for the non-existent digital equilibrium state. Figure 6 shows a circuit with an odd inverter cycle, which generates the constraint $B \wedge E = 0$.

Some proponents of strict two-phase clocking prohibit all cycles in the inverter graph[4]. The extra restriction does not seem necessary, and prohibits many useful circuits (including most static RAM cells).

The subtle dynamic feedback in Figure 7(based on an example by Noice[4]) is easily detected by rule 6, which requires $A \wedge D \wedge I \wedge \neg I- = 0$. Because of delays in the circuit, the constraint may be violated when A rises.

Ring oscillators and RAM bias generators routinely violate rule 6, and need detailed analog simulation to ensure correct design.

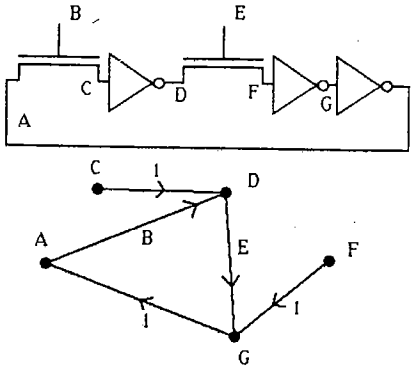

Figure 6: Odd inverter cycle.

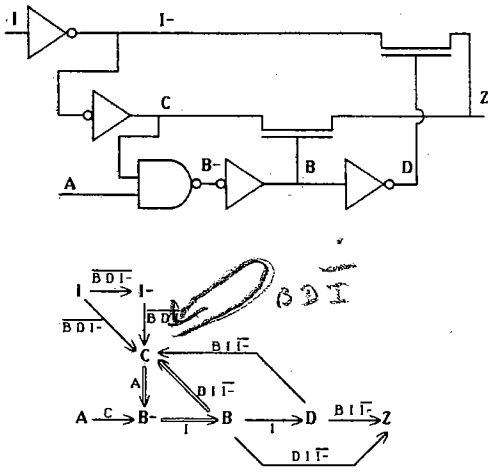

Figure 7: Subtle dynamic inverter cycle.

Exclusion constraint checker

An exclusion constraint checker has been built. Its inputs are a list of switches describing the circuit (esim format) and a set of known constraints on the signals of the circuit. The output from the tool is a list of additional constraints that must be satisfied for the circuit to meet the exclusion rules. The constraints are labeled according to the rule that generated them, so that the appropriate corrective action can be taken.

The current tool checks the rules that can be expressed as excluding paths to distinguished nodes in the switch graph (rules 1–4). Path expressions are built by doing symbolic LU-factorization of the sparse adjacency matrix of the switch graph[1,2]. The checking tool runs fast on small examples, but is somewhat slow on real chips (10 minutes on a Vax 11/780 for a 4000 transistor chip). Most of the time is spent simplifying the boolean expressions.

Current Work

Experiments are being done with two representations of boolean expressions, a compact DNF format and a canonical tree representation[5]. Minor modifications to the graph representation are being added to check rule 5. The inverter graph construction and a fundamental cycle finder[6] is being added to check rule 6.

Conclusions

We have presented a new set of design rules that are simple to check and and easy for designers to learn. They provide an explicit check for the implicit assumptions of many current verification tools.

The exclusion constraint checker has been tested on several circuits by novice designers. It appears to provide concise, understandable information about the limitations of the designs.

References

[1] Robert Tarjan. "Fast algorithms for solving path problems" *Journal of the Association for Computing Machinery* **28**(3), 1981, 594–614.

[2] Alan George and Joseph Liu. *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs NJ, 1981.

[3] Kevin Karplus. *A Formal Model for MOS Clocking Disciplines*, Cornell Computer Science Technical Report 84-632, August 1984.

[4] David Cooke Noice. *A Clocking Discipline for Two-phase Digital Integrated Circuits*, Stanford PhD Thesis, January 1983. University Microfilms 8314482.

[5] Randal E. Bryant. *Graph-based Algorithms for Boolean Function Manipulation*, Carnegie Mellon Computer Science Technical Report CMU-CS-85-135.

[6] E. M. Reingold, J. Nievergelt, and N. Deo. *Combinatorial Algorithms, Theory and Practice*, Prentice Hall, Englewood Cliffs NJ, 1977.