Hierarchical Simulation of MOS Circuits Using Extracted Functional Models

Jalal A. Wehbeh and Daniel G. Saab

Center for Reliable and High-performance Computing, University of Illinois at Urbana-Champaign, Urbana, IL 61801.

Abstract

A method for extracting functional models from the switch (or gate) level description of a circuit is presented. Model extraction is done incrementally while maintaining the hierarchical structure of the circuit. Extracted models are represented by Boolean functions that are compiled and linked to the simulator. A prototype simulator was implemented within the CHAMP framework. Simulations using extracted models executed 2.5 to 3 times faster than ordinary switch-level simulations without any loss of accuracy.

1 Introduction

In designing digital MOS circuits, designers often take advantage of some special characteristics of MOS devices, such as charge storage, charge sharing, ratioed logic and bidirectional signal flow to make their designs more efficient. Switch-level simulation allows for fast simulation of such circuits. Simulating a whole chip at the switch-level, however, is becoming more and more difficult as the number of transistors on a chip continues to increase.

This paper presents a technique that we have used to gain speedup during the simulation of MOS circuits, while maintaining the accuracy of switch-level simulation. Starting from a transistor or gate-level description of the circuit, we extract functional models from this description and use them to do simulation. This has the advantage of reducing the amount of work that needs to be done at each pass of the simulation by reusing information that has been gathered in a preprocessing phase. The hierarchical structure of the circuit is maintained during simulation, thus keeping a compact representation (less memory) by reusing module descriptions across different parts of the circuit. In addition, model generation can be done incrementally, which greatly reduces the preprocessing time for a circuit with modules that have been encountered in earlier runs of the simulator for that circuit or for other circuits that use those modules.

1.1 Switch-level simulation

The notation used in this paper for switch-level simulation is similar to that of [1]. An ordered set $S_T = \{s_k, s_{k+1}, \ldots, s_{k+l-1}\}$ is used to represent transistor strength levels. Nodes in the circuit are of two types: input and storage nodes. Input nodes (Vdd, Gnd, and primary inputs) are assumed to have infinite drive capacity ($s_i$). Storage nodes are assigned storage strengths from the set $S_N = \{s_0, s_1, \ldots, s_k-1\}$. Each node in the circuit is also assigned a state, which consists of two quantities: logic value and signal strength. The logic value of a certain node is an element of the set $\{0, 1, X\}$ and its signal strength an element of $S = \{s_0, s_1, \ldots, s_k, s_{k+1}, \ldots, s_{k+l-1}, s_T\}$. We define two operators $||n||_N$ and $||t||_T$ which give the strength of node $n$ and transistor $t$ respectively.

A switch-level circuit is described as a graph with vertices corresponding to nodes in the circuit and edges to the transistor channels. Each component of the graph, also called de-connected component (DCCC) can be solved independent of other DC-CCs. Conducting and potentially conducting ($X$) transistors form paths in the network. A path $p$ is a sequence of nodes $(n_0, n_1, \ldots, n_m)$, such that $(n_{i-1}, n_i)$ is an edge for $i = 1, \ldots, k$ and $||n_0||_N \geq \max_{i=1}^{k} (||n_i||_X)$. A path is called a definite path, if it does not have an $X$-edge on it. Each path $p = (n_0, n_1, \ldots, n_m)$ has a strength $||p||_P = \min_{i=1}^{m} \{||n_0||_N, ||(n_{i-1}, n_i)||_T\}$ associated with it. A path $p$ is said to be blocked at node $n_i$ if there is another definite path $p_q$ leading to $n_i$ with $||n_0||_P > ||p||_P$. The state of a node $n$ can be found by examining all unblocked paths leading to $n$, if all of them set $n$ to $0(1)$, then the logic value at $n$ is $0(1)$, otherwise it is $X$. The signal strength at $n$ is equal to that of the unblocked path(s) leading to it.

When a hierarchical (as opposed to flat) circuit description is used for simulation, some DCCCs may have parts of them on different modules. These DC-CCs are split into smaller DCCCs corresponding to the different modules that they belong to. The signal propagation between the modules is no longer unidirectional. Nodes at which DC-CCs are split are called "input/output" nodes. A module may have to be evaluated more than once due to changes at these nodes from other modules. Input/output nodes are usually few and do not severely affect the performance of a hierarchical simulator.
1.2 Logic representation

A Boolean function \( f \) can be represented as a sum-of-product (SOP) terms or cubes. For every DCCC \( D \), we generate a Boolean description for the output nodes, as well as some of the storage nodes that affect the state of those output nodes. These functions are extracted from the circuit level description. As a consequence, they are completely specified with no don't-care sets. We associate two Boolean functions with every node \( n: \) \( \text{Onset}(n) \) and \( \text{Offset}(n) \). The onset (offset) of a node is the set of cubes that set the logic value of that node to 1 (0). Otherwise the node is set to \( X \). To account for the signal strength level at some node \( n \). We define \( \text{Onset}(n) \) (\( \text{Offset}(n) \)), to be the part of \( \text{Onset}(n) \) (\( \text{Offset}(n) \)) that may set node \( n \) to 1 (0) with a signal strength that is greater than or equal to \( j \). Models for higher-level modules are represented in a similar fashion by combining the models generated for lower-level modules.

2 Previous Work

Saab in [2] presents CHAMP a multi-level hierarchical logic and fault simulator. In [3] ANAMOS, an approach where the circuit is first flattened and then a two-level hierarchy is re-extracted without partitioning DCCCs, using graph isomorphism techniques, is presented.

Pflister of IBM was probably the first to describe arbitrary MOS transistor circuits as boolean functions, while doing switch-level simulation on the Yorktown Simulation Engine. The SLS [4] program was later developed to do the same for a general purpose computer. Hajj and Saab [5] describe a method to generate symbolic logic expressions at specified nodes in a MOS digital circuit in terms of its inputs. Ditlow et al in [6], use an algorithm where boolean equations are generated for the output nodes and solved to get a consistent steady-state solution. Bryant in [1], gives an algorithm to generate boolean equations for a DCCC with different transistor and node strength levels. These equations are then solved using Gaussian elimination for binary matrices to yield a set of boolean formulas that describe the function of the DCCC. This algorithm is used in COSMOS. Other approaches concentrate on extracting complex logic gates from switch-level descriptions [7, 8]. Blaauw et al. in [9] use a hierarchical circuit description and generate behavioral models for commonly used modules. The user classifies the types of modules and table models are produced for small modules by exhaustive simulation. Module specific code allows for generation of behavioral models for some structures such as PLAs and bus modules.

In this work, we keep the hierarchical structure of the circuit and generate covers for parts of DCCCs that lie within a primitive-module and then combine them to get the model for that module. Covers for higher-level modules are generated by combining covers of lower-level modules. Covers are generated for each low-level module once and used by different high-level modules that share its description. Before generating covers, a library is checked to see if they were generated by some earlier run of the simulator or this or some other circuit. Date stamps are used to verify that the covers were not generated before some changes in the module description. Logic minimization routines are used to keep the sizes of generated covers as compact as possible. Generated models are compiled and linked to the program in a preprocessing phase for use in simulation.

3 Model Extraction

The first part of the procedure generates the functions for each DCCC separately. Consider a DCCC, \( D \), that we wish to characterize by its covers. The onsets and offsets for nodes in \( D \) are calculated one strength level at a time, starting from the strongest strength level and working towards the weakest one. Initially, for each input node \( i \), \( \text{Onset}^1(i) \) (\( \text{Offset}^1(i) \)) consists of a single cube that is either a universal (empty) cube (for \( Vdd \)) an empty (universal) cube (for \( Gnd \)) or a cube with a single (complemented) literal corresponding to node \( i \). Calculating \( \text{Onset}^1(n) \) and \( \text{Offset}^1(n) \) for node \( n \) in \( D \), where \( s_k \leq s_j < s_t \), involves examining the logic values of the input nodes and the state of transistors \( t \) with \( |lltll| > j \). Starting from each input node, we do a depth first traversal of the graph. Two functions are kept for each path \( p \). \( \text{Onset}^1_p(n) \) and \( \text{Offset}^1_p(n) \). Traversing a new edge involves intersecting the cubes of these functions with the controlling signal at that transistor's gate. Blocking information at each node \( n \) is maintained and used to eliminate paths blocked by \( \text{Onset}^{k+1}(n) \) and \( \text{Offset}^{k+1}(n) \). After examining all strength levels up to and including level \( k \), if the cubes forming the onset and the offset of the output nodes form a tautology, then the DCCC we are examining is combinational (provided that no gate node of some transistor of \( D \) is in \( D \)), and we need not examine the effect of storage nodes. If not, the effect of storage nodes needs to be calculated. Calculating \( \text{Onset}^k(n) \) and \( \text{Offset}^k(n) \) for node \( n \), when \( s_0 \leq s_j < s_k \), is done in a fashion similar to that of the other case, except that paths originate from nodes \( n \) with \( |lltll| = s_j \). Figure 1 shows how the algorithm calculates the onset and offset at the output of a DCCC. This procedure identifies the relevant storage nodes (storage nodes that affect output nodes), and only the covers of output and relevant storage nodes are kept.

The second part of the procedure generates covers for higher level modules by iteratively combining the generated covers. The user specifies which modules are to be represented using models by asking for a C-function to be generated for those modules in the circuit description file. Combining covers gives us a more global view about the function of the circuit. The model generator can also extract functional models for modules described using logic gates or covers. This is done by using logic functions to represent their behavior and applying the combining routines to those functions. Figure 2 shows the effect of combing 2 simple DCCCs. Note that the simulation re-
4 Hierarchical Simulation

The functional model extraction routines were implemented within the CHAMP [2] simulation environment. CHAMP represents a MOS circuit by data structures that preserve the hierarchy of the circuit. A list of the the input and output (and possibly other) nodes is supplied with the headings of each module. In addition a list of the internal nodes is presented in the module description. A module description can have submodules which can be either other modules or simply transistors. The lowest level modules will correspond to leaves in the tree of the hierarchical description. These can either be C-functions, logic gates, or transistor level modules. Transistor level modules are divided into DCCCs for evaluation during simulation. For modules described by C-functions, a list identifies which pins are input and which ones are output pins, and function calls are used for simulation. Simulation between modules will in general involve unidirectional signal flow except for the the special case of input/output pins. The user can specify which modules are to be represented as C-functions for simulation. If such a function exists and was extracted or written after that module was last changed, then it is used, otherwise a call to the model extractor is made and the function is generated.

Hierarchical simulation forces some DCCCs to be split across different modules, which results in pins acting both as inputs and outputs at the same time. In hierarchical switch level simulation, these pins require that the modules to which they connect be resimulated until their state stops changing. This usually requires one or two iterations and is an inexpensive price to pay as long as such pins are few (as is the case with most of the circuits at which we looked). When extracting functional representations for such modules, we can do the following. If the modules are to be all represented by the same functional model, then the corresponding part of the circuit is flattened and covers for the new DCCC are generated. Otherwise we note that for strength levels less than \( k \) (storage), these signals can be analyzed as regular storage nodes with little modifications, while for strength levels that are greater than or equal to \( k \) we can introduce an additional variable that corresponds to the signal strength at that node as shown in Fig 3. Bounds on the possible levels are easily found by examining the strength levels of transistors connected to that node.

5 Results

A prototype program was implemented in the CHAMP framework. Several MOS circuits were simulated and the results of some typical simulations are shown in Table 1. The "Trans" column of the table gives the number of transistors in each circuit. The cir-
circuits were run as switch level circuits at first. Results of these runs appear in the column marked “switch-level.” Functional models were then extracted and the circuits resimulated. The results are shown in column marked %witch.

We note that there is a factor of 2.5-3 speedup by using the covers for simulation. These circuits were also simulated using COSMOS, and the results for circuits that could be handled were comparable to our method (COSMOS ran out of memory for the IIR Filt. circuit on a SunSpark ELC workstation). Resulting scheme can be used for that part of the circuit to keep the size of those covers reasonable. These are some of the modifications that we are looking into incorporating in our program.

6 Conclusion & Future Work

A method for performing hierarchical simulation of MOS circuits using functional models extracted from the switch or gate level descriptions of the circuits was presented. By maintaining the hierarchical structure of the circuit, less memory is used for simulation since a lot of structures are shared by different modules. Simulation time was lower than that of switch level simulators. The time spent in the preprocessing phase to extract behavioral models for different modules in the circuit is more than compensated for by doing compiled simulation using these models. Furthermore, doing incremental model extraction virtually eliminates the overhead of generating models for circuits that have been slightly modified since the last time they were simulated or for circuits built using standard cell libraries.

The functional models that we extract can be used for other applications, such as fault simulation and test generation at the switch-level. These are currently active research areas that we are pursuing. Combining covers of smaller modules can in some cases generate models that are very large. In these cases some partitioning scheme can be used for that part of the circuit.

References


