

The steady down scaling of Complementary Metal Oxide Semiconductor (CMOS) device dimensions has been the main stimulus to the growth of microelectronics and com-

The subcircuit extraction problem

Nian Zhang, Donald C. Wunsch II
and Frank Harary

puter aided Very Large Scale Integration (VLSI) design. But the more an Integrated Circuit (IC) is scaled, the higher its packing density becomes. Current state-of-the-art transistor design is reaching sub-100-nm gate lengths. If current trends continue, you would have a device with 425 million transistors in 2005 and a processor with 1.8 billion transistors by 2010, said Pat Gelsinger, Intel's vice president and chief technology officer. The increasing size of chips, measured in either area or number of transistors, and the waste of the large capital investment (i.e. current cost is about \$1-2 billion) involving fabricating and testing circuits that do not work, make layout analysis and verification an important part of physical design automation.

VLSI layout analysis and verification

Analysis and verification for physical design can be divided into three areas:

- 1) *Design rule checking* analyzes mask geometries to determine if they meet the size, spacing and enclosure rules specified by the fabrication technology.

- 2) *Circuit extraction and connectivity verification* determines the equivalence of the physical circuit topology to the schematic from which the physical circuit was synthesized. Synthesis creates new representations, or provides optimization to existing representation, for objects being designed.

- 3) *Parameter extraction* determines electrical parameters from the layout information that can be used in simulating the timing of the signals.

Circuit extraction (the first part of #2) is performed after the mask layout design is completed. The goal is to cre-

ate a detailed net-list (or circuit description) for the simulation tool. The mask layout only contains physical data. In fact, it just contains coordinates of rectangles drawn in different colors (i.e. layers). The circuit extractor is capable of identifying the individual transistors and their interconnections (on various layers), as well as the parasitic resistances and capacitances that are inevitably present between these layers. It then generates a net-list associated with the layout. Thus, the "extracted net-list" can provide a very accurate estimation of the actual device's dimensions and the device's parasitics that ultimately determine the circuit's performance. The extracted net-list file and parameters are subsequently used in the *Layout-versus-Schematic (LVS)* comparison and in detailed transistor-level simulations (i.e. post-layout simulation).

Layout-versus-Schematic (LVS)

After the mask layout design of the circuit is completed, the design should be checked against the schematic circuit description created earlier. This process, called *Layout-versus-Schematic (LVS)*, will compare the original network with the one extracted from the mask layout, and prove that the two networks are indeed equivalent. This is done in two steps: the first step, known as circuit extraction, converts the layout into a machine-readable network description; next, the extracted circuit is compared to a description of the original schematic.

One primary difficulty is the dissimilarity in the labeling used in the extracted schematic relative to the original schematic. Designers are frequently confronted with different net-lists representing the same design.

For example, one net-list might be generated from a schematic representation of a circuit, while the other is based on an extraction program from a physical layout of that circuit. Inevitably, the two net-lists employ different names for the nets and devices of the circuit and list the objects in different orders.

What's more, a transistor level net-list for a very large Application Specific Integrated Circuit (ASIC) forms an enormous graph. The verification process can be very difficult and time-consuming if we must ensure that every node in the net-list extracted from the mask layout corresponds exactly to its match-

ing element in the original net-list. Moreover, the process can very quickly become bogged down in the thousands of mismatch errors that are inevitably generated initially.

The most efficient way to overcome these difficulties is to identify a related collection of interconnected primitive devices in a circuit as a gate-level component. This is usually called the *subcircuit extraction problem*. By converting a transistor net-list into a gate, we can handle many more transistors. In addition, we can easily check whether two schematics represent the same circuit at the gate-level representation. If they do, then the program produces a mapping that associates each object in one net-list with the corresponding object in the other. More importantly, if the two net-lists represent different circuits, the program will pinpoint the differences. Furthermore, gate-level simulation is

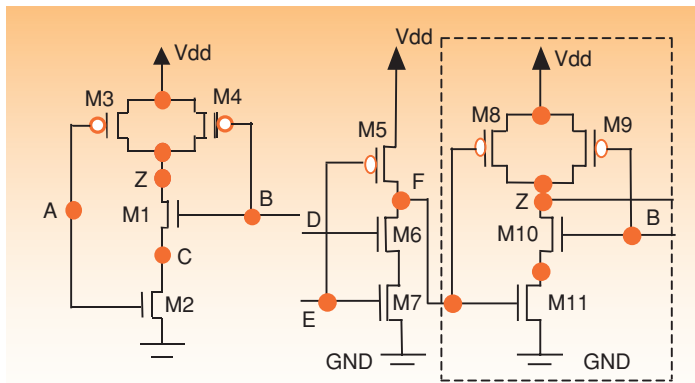


Fig. 1 Pattern circuit

Fig. 2 Main circuit

more time-efficient than transistor-level simulation in checking the timing performance of an ASIC.

Why go through the conversion trouble? The *LVS* step provides an additional level of confidence for the integrity of the design, and ensures that the mask layout is a correct realization of the intended circuit topology. Any errors that may show up during *LVS*, such as unintended connections between transistors, or missing connections/devices, etc. should be corrected in the mask layout—before proceeding to post-layout simulation.

Keep in mind, though, that the *LVS* only guarantees a topological match. For example, if the designer forgets to put a substrate contact in a cell, the layout circuit extractor will still think the substrate is correctly grounded. But, physically, the ground could be very far away. The resulting high resistance

could affect the system's performance.

A subcircuit extraction problem

An example of the subcircuit extraction problem follows. A 2-input NAND gate shown in Fig. 1 serves as the pattern circuit. The circuit shown in Fig. 2 serves as the main circuit. The subcircuit extraction problem is whether or not there is any NAND gates in the main circuit. And, if there are some, how many? As we can tell, the net-list composed of M8, M9, M10 and M11 in the main circuit is equivalent to the pattern circuit.

The problem of subcircuit extraction can be transformed to a subgraph isomorphism problem. Given a graph G , a subgraph, S , has all its nodes and its edges in G . The subgraph isomorphism detection can be defined as: Given a graph S and a larger graph T , find all the subgraphs of T that are equivalent

to S . Similarly, the subcircuit extraction problem is to identify all the subcircuits in the main circuit that are equivalent to the pattern circuit.

The subgraph isomorphism problem is known to be NP-complete in general. In another words,

the run time to detect a subgraph isomorphism between two graphs is, in the worst case, exponential to the number of vertices of these graphs.

Some background on subcircuit extraction

Specialized algorithms have been devised to perform subcircuit extractions since 1983. Early algorithms relied on the specific characteristics of the technology or circuits being transformed. They were not easily applied to different technologies or circuit types, such as analog circuits. Moreover, these techniques relied on assumptions about the subcircuits being extracted, and did not generalize to allow arbitrary subcircuits to be found.

Beginning in 1994, several varieties of advanced algorithms have been coined based on graph theory. By treating the subcircuit extraction as a sub-

graph isomorphism problem that assumes nothing about the underlying circuits, we achieve a true *technology-independent* solution. *Technology-independence* means the same algorithm can be used in many different contexts. They include digital and analog circuits, MOS (metal-oxide-semiconductor) and bipolar technologies, and circuits using varying levels of abstraction.

Miles Ohlrich et al were the first research group to solve the subcircuit extraction problem based on a solution to the subgraph isomorphism. Their algorithm has been implemented in commercial software called SubGemini. Because of its comprehensive experimental results and its fast run time, SubGemini has become a frequently referenced algorithm. The SubGemini works in two phases.

In Phase I, SubGemini identifies all possible locations of the subcircuit in the main circuit. It accomplishes this task by applying a partitioning algorithm to both the subcircuit and the main circuit in order to choose a key vertex, K , in the subcircuit. It also identifies all the possible vertices in the main circuit that might match the key vertex. This set of vertices is called the *candidate vector*, CV. Phase I acts as a filter that tries to reduce the number of instances that need to be checked.

In Phase II, each instance is checked to determine if it is part of a subcircuit. SubGemini's experimental results show that the typical running time for large CMOS circuits is approximately linear, in the total number of devices within the subcircuits being matched. However, their relabeling algorithm relies on the assumption that external nets are not shorted to other external nets of the same subgraph, within the larger circuit. As a result, this algorithm cannot find an instance of the pattern circuit in a shorted circuit.

Huang et al solved this problem by creating a circuit matrix for each subcircuit according to the technology file, and then partitioning each circuit matrix. This partitioning procedure yields a unique ordering of the subcircuit's nodes. Thus, a unique code is generated for each circuit. This algorithm has been implemented on a circuit with up to 15,000 transistors in 28 seconds.

Vijaykrishnan et al in 1996 presented an approach called SUBGEN to model the subcircuit extraction problem using the genetic algorithm. It makes use of the fitness function and genetic opera-

tors—crossover and mutation—to find the candidate strings with the maximum fitness. SUBGEN can identify different kinds of subcircuits. But, it cannot guarantee to find all the subcircuits. This is because a genetic algorithm is in of itself an approximation algorithm. Thus, it cannot guarantee that it will find the optimal solution.

The DECIDE algorithm created by Chang et al in 2001, adopts a recursive scheme to achieve the identification operation. A function assigns a weight value to each node based on its neighbors. A node with a typical type (i.e. N-type, P-type device or a terminal), and with typical neighbors, has a unique weighting value. Thus, the weighting value is used to reduce the number of instances that need to be checked. As a result, the candidate set is formed. This algorithm has been implemented on a circuit up to 100,000 transistors in 16.6 seconds.

In 2002, Wunsch and Zhang proposed a neural networks based Heuristic Dynamic Programming (HDP) algorithm for subcircuit extraction. This approach is the first neural networks approach to solve the subcircuit extraction problem. This algorithm has been implemented in 16.002 seconds on a circuit with up to 100,000 transistors. The pair also proposed a fuzzy attributed graph approach and a heuristic search approach to solve the subcircuit extraction problem. The fuzzy attributed graph approach has been implemented on a circuit with up to 100,000 transistors in 122.125 seconds. (The heuristic search method took 230.4708 seconds.)

Circuit setup

We will now discuss this neural networks based Heuristic Dynamic Programming (HDP) approach in detail. We will use the Circuit Description Language (CDL) format circuit file as our input file. It has the form: $Mxx \ d \ g \ s \ b \ type$. In this case, Mxx is the device name of the Metal-Oxide-Semiconductor Field Effect Transistor (MOSFET). Mxx has three neighbors: d is the drain, g is the gate, s is the source, b is the bulk, and $type$ denotes the type of the device (i.e. N-type or P-type).

A circuit graph contains two types of

nodes: device and terminal (i.e. a node connecting two devices). A device is represented by a square, and a terminal is represented by a circle. Therefore, the circuit graph can be considered like a bipartite graph, in which device vertices connect to only terminal vertices, and terminal vertices connect only to device vertices. For example, the 2-input NAND gate in Fig. 1 can be represented as a circuit graph in Fig. 3 a).

We represent the device as a negative integer, and the terminal as a positive integer. Since $M = 77$ in the ASCII character set, if we then express it as a radix - 10 integer, device $M1$ becomes -

Mxx	d	g	s	b	$type$
M1	19	17	20	1	N
M2	20	7	1	1	N
M3	2	17	19	2	P
M4	2	7	19	2	P

$(77 * 10 + 1) = -771$. We then convert all the transistors' character names into negative integers in the same manner.

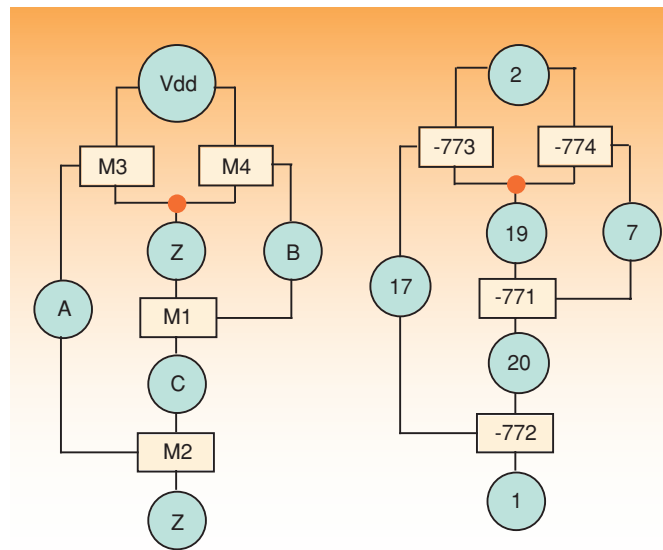


Fig. 3 (a) Circuit graph of a 2-input NAND gate. (b) Coded circuit graph of a 2-input NAND gate.

By connecting the device to its three neighbors, a circuit graph is constructed as shown in Fig. 3 b).

Since the input file might have other kinds of circuits besides the 2-input NAND gate, we proposed an effective approach to form the candidate subcircuits set. First, we analyze the circuit file and partition it into several net-lists, where the boundary is the line with a '2' (i.e. Vdd) in the *source* terminal col-

umn. If the amount of the transistors in a net-list is four, it is considered to be a candidate net-list in the first round. The reason is that a 2-input NAND gate consists of four transistors. Second, we construct circuit graphs for the aforementioned net-lists.

Third, we apply the *Breadth-first Search (BFS)* algorithm to convert each of the circuit graphs to an integer sequence. The *Breadth-first Search (BFS)* algorithm systematically explores the edges of the graph to “discover” every node that is reachable from root s . It computes the distance (smallest number of edges) from s to each reachable node. The algorithm discovers all nodes at distance k from s before discovering any nodes at distance $k+1$. Any sequence that has the same amount of elements as that of the 2-input NAND gate will be considered as the candidate subcircuit.

For example, we have a 60-transistor CDL file that contains ten (10) 2-input NAND gates and ten (10) 2x drive inverters. As we know, a 2-input NAND gate consists of four transistors, while a 2x drive inverter consists of two transistors. The circuit file can be partitioned into 20 net-lists. Since the amount of the transistors in a net-list is four, it is considered to be a candidate net-list in the first round. Thus, we obtain 10 candidate net-lists in the first round. Second, we construct a graph for each candidate net-list by connecting devices to their three terminal neighbors. Then we convert each circuit sequence into an integer sequence by using the *Breadth-first Search* algorithm.

For instance, a candidate net-list shown in Fig. 3 b) can be represented as [2 -773 -774 17 19 7 -772 -771 20 1]. If the amount of the elements in any candidate integer sequence is the same as

the pattern circuit (i.e. 10 for a NAND gate), we pick it as a candidate subcircuit in the second round. All those that don't have 10 elements in their sequence will be ruled out. Therefore, we extract four net-lists as our candidate inputs.

Subcircuit identification

The neural networks based HDP network is used to implement the subcir-

circuit identification. It consists of three neural networks, i.e. the *action* network, the *plant* network and the *critic* network. The function of the action network is to determine whether the input element is 1 (i.e. stands for terminal) or -1 (i.e. stands for device). But its decision might not be right because the network weights are randomly initialized. The action network can give a correct decision only by adjusting the weights.

The plant network compares the decision with the actual input. If they are equivalent, the plant network will output a -1 (i.e. reward); otherwise it outputs a "1" (i.e. penalty). The output is used by the critic network.

The function of the critic network is to adjust the weights in the critic network and action network. After adjusting the weights, the action network can correctly tell whether the input element is 1 or -1. If all the elements of the output of the plant network are "-1" (i.e. reward), then we can say the candidate subcircuit is equivalent to the pattern circuit.

For example, the candidate subcircuit is [2 -773 -774 19 17 7 -772 -771 20 1], and the pattern circuit is a 2-input NAND gate represented as [1 -1 -1 1 1 1 -1 -1 1 1]. The plant network's output gives [-1 -1 -1 -1 -1 -1 -1 -1 -1 -1]. Thus, we conclude that it is a 2-input NAND gate. Another example, if the candidate subcircuit is [2 35 5 -827 -828 6 -825 -826 36 1], the plant network's output becomes instead [-1 1 1 1 1 -1 -1 -1 -1 -1]. Therefore, it is not a 2-input NAND gate.

Conclusions

Subcircuit extraction is becoming a more critical issue with the increasing design sizes of very large scale integrated circuits (VLSICs). We can evaluate the efficiency of a subcircuit extraction algorithm by its run time and identification correctness. Based on current research, the run time depends on the main circuit size, the number of candidate subcircuits in the main circuit, and the size of the pattern circuit. A good subcircuit extraction algorithm should be able to identify different kinds of pattern circuits. Current algorithms can identify 2-input NAND, 2-input NOR, OAI (Or - And - Inverter) gate, 4:1 Multiplexer, 16:1 Multiplexer, 64:1 Multiplexer, Inverter, First In First Out (FIFO) buffer, etc. Moreover, this approach should be able to identify several different kinds of pattern circuits from a given main circuit.

In the future, one of the most important tasks for us is to convert current stand-alone subcircuit extraction algorithms into economic benefits. We should make every effort to find those companies, for example, *Cadence*, *Mentor Graphics*, etc., who would like to incorporate these algorithms into their VLSI layout verification software to speed up the process.

Read more about it

- Miles Ohlrich, Carl Ebeling, Eka Ginting, and Lisa Sather, "SubGemini: Identifying Subcircuit using a Fast Subgraph Isomorphism Algorithm," *30th ACM/IEEE Design Automation Conference*, Dallas, Texas, June 14 - 18, 1993.
- Nian Zhang and Donald C. Wunsch, "A Novel Subcircuit Extraction Algorithm using Heuristic Dynamic Programming (HDP)," *2002 International Conference on VLSI*, Las Vegas, Nevada, June 24 - 27, 2002.
- Nian Zhang, Frank Harary and Donald C. Wunsch II, "CMOS IC Topology Design Verification by Heuristic Dynamic Programming," *Proceedings of ANNIE '02*, St. Louis, MO, Nov. 10 - 13, 2002.
- Nian Zhang and Donald C. Wunsch II, "A Fuzzy Attributed Graph Approach to Subcircuit Extraction Problem," *The IEEE International Conference on Fuzzy Systems*, St. Louis, MO, May. 25 - 28, 2003. (Won Best Student Paper Award)

About the authors

Nian Zhang (an IEEE Student member) was born in Wuhan, China. She received the B.E. degree in electrical and electronics engineering from Wuhan University of Technology, Wuhan, China in 1996, and the M.S. degree in automatic control engineering from Huazhong University of Science and Technology, Wuhan, China in 1999. She is a Ph.D. candidate in computer engineering at the University of Missouri-Rolla, Rolla, MO. She has been conducting active research on VLSI layout verification using neural networks based adaptive critic designs, and other computational intelligence approaches. Her homepage is <http://www.umn.edu/~nzhang>.

Donald Wunsch II (Senior Member IEEE) received the Ph.D. in Electrical Engineering and the M.S. in Applied Mathematics from the University of Washington in 1991 and 1987, the B.S. in Applied Mathematics from the

University of New Mexico in 1984, and completed a Humanities Honors Program at Seattle University in 1981. Since July 1999, he is the Mary K. Finley Missouri Distinguished Professor of Computer Engineering in the Department of Electrical and Computer Engineering, University of Missouri-Rolla. He heads the Applied Computational Intelligence Laboratory. Previously, he was Associate Professor at Texas Tech University. Prior to joining Tech in 1993, he was Senior Principal Scientist at Boeing, where he invented the first optical implementation of the ART1 neural network. He has also worked for International Laser Systems and Rockwell International, and consulted for Sandia Labs, White Sands Missile Range, Texas Tech, Boston University, and Accurate Automation Corporation. He is currently Vice-Chair of the IEEE Neural Networks Society Technical Committee and a member of the International Neural Networks Society Board of Governors, and was General Chair of the IEEE / INNS International Joint Conference on Neural Networks 2003. His homepage is <http://www.ece.umn.edu/~dwunsch>.

Frank Harary holds the Ph.D. in mathematical logic from the University of California, Berkeley, 1948, and has been awarded three honorary doctorates to date: at the University of Aberdeen, Scotland 1975 (mathematics); University of Lund, Sweden 1978 (social sciences); University of Exeter, England 1992 (computer science).

He holds fellowships at both Cambridge University (Churchill College) and Oxford University (Wolfson College) and is also a Fellow of the Indian National Academy of Sciences. He was awarded a Humboldt Senior Fellowship Prize in 1978.

He founded both the *Journal of Combinatorial Theory* (1966) and the *Journal of Graph Theory* (1977, when it won the American Association of Publishers' award for "Best new journal of the year." He has published over 600 papers, written seven books and edited 10 others. His 1969 book, *Graph Theory*, has become the fifth most cited work in the mathematical research literature. He plans to revise it "soon." His 1970 paper, "A formal system for information retrieval from files" was included in the 1996 book, *Great papers in computer science*. His homepage is <http://www.cs.nmsu.edu/~fnh>.