DERIVATION OF SIGNAL FLOW FOR SWITCH-LEVEL SIMULATION †

David T. Blaauw, Daniel G. Saab, and Junsheng Long

Computer Systems Group
Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Ill 61801, U.S.A.

Jacob Abraham

Department of Electrical and
Computer Engineering
University of Texas at Austin
Austin, TX 78712, U.S.A.

ABSTRACT: This paper presents a new algorithm for deriving the direction of signal flow in MOS circuits. The algorithm detects so-called unidirectional transistors. In a unidirectional transistor, signal flow is restricted to one direction during switch-level simulation, without compromising the simulation results. The algorithm uses a static analysis of the switch-level characteristics of the circuit, such as the transistor strengths and node capacitor sizes. It was implemented and used in the simulation of a large, commercial microprocessor. For this processor, 98.5% of the transistors were determined to be unidirectional by the algorithm. The simulation time for this processor decreased significantly when unidirectional transistors were detected.

1. Introduction

Switch-level simulation has become a widely used tool for circuit verification [1, 2]. In switch-level simulation, a transistor is modeled as a bidirectional switch in series with a resistor. Although signals can propagate through this switch in two directions, in practice only a small percentage of transistors exhibit bidirectional signal flow. For most transistors, signal flow can be restricted to one direction. Detection of unidirectional signal flow through transistors simplifies their simulation and therefore increases the performance of the overall simulation.

Erroneously restricting signal flow through a transistor can hide design errors. In directional derivation care must be taken so that a transistor is set unidirectional only if it exhibits unidirectional signal flow under all possible circuit states. Therefore, the algorithm proposed in this paper performs a pattern-independent, or static, analysis of the circuit. The algorithm inspects the switch-level properties of circuitry surrounding a particular transistor to determine if it can be restricted to unidirectional signal flow.

The benefit of signal flow derivation for event-driven simulation and for compiled simulation is discussed in Section 2. Section 3 presents previous work in this area and criteria for reliable direction derivation for switch-level simulation and the algorithm is explained in Section 4. The performance of the algorithm when used for a large, commercial microprocessor is given in Section 5, as well as some concluding remarks.

† This research was supported in part by Semiconductor Research Corporation Contract 88-DP-142, and in part by Motorola, Inc. Austin, TX.

2. Direction Derivation in Simulation

Currently, most switch-level simulators assume that all transistors are bidirectional. This ensures that no bidirectional transistors are restricted in their signal flow, thereby guaranteeing the integrity of the simulation. However, this assumption is very conservative; in practice only a very few transistors are actually bidirectional. The potential gain in simulation speed from detection of unidirectional devices is lost.

Both event-driven simulation and compiled simulation speeds are increased with the identification of unidirectional transistors. In event-driven simulation, a transistor is evaluated by propagating signals across its channel in both directions. Two unidirectional evaluations are therefore required. If, however, a transistor is determined to be unidirectional, only one such evaluation is required and the evaluation cost of the transistor is reduced by 50%. The detection of unidirectional transistors further decreases the simulation time by eliminating the occurrence of false events which are due to propagation of signals across unidirectional transistors in the wrong direction. The exact performance increase obtained from detecting unidirectional transistors will depend on the simulation algorithm, the number of explicit transistors in the circuit description, and the percentage of transistors that are unidirectional.

Compiled simulation is also improved with the identification of unidirectional transistors. Compiled simulation relies largely on path finding algorithms and identification of unidirectional transistors reduces the number of possible paths to be traced [3]. For instance, in the algorithm used by the SLS simulator [4, 5] transistors are evaluated with a sequence of unidirectional evaluations. A bidirectional transistor requires two evaluations in this sequence whereas a unidirectional transistor requires only one. A large percentage of unidirectional transistors will therefore drastically reduce the evaluation cost.

3. Previous Work in Direction Derivation

Until recently, direction derivation relied mostly on manual flagging of the unidirectional transistors [6]. This approach has several serious disadvantages. If all transistors are assumed to be bidirectional unless explicitly flagged, the designer must go through the laborious and error prone task of identifying all unidirectional transistors. This slows down the design/simulation cycle and requires that a person familiar with the design be involved in the simulation. Most importantly, unidirectional evaluation of a bidirectional device can hide a design

error from the simulator. An error in flagging compromises the accuracy of the simulation and allows important design errors to be undetected. Reliable direction derivation must therefore be performed automatically by reliable algorithms.

Significant work in direction derivation has been performed in the area of timing analysis [7,8]. These timing analysis programs use static, local analysis of transistors to detect unidirectional devices. The identification of unidirectional transistors reduces the number of possible paths and eliminates the identification of false critical paths. Signal flow identification for timing analysis differs from that of switch-level simulation in that timing analysis assumes that the logic operation of the circuit is correct. For instance, for a two input decoder, activation of the decoder inputs is mutually exclusive under correct logic operation. Therefore, only one input path conducts at any one time and signal flow through the transistors is unidirectional. However, due to a design error or an injected fault during fault simulation, the two input paths can become conducting simultaneously. In this case, it is possible for signal flow to propagate forward through one input and backward through another. Figure 1 shows how signals can flow forward and backward through transistors $T1$ and $T2$ when the gate of $T2$ is struck-at-1. Since the transistors now conduct bidirectional signal flow, they can no longer be marked as unidirectional.

A transistor that behaves unidirectionally in a logically correct circuit state can, therefore, behave bidirectionally in an incorrect circuit state. In switch-level simulation, such a circuit state could easily occur due to a design error or injected fault and must be accurately modeled. Static direction derivation for switch-level simulation must, therefore, ensure that a transistor be set only unidirectional if it behaves unidirectional under all possible circuit states.

4. The Direction Derivation Algorithm

To perform reliable direction derivation for switch-level simulation, a new algorithm was developed. The analysis of the transistors is static or pattern independent. This has the advantage that the algorithm is performed as a preprocessing step and can be easily used with already existing CAD tools. The algorithm accepts a circuit description in terms of both MOS transistors and simple gates. The detection of unidirectional transistors is based on an analysis of the signal strengths of the circuit components and the capacitance of circuit nodes.

A transistor is restricted to unidirectional signal flow for one of two reasons: either signal flow through the transistor in one direction does not occur at all or signal flow in that direction occurs, but can be ignored. The first situation occurs when the circuit environment of the transistor is such that signals through the transistors can flow only in one direction. Figure 2 shows two small circuits with this situation. In Figure 2(a), a transistor is shown between two inverters, where the size of the transistors is proportional to their driving strength. Since the driving strength of inverter IA is stronger than that of IB, signal flow through $T1$ will always be directed from IA to IB. In Figure 2(b) transistor $T1$ is shown driving the gate of transistor $T2$. Suppose also that node $N2$ has a smaller capacitance than node $N1$. A signal can flow from node $N2$ to node $N1$ only when $T0$ is turned off. However, since the node capacitance of node $N2$ is smaller than that of node $N1$, it is impossible for a signal from node $N2$ to override the signal on node $N1$. Transistor $T1$, therefore, displays signal flow in only one direction and can be set unidirectional in the direction from $N1$ to $N2$.

The second reason for setting a transistor unidirectional occurs when signal flow in one direction through the transistor has no effect on the operation of the circuit. The transistor conducts signal flow in both directions, but one of these does not affect the circuit operation and is ignored. This is the case when signal flow in that direction can never reach the gate of a transistor. Figure 3 shows an example of such a situation. The driving strength of the transistors is indicated after their labels. Since the driving strength of $T2$ is greater than that of $T0$ it overrides $T0$ and can thus force a signal through $T1$ from $N1$ to $N0$.

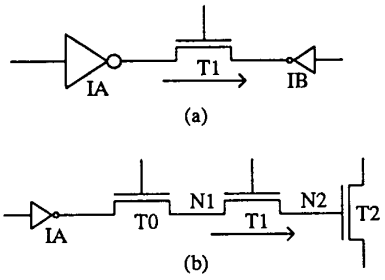

Figure 2. Example circuits with unidirectional signal flow through a transistor.

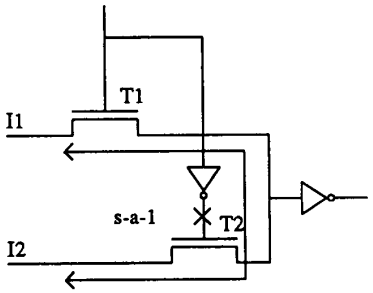

Figure 1. Signal flow through a decoder under fault.

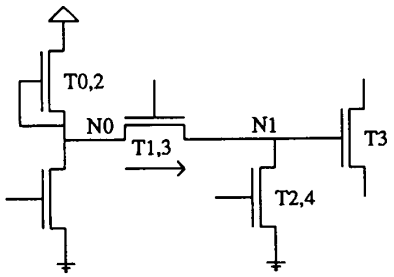

Figure 3. A circuit where signal flow in one direction can be ignored.

Transistor $T1$ therefore displays bidirectional signal flow. However, since $N0$ is connected to a permanent signal source ($T0$), a signal forced from $N1$ to $N2$ cannot persist after $T1$ has been turned off. Therefore, it can never propagate back to $N1$ at a later time and affect the circuit operation. Since there are no transistor gates connected to $N0$ to the left of $T1$, the signal can also not affect the operation of the circuit to the left of $T1$ and can be ignored. Although $T1$ has bidirectional signal flow, it can be evaluated unidirectional without compromising the simulation accuracy.

The direction derivation algorithm consists of three phases. The first phase, called the build phase, constructs a directed, labeled graph. The second phase, called the mark phase, traverses the graph and marks unidirectional transistors. The third phase, the implication phase, examines the orientation of transistors globally and detects the remaining unidirectional devices. The three phases of the algorithm are each explained in more detail below.

4.1. Building the Graph

A graph of the circuit is constructed such that a node in the circuit is represented by a vertex in the graph and a device in the circuit is represented by two edges: one directed from the source terminal of the transistor to the drain terminal and one set in the opposite direction. Power supply and circuit input nodes connected to multiple elements of a channel-connected component are regarded as independent vertices in the graph. The edges in the graph are labeled with the four attributes of **signal strength, signal quality, circuit capacitance,** and **observability**. During the build phase of the algorithm these attributes are propagated along the graph so that each transistor is aware of the information surrounding it. The graph building algorithm is based on a depth-first search of the dc-connected components in a circuit. It starts from each power node or circuit input node and propagates the edge attributes to internal nodes. During this search, the new signal strength, signal quality, node capacitance, and observability for each edge is calculated. The graph attributes are explained below:

Signal Strength

The signal strength of an edge indicates the maximum strength of a signal flowing through the device in the direction of the edge. In a switch-level description, signal strengths form the following ordered set [1]:

$$0 < \kappa_1 < \cdots < \kappa_{max} < \gamma_1 < \cdots < \gamma_{max} < \omega$$

The strength κ_i refers to a charged signal strength from a node with node size i. The strength γ_i is the driving strength through a transistor of size i and ω refers to the strength of a circuit input or power node. The signal strength of an edge is determined by the maximum signal strength of signals flowing into the edge from other devices and is limited by the transistor size. Power and input lines supply signals with an ω strength. A signal is then reduced in strength when traveling through a transistor. For an edge from node v to node w the signal strength can be expressed as follows:

$$streng(v \rightarrow w) = \min \{ \max_{u \neq w} \{streng(u \rightarrow v), \kappa_i\}, TranSize \},$$

$$where\ i = nodeSize(v)$$

Signal Quality

The signal quality describes the persistence of a signal. It is said to be **permanent** if a signal always flows through a particular device. For example, the output of a gate, depletion mode pull-up transistors, and power nodes each have edges with a permanent signal quality. A signal is **intermittent** if the signal sometimes goes through the device and sometimes does not, as is the case with a pass transistor.

Circuit Capacitance

The circuit capacitance describes the size of the circuit nodes in either side of a transistor. The circuit capacitance for an edge $v \rightarrow w$ indicates the maximum node size on the v size of the transistor. The circuit capacitance attribute is calculated as follows:

$$cap(v \rightarrow w) = \max \{ nodeSize(v), \max_{u \neq w} \{cap(u \rightarrow v)\} \}$$

Observation

The observation attribute indicates if there is a transistor gate sensing the signal. The observation attribute originates from either a gate input, transistor gate, or a circuit output. An edge directed from such an observation point is labeled with the observation attribute. The attribute is then propagated along the graph. The observation attribute is important since signal evaluation toward this point is required even if the signal path to this point is weak.

Figure 4 shows an example of a transistor circuit and its corresponding graph constructed in the build phase. The node and transistor sizes are indicated after their labels. Each edge in the graph is labeled with a four attributes indicating, respectively, the signal strength, signal quality (I = intermittent, P = permanent), circuit capacitance, and the observability (O = observation, N = no observation) attributes. Power nodes, circuit inputs,

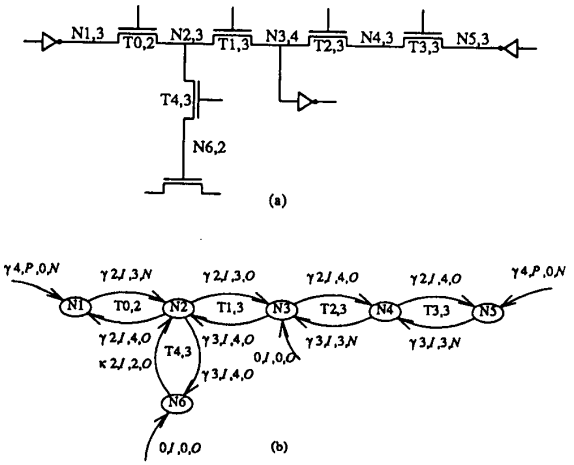

Figure 4. Example of a circuit and its associated graph.

and nodes that are gate inputs and outputs have an additional edge directed from 'NULL' to that node.

4.2. Marking the Transistor

After the build phase constructs the graph and propagates the attributes along its edges, the mark phase inspects the graph locally to a transistor to determine whether a transistor is unidirectional or not. Each transistor is examined twice, once in each direction. The following four rules define when a transistor (T), with channel terminals v and w, is set unidirectional in the direction from $v \rightarrow w$. The rules fall in two sets.

Set 1: There is one or more edges $u \rightarrow v$, where $u \neq w$, such that edge($u \rightarrow v$).qual == permanent and edge($u \rightarrow v$).strength > edge($v \rightarrow w$).circCap.

Rule 1: ($v \rightarrow w$).obs == FALSE.
Since there is a permanent signal incident on v that overrules any stored charge on the node v side of T, a signal propagated from w to v will not persist. Furthermore, since such a signal is not sensed on the node v side of T, it can not affect the circuit operation and T can be set unidirectional in the direction $v \rightarrow w$.

Rule 2: Max{($u \rightarrow v$).str } > Max {($x \rightarrow w$).str},
where $x \neq v$, $u \neq w$ and ($u \rightarrow v$).qual=perm. In this case, there is a permanent signal incident on v that is stronger than the strongest signal incident on w. A potential signal from w to v is thus overridden and the transistor can be set unidirectional from v to w.

Set 2: There is no edge $u \rightarrow v$, where $u \neq w$, such that edge($u \rightarrow v$).qual == permanent and edge($u \rightarrow v$).strength > edge($v \rightarrow w$).circCap.

Rule 3: ($v \rightarrow w$).obs == TRUE and ($w \rightarrow v$).str < nodeSize(v).
Since there is an observation attribute from v to w, signals propagated across T from w to v can affect the operation of the circuit. However, since the strength from w to v cannot change the state of node v, node w can never affect node v. Transistor T can thus be set unidirectional in the direction $v \rightarrow w$.

Rule 4: ($v \rightarrow w$).obs == FALSE and {($w \rightarrow v$).str < nodeSize(v) or ($v \rightarrow w$).cap < nodeSize(w)}.
In this case, signals propagated from w to v cannot be sensed by a transistor gate on the v side of T and must, therefore, be propagated back from v to w to affect the circuit operation. The signal must be propagated from w to v, stored at the node v size of T and, at a later time, propagated back from v to w. This can only occur when the signal from w to v can change the charged signal on v (edge($w \rightarrow v$).strength $\geq$ nodeSize(v)) and the circuit capacitance on the node v size of T is larger or equal than the size of node w (edge($v \rightarrow w$).cap $\geq$ nodeSize(w)). If these conditions are not true, T can be set unidirectional from v to w which, by Demorgan's rule, is if edge($w \rightarrow v$).strength < nodeSize(v) or edge($v \rightarrow w$).cap < nodeSize(w).

Figure 5 shows the graph of Figure 4 after the mark phase has been executed. Only transistor $T1$ is left bidirectional. Transistors $T0$ and $T3$ are set unidirectional by rule 1, transistor $T2$ is set by rule 4, and transistor $T4$ is set by rule 3. Inspection of the circuit shows that $T1$ must remain bidirectional, since a signal can be propagated from $N2$ to $N3$, stored at $N3$, and, at a later time, be propagated back to $N2$.

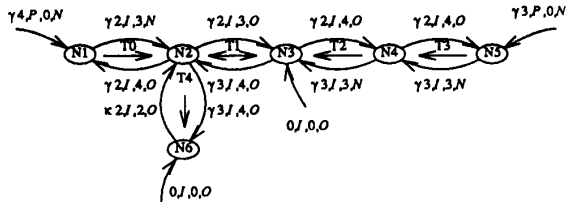

Figure 5. Identification of unidirectional transistors.

4.3. Implication phase

The first two phases of the algorithm discover a large percentage of all unidirectional transistors. In some situations, however, it is possible that a unidirectional transistor is not detected by the algorithm and is left bidirectional. If the circuit in Figure 4(a) is modified to that of Figure 6, transistor $T3$ will no longer be marked as unidirectional since node $N5$ no longer has a permanent signal strength incident on it. Transistor $T3$ is, however, part of a contiguous chain of uniformly directed transistors. Since $T5$ and $T6$ are directed toward $N5$, and $T2$ is directed from $N4$ to $N3$, propagation through $T3$ from $N4$ to $N5$ cannot affect the circuit operation. Transistor $T3$ can thus be set unidirectional from $N3$ to $N4$. In other words, if all transistors on the left of a transistor T are unidirectional in the direction toward it, and all transistors on its right are unidirectional in the direction away from it, T must also be unidirectional in the direction from left to right. Such a transistor is said to be forced unidirectional by implication. Since this situation depends on transistors that are already set unidirectional, it can only be detected after the build and mark phases have been completed. In the implication phase a single or chain of bidirectional transistors that is embedded in a chain of unidirectional transistors is detected and set unidirectional in the direction of the chain. The algorithm is described in pseudo C-code in Figure 7.

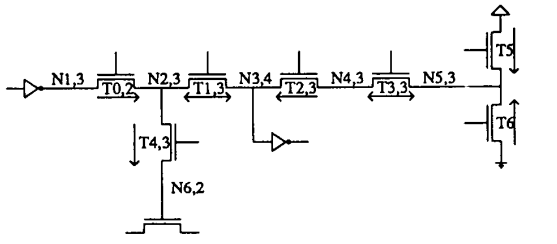

Figure 6 Circuit of Figure 4(a) modified so implication is necessary.

```
imply(u, v)
{
  resolve(u, v);
  if (label[v] == 0) {
    label[v] = 1;
    for (all transistors q with channel { w, v },
        where w ≠ u) {
      start(v, w, q);
      imply(v, w);
    }
  }
}

start(v, w, q)
{
  if (!stackEmpty && q is bidirectional)
    push(q);
  else if (stackEmpty && q is bidirectional &&
        all transistors with channel { x, v },
        where x ≠ w are toward v) {
    dir = FORWARD; push(q);
  } else if (stackEmpty && q is bidirectional &&
          all transistors with channel { x, v },
          where x ≠ w are away from v) {
    dir = BACKWARD; push(q);
  }
}

resolve(u, v)
{
  if (!stackEmpty) {
    if (all transistors with channel { x, v },
       where x ≠ u, are away from v and dir == FORWARD) {
       while(!stackEmpty)
         pop-stack-and-set(direction u → v);
    } else if (all transistors with channel { x, v },
      where x ≠ u, are toward v and dir == BACKWARD) {
      while(!stackEmpty)
        pop-stack-and-set(direction v → u);
    } else if (only one transistor with channel { x, v },
          where x ≠ u and
          that transistor is bidirectional ) {
      continue;
    } else
      while(!stackEmpty) pop-stack;
  }
}
```

Figure 7. Directional implication algorithm.

5. Results and Conclusions

The proposed algorithms were implemented in the C-language and executed on Sun-4 work stations. The program was tested on the circuit description of a large microprocessor. The circuit description of the processor is hierarchical and has as primitives both gates and MOS transistors. It contains approximately 40,000 transistors. With the proposed algorithm, 98.5% of all transistors were detected as unidirectional transistors.

The execution time of the proposed algorithm is roughly equal to the time required for parsing the circuit description. Since the simulation time is several orders of magnitude greater that this and the direction derivation is only performed once per simulation, the added computational cost is insignificant. Rather, the saving in simulation time outweighs the added computational cost. The processor was simulated using the CHAMP [9] simulator, which is a hierarchical event-driven simulator. The speedup achieved using directional analysis varied for different circuit blocks. For circuit blocks containing dense transistor constructs, the achieved speedup was approximately 2.1 times. For hierarchically defined circuit blocks containing both gates and transistors, the achieved speedup was smaller. After directional derivation was performed for the entire processor, the overall simulation speed increased by over 1.3 times.

To ensure the accuracy of the algorithm, the processor was simulated both when transistors were set unidirectional and when all transistors were left bidirectional. The simulation results were then compared to ensure that the produced results were identical in both signal state and signal strength.

In conclusion, the proposed algorithm presents an efficient and accurate means of identifying unidirectional transistors in a circuit. Both event-driven and compiled simulation can model unidirectional transistors simpler and faster than bidirectional transistors. In directional derivation for switch-level simulation, signal flow through a transistor must be considered both under logically correct and incorrect circuit operation. The algorithm was tested on a large microprocessor. For this processor, 98.5% of all transistors were set unidirectional. Since unidirectional transistors are conceptually and computationally much simpler, significant benefit can be had from this process.

REFERENCES

[1] R.E. Bryant, "A Switch-Level Model and Simulator for MOS Digital Systems," *IEEE Transactions on Computers*, vol. C-33, No.2, pp. 160-177, Feb. 1984.

[2] R.E. Bryant, D. Beatty, K. Brace, K. Cho, and T. Scheffler, "COSMOS: A Compiled Simulator for MOS Circuits," *Proc. ACM IEEE 24th Design Automation Conference*, pp. 9-16, 1987.

[3] D. T. Blaauw, R. B. Mueller-Thuns, D. G. Saab, and J. A. Abraham, "Automatic Generation of Behavioral Models from Switch-Level Descriptions," *Int. Design Automation Conference*, pp. 179-184, 1989.

[4] I. Spillinger and G. M. Silberman, "Improving the Performance of a Switch-Level Simulator Targeted for a Logic Simulation Machine," *IEEE Transactions on CAD*, vol. CAD-5, No. 3, pp. 396-404, July 1986.

[5] Z. Barzilai, D. K. Beece, L. M. Huisman, V. S. Iyengar, and G. M. Silberman, "SLS - A Fast Switch-Level Simulator," *IEEE Transactions on CAD*, vol. CAD-7, No. 8, pp. 838-849, Aug. 1988.

[6] N. P. Jouppi, "Derivation of Signal Flow Direction in MOS VLSI," *IEEE Transactions on CAD*, vol. CAD-6, NO. 3, pp. 480-490, May 1987.

[7] N. P. Jouppi, "TV: An nMOS Timing Analyzer," *Proc. of the Third Caltech VLSI Conference*, pp. 71-85, March 1983.

[8] J. K. Ousterhout, "A Switch-Level Timing Verifier for Digital MOS VLSI," *IEEE Transactions on CAD*, vol. CAD-4, No. 3, pp. 336-349, July 1985.

[9] D. G. Saab, R. B. Mueller-Thuns, D. T. Blaauw, J. A. Abraham, and J. T. Rahmeh, "CHAMP: Concurrent Hierarchical And Multilevel Program for Simulation of VLSI Circuits," in *Proc. IEEE Int. Conference on Computer-Aided Design*, Santa Clara, CA, 1988.