Graph-Based Recognition of High-Level Structures in Transistor Circuits

Liudmila Cheremisinova
United Institute of Informatics Problems
of NAS of Belarus
Minsk, Belarus
cld@newman.bas-net.by

Dmitry Cheremisinov
United Institute of Informatics Problems
of NAS of Belarus
Minsk, Belarus
cher@newman.bas-net.by

Abstract. **The problem of converting a flat transistor circuit into a hierarchical circuit of logical gatesis considered. The problem arises in layout versus schematic verification and reverse engineering of integrated circuits. The offered subcircuit recognition algorithm collects transistors into gates without using any predefined cell library. Graph-based methods are proposed for solving some key problems of subcircuit (CMOS gates) recognizing and logical network extraction. The presented graph methods have been implemented in C++ as a part of a decompilation program, which was tested using practical transistor-level circuits.**

Keywords: **VLSI layout verification, reverse engineering, subcircuit extraction, graph matching**

I. INTRODUCTION

Modern digital circuits contain up to a billion primitive elements at the transistor level, and the circuit complexity is rapidly increases while time-to-market is imposed to decrease. Typically, digital system designers move from a gate-level netlist to a physical layout and mask and rarely proceed in the opposite direction. However, in recent years the study of reverse engineering of digital circuits has become increasingly important.

The step toward raising the level of circuit description is performed by decompiling transistor circuit to replace its representation at a low (transistor) level with a higher-level representation (logic gate level). Tools for solving the task can be used for supporting many tasks of designing integrated circuits, such as functional verification [1], fault simulation and automatic test pattern generation [2], hardware Trojan detection [3], circuit reengineering [4], static timing analysis, etc. At first, the main application of the means of transistor circuit decompilation was verification of the software implementations and finding logical bugs. In recent years, the validation of the integrity of untrusted design becomes a pressing issue. It is recognized, reverse engineering techniques can help detect hardware Trojans and malicious design changes [3].

In the paper we consider the problem of extraction of logical networks from transistor-level circuit netlists in SPICE. In graph interpretation, the problem is formulated as recognition of subgraphs corresponding to logical gates and other subgraphs that are often encountered in a given graph. The problem complexity was thought to be tremendous, but VLSI transistor netlists tend to be sparse enough and have the specific structure, so runtimes did not grow unreasonably, because a sensible data structures and data processing methods were adopted.

There were many attempts to solve the problem of extracting the hierarchy of large-scale subcircuits from a transistor circuits for various VLSI technologies, restrictions, solution methods. An overview of known approaches can be found in [5, 6]. Some methods of logical network extraction are based on structural recognition and use rule-based methods in which CMOS gate structures are recognized as channel connected sequences of MOS transistors [5, 7]. The other approaches [8] are based on mapping transistorlevel circuit into a graph and treating subcircuit pattern matching problem as subgraph isomorphism one. Some methods suppose that subgraphs to be found are known and the problem is reducible to pattern recognition [8].

The proposed paper presents methods and decompilation program for the most general case with no predefined cell library. Moreover, the methods make it possible to recognize subcircuits that implement the same logical functions but are not topologically isomorphic. The method is based on solving well-known graph problems, which are modified to process large transistor-level descriptions in a short time. The presented graph methods have been implemented in C++ as a part of a decompilation program, which was tested using practical transistor-level circuits.

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

II. TRANSISTOR CIRCUIT REPRESENTATION

The source and resulting circuit netlists are presented in SPICE (Simulation Program with Integrated Circuit Emphasis) format [2]. The main part of the circuit netlist in the format is the list of transistors, in which each transistor terminal (drain, gate, source and substrate) is indicated by the label of the net connecting it with the rest of the circuit: M <name><nd><ng><ns><nb><model-name>,where M and «model-name» are the title and the type of the transistor; nd, ng, ns and nb are the labels of nets connected with its drain, gate, source and substrate terminals. For example, the transistor instance description «mp 2 1 3 3 P» is an abbreviated notation for the pairs (mp.d, 2), (mp.g, 1), (mp.s, 3), (mp.b, 3), in which the name «mp» of p-MOS transistor is taken out and the names of its terminals are omitted.

This netlist format defines an electrical circuit as consisting of elements connected to each other by nets. The convenient natural way to represent such circuits is to use an undirected bipartite graph $G = (V_1, V_2, E)$, $V_1 \cap V_2 = \varnothing$, were vertices are divided into classes V_1 and V_2. The vertices fromV_1 correspond to transistor terminals and circuit ports (primary inputs and outputs), and the vertices fromV_2 correspond to nets, i.e. connections between the terminals. Each edge $e \in E$ has one end in V_1 and the other in V_2.

III. DEFINITIONS AND NOTATION

As stated above, transistor circuits are modeled as bipartite graph $G = (V_1, V_2, E)$, $V_1 \cap V_2 = \varnothing$. Throughout this paper we assume that the graph is undirected and vertex-colored, i.e., each vertex has a color associated with it, that is drawn from a predefined set of vertex colors $L(V)$. Transistor-level circuits made by CMOS technology have several types of their nodes: terminals (drain, gate, source and substrate) of n-MOS and p-MOS transistors, input/output ports (external nets), power supply terminals (Vdd and Gnd) and internal nets. So each graph vertex corresponding to n-MOS terminal is assigned by one of the first four colors, p-MOS terminal is assigned by one of next four colors. Then input/output ports, Vdd and Gnd nets, internal nets have unique colors.

The graph corresponding to a MOS circuit is connected (there is a path between any pair of vertices in the graph) and sparse. Two bipartite colored graphs, $G^1 = (V_1^1, V_2^1, E^1)$ and $G^2 = (V_1^2, V_2^2, E^2)$, are isomorphicif there is a one-to-one mapping f: $V_1^1 \leftrightarrow V_1^2$ and $V_2^1 \leftrightarrow V_2^2$ between vertices of graphs such, that for each $v \in V_1^1 \cup V_2^1 L(v) = L(f(v))$ and each edge in E^1 is mapped into a single edge in E^2 and vice versa, i.e. $(v, u) \in E^1$iff$(f(v), f(u)) \in E^2$.

Given graph $G_s = (V_s, E_s)$ is a subgraph of $G = (V, E)$if $V_s \subseteq V$ and $E_s \subseteq E$. Two subgraphs $G_1 = = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ of a graph G are called edge-disjoint if they do not share edges, i.e., they use different sets of edges from E: $E_1 \cap E_2 = \varnothing$.

IV. GRAPH-BASED FORMULATION OF SUBCIRCUIT EXTRACTION PROBLEM

The proposed subcircuit extraction application begins with the construction of a graph model from the SPICE description and hierarchical hash tables for storing the syntax elements of the analyzed circuit [9]. After that, a preprocessing of the circuit is performed, during which some standard fragments are searched for. For example, identification of groups of identical MOS transistors (with the same signals supplying their terminals), connected in series or in parallel, pass gates is fulfilled.

The goal of the transistor circuit decompilation is to recognize subcircuits, which implement logic gates, or, if we cannot, to split the circuit into sufficiently large subcircuits that look like as logic gates In graph interpretation the problem is solved by partitioning a graph into sufficiently large edge-disjoint subgraphs in such a way, that they can be partitioned into the minimum number of classes of isomorphic graphs.

The algorithm realizes two-step process. First, it uses rule-based structural approach in which CMOS gate structures are recognized as channel connected sequences of transistors. Then frequent subcircuit pattern recognition is done to gather the rest transistors into restricted number of identical functional blocks. Finally, the set of all subcircuits, both implementing and not implementing CMOS gates, is partitioned into classes of topologically identical. Subcircuits of the same class represent the same functional block in resulting hierarchical description of two-level decompiled circuit. In graph interpretation the task is to classify subgraphs into classes of isomorphic.

As result of the mentioned steps performing, a hierarchical mixed gate-block-transistor netlist is generated. In the next step, the extraction of logic network from the hierarchical transistor-level circuit is done. That makes it possible to recognize more complex elements than gates. In graph interpretation the task is to extract (out of undirected graph) connected subgraphs only with those vertices that correspond to CMOS gates, and to convert the resulting undirected subgraphs into oriented ones.

V. PARTITIONING A GRAPH INTO CONNECTED SUBGRAPHS

In MOS transistor circuit, correct subcircuits are among channel connected sequences of transistors. So,

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

first, the proposed method of the subcircuit recognition algorithm uses the structural approach to divide the transistor-level circuit into subcircuits, that are channel connected sequences of transistors, as well as in [5, 7]. A group of channel connected transistors is a cluster with three types of external connections: the cluster inputs are fed only to the transistor gates; the outputs are supplied only outside the cluster; there are connections to Vdd and Gnd terminals. Fig. 1 shows an example of grouping transistors into two channel connected components.

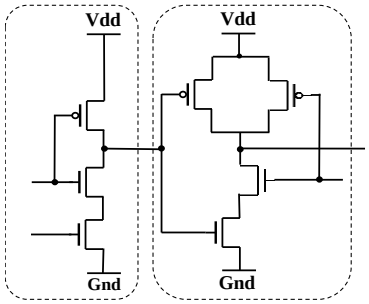

Fig. 1. Channel connected components of MOS transistor circuit

The task of recognition of the clusters is solved on a graph H, obtained from the introduced graph $G = (V_1, V_2, E)$ by removing gate terminals of transistors; by introducing local Vdd and Gndnets (with unit vertex degrees) instead of global ones and by shorting the drain and source terminals for each transistor.

In graph interpretation, channel connected groups of MOS transistors correspond to edge-disjoint connectivity component of the graph H. The search for graph H connectivity components is done by using the well-known depth-first search (DFS) algorithm that starts at some arbitrary unconsidered vertex and explores paths from it as far as possible along each branch before backtracking. Reaching a backtracking results in a new connectivity component. When implementing the DFS algorithm, the initial graph $G = (V_1, V_2, E)$ is not transformed explicitly into the graph H. Instead, the DFS algorithm was tuned to the modification of data structure for storing the graph G.

VI. LOGIC GATE STRUCTURAL RECOGNITION

The CMOS gate consists of two blocks separated by a connection net (output net) (Fig. 2). The first block is formed by n-MOS transistors (pull-down network), which are connected in series by their source/drain terminals. The second block is formed by p-MOS transistors (pull-up network), which are connected in parallel. The pull-down network is placed between the connection node and Gnd, and the pull-up network between Vdd and the connection node. The conductivities of the blocks are complementary, no matter what the input signals (on transistors gates) are, there is a valid path to output node either from Gnd or from Vdd.

A CMOS gate is a group of channel connected transistors; the opposite is not always true. The necessary conditions for the group to belong to the class of CMOS gates are the following ones: the only chain connecting the pull-down and pull-up groups is the output (connection) node; all paths from the connection node go to Gnd or Vdd; pull-down and pull-up networks have the same number of transistors and implement mutually inverse functions. For instance, the right group of channel connected transistors in Fig. 1 is a NAND gate, but the left one is not.

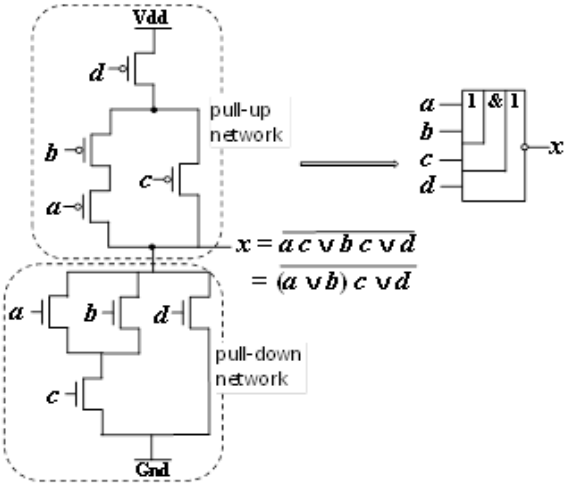

Fig. 2. CMOS gate: its transistor structure and implemented function

Thus, among channel connected components, there are those that implement standard CMOS gates. Their pull-down and pull-up networks have the same number of transistors and implement mutually inverse functions. The task is to find out such subcircuits and their functionality.

In graph interpretation it consists in tracing all paths between vertices corresponding to connection node and Gnd (or Vdd). Each path gives a conjunction of the conductivity variables fed to gate terminals of the transistors from the path. The OR of all such conjunctions yields disjunctive normal form (DNF) for the expression. If the conductivity functions f_n and f_p of pull-down or pull-up networks are complementary ($f_n = \overline{f_p}$) then the analyzed channel connected group is a CMOS gate.

To classify CMOS gates extracted from the transistor circuit, it is convenient to represent the recognized functions as parenthesized algebraic expressions. Such a form is constructed by the algebraic factoring of the found DNF [9]. For the CMOS gate in Fig. 2 we have

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

$f_n = ac \vee bc \vee d = (a \vee b)c \vee d$, $f_p = \overline{a}\overline{b}\overline{d} \vee \overline{c}\overline{d} = (\overline{a}\overline{b} \vee c)\overline{d}$

and $f_n = \overline{f_p}$. Thus, it is standard NOAO2 CMOS gate.

Transistor subcircuits of CMOS gates are divided into classes of functionally identical according to the formulae of implemented logic functions.

Often it is important to take into account not only functional aspect but the topological one too. The aspect requires dividing a class of functionally equivalent CMOS gates into subclasses of topologically equivalent ones. Some features of the topological implementation of circuits at the transistor level, which must be taken into account are given in [9]. For example, we should take into account the interchangeability of drain and source terminals of MOS transistors which results in existence of topologically different subcircuits that implement the same logic function. For instance, there are four subcircuit variants for a CMOS inverter. If, in a decompiled circuit, all variants of a logic gate subcircuit are represented by the same subcircuit, then the decompiled and original circuits will be topologically not isomorphic. Topological equivalence of CMOS gate implementations can be established by checking whether the corresponding graphs are isomorphic or not.

VII. GRAPH ISOMORPHISM AND CANONICAL LABELING

One of the key operations required to partition the set of subgraphs into classes of isomorphic ones consists in checking whether two subgraphs are identical or not. One way of performing this check is to perform a graph isomorphism operation. But in our case, when many such checks are required for the same set of subgraphs, a better way of performing the task is vertex canonical labeling. It assigns to each graph a unique code (a sequence of bits) that is invariant on the ordering graph vertices. Comparing canonical labels allows to partition the set of graphs into classes of pairwise isomorphic graphs.

In general, calculating canonical labels is computationally hard, but in our case, the complexity of the task is reduced due to taking into account special properties of subgraphs under classification: they are vertex-colored, sparse and small enough. Canonical labels of graphs, makes it is possible to sort them in a unique and deterministic way.

Canonical labeling is done in an iterative manner in the process of building a sequence of vertex partitions that defines an ordering of the graph vertices. Assume we have an ordered collection of subsets of the vertices (V_1, V_2,…, V_k), whose union is V. They say that all vertices from the same subset V_i have the same label i. The set of these subsets represents the partition on the set of graph vertices, constructed from the initial partition that is specified by colors and degrees of vertices.

At first, the number and sizes of these subsets V_i must be the same for both compared graphs, i.e. the graphs have identical partitions of the set V. Then we repeatedly apply a relabeling step, which assigns to each vertex v a classifier: $C(v) = (n_1, n_2,\ldots, n_k)$, where n_i is the number of vertices in subset V_i that are adjacent to v. Using these classifiers, each subset V_i can be partitioned into subsets, where each subset should include all vertices with the same classifier. These subsets are lexicographically ordered according to their classifiers. If a division has been fulfilled, then all classifiers are recalculated (and vertices are relabeled). No division will be done if all vertices in each subset V_i have identical classifiers.

It is clear from the description that the essential idea is to relabel vertices so that each new classifier reflects information about a gradually increasing region around the vertex. In an ideal situation, after exhaustive applying the relabeling process, all subsets in partition (V_1, V_2,…,V_k) will become singletons (containing exactly one member), such a graph canonical labeling is called discrete. If two compared graphs have the same canonical labeling then they are isomorphic with each other.

VIII. GRAPH-BASED SUBCIRCUIT RECOGNITION METHOD

After structural recognition of logic gates and pass gates there are two main unrecognized groups of transistors. They are separate ungrouped transistors and found channel connected components of MOS transistors, which have been not recognized as CMOS gates, so they are assigned to be pseudo gates. In our case when there is no cell library, all we can do is to classify remaining pseudo gates into classes of pairwise identical subcircuits.

In graph interpretation the task consists in testing isomorphism between graphs by means of comparing their canonical labelings. To simplify the canonization problem, the subcircuit graphs are complemented with edges connecting all four terminals for each transistor. As the prototype of the program for computing canonical isomorphs, the program «bliss» [10] has been modified that provides fast handling of large and sparse graphs. The experiments with the program of graph canonical labeling have shown that the canonization of pseudo gate graphs results in becoming discrete canonical labelings.

The graphs of pseudo gates with the same initial partitions on the set of its vertices are considered one

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.

by one. For each of them, a canonical labeling is generated and a hash of the canonized graph is computed. Graphs with equal hashes are isomorphic and they are changed in a hierarchical SPICE description with their canonical isomorph.

IX. Logic Network Construction

The next step is to extract from the mixed circuit a logical network which consists only of CMOS and pass gates. In graph interpretation, logical network is directed connected graph $H = (W, A)$. The set of vertices W is partitioned into three subsets: network inputs and outputs, and internal vertices. Each vertex is labeled with input or output variable, or, if it is internal vertex, with the function realized by the corresponding gate. A directed edge $a = (u, v) \in A$ goes from the source vertex u to the target vertex v ($u, v \in W$). Further we consider that graph $H = (W, A)$ is specified by the adjacency list, i.e. an array D of the length $|W|$ where each entry $D[i]$ is a pointer to a linked list of all the out-neighbors of vertex $w_i \in W$.

The connected graph $H = (W, A)$ is extracted from undirected bipartite graph $G = (V_1, V_2, E)$ corresponding to the object mixed circuit. Graph H is contained in G as the connected component C that includes only vertices corresponding to CMOS or pass gates. There can be more than one such a component in a graph G. Each undirected connected subgraph corresponding to a connected component in an undirected graph G is transformed into a directed connected graph $H_i = (W_i, A_i)$ of some logical network. The transformation is carried out in the process of traversing the subgraph along the paths in-going or out-going from the vertices labeled as gates.

The search of the next connected component C begins with any unconsidered vertex labeled as gate and is done by the breadth-first search (BFS) method considering only vertices labeled as gates. BFS allows not only to find out a connected component C, but also to get its topological sorting, which orders the vertices so, that the ordering respects reachability. In other words, if a vertex u is directly reachable from v, then the edge $(u, v) \in E$ generates $(v, u) \in A$, and if the vertex v belongs to the i-th graph rank then the vertex u belongs to the $(i+1)$-thrank. The proposed method provides to extract logic network that is ranked lexicographically. From a lexicographically ordered network of logical gates, it is easy to pass to the formulas of logical equations that specify the output functions of the network.

The next task connected with the logic network extraction is to determine its primary inputs and outputs. It is solved by considering fan-ins and fan-outs for all vertices of the graph $H = (W, A)$. If all vertices from both fan-in and fan-out of some vertex $v \in W$ are labeled as gates then the vertex v is an internal one. Non-internal vertices are referred to primary inputs or primary outputs, depending on which of the fan-in and fan-out sets contains a non-internal vertex.

After the gate-level networks are extracted it is possible to recognize more complex elements, than gates, when a cell library is known.

X. Conclusion

In the paper we present graph-based methods for solving the task of extracting gate-level circuits from transistor-level descriptions for the most general case when any predefined cell library of logic gates is unknown. We have used well-known graph methods, modifying them so that they process large data of special types in a short time. The proposed methods were implemented in C++ as a part of a decompilation program. The program was tested using practical and automatically designed transistor-level circuits. The tested circuits had up to 100000 transistors. Some results of experiments can be found in [11].

References

[1] M. S. Abadir, J. Ferguson,“An improved layout verification algorithm (LAVA)”, Proc. of European Automation Conf., 1990, pp. 391–395.

[2] R. J. Baker,“CMOS circuit design, layout, and simulation”, 3rd ed., Wiley-IEEE Press, 2010.

[3] R. Torrance and D. James, “The state-of-the-art in IC reverse engineering”, Proc. of the 11th Int’l. Workshop on Cryptographic Hardware and Embedded Systems, CHES ’09, 2009, pp. 363–381.

[4] V. D. Hunt,“Reengineering: Leveraging the Power of Integrated Product Development”, Wiley, 1993.

[5] L. Yang, C-J.R. Shi, “FROSTY: A program for fast extraction of high-level structural representation from circuit description for industrial CMOS circuits”,Integration, the VLSI Journal, vol. 39(4), 2006, pp. 311–339.

[6] N. Zhang, D. C. Wunsch, F. Harary,“The subcircuit extraction problem”, Proc. IEEE Intern. Workshop on Behavioral Modeling and Simulation, 33(3),2003, pp. 23–25.

[7] “Logic Gate Recognition in Guardian LVS – Silvaco”. In https://www.silvaco.com/content/appNotes/iccad/2-003_ LogicGates.pdf (access date: 4.1.2021).

[8] M. Ohlrich, C. Ebeling, E. Ginting, L. Sather, “SubGemini: identifying subcircuits using a fast subgraph isomorphism algorithm”, Proc. IEEE/ACM Design Automation Conf., 1993, pp. 31–37.

[9] D. I. Cheremisinov, L. D. Cheremisinova, “Extracting a logic gate network from a transistor-level CMOS circuit”, Russian Microelectronics, vol. 48(3), 2019, pp. 187–196.

[10] T. Junttila, P. Kaski, “Engineering an efficient canonical labeling tool for large and sparse graphs”, Proc. of the Meeting on Algorithm Engineering & Expermiments, New Orleans, Louisiana, January 6 July 2007, pp. 135–149.

[11] D. Cheremisinov, L. Cheremisinova,”Subcircuit pattern recognition in transistor level circuits”, Pattern Recognition and Image Analysis, vol. 30(2), 2020, pp. 160–169.

Pattern Recognition and Information Processing (PRIP'2021) : Proceedings of the 15th International Conference, 21–24 Sept. 2021, Minsk, Belarus. – Minsk : UIIP NASB, 2021. – 246 p. – ISBN 978-985-7198-07-8.