An Algorithm For Identifying The Recurring Subcircuits

Xiaobai Li[1] **Honglei Qin**[1] **Rongling lang**[1]

[1]School of Electronic and Information Engineering in Beijing University of Aeronautics and Astronautics, Beijing, China, 100083

Abstract

The problem of detecting the given subsystems in complex circuits is now an important problem in the computer aided design of VLSI. The algorithm for solving the problem of detecting recurring subsystems is presented in this paper. The algorithm is structure independent, namely any circuit which can be described as a digraph can be handled by this algorithm. The computational complexity is reduced by dividing the algorithm into four phases, such as preprocessing, locating, decomposing and labeling. The experiment results indicate that the run time of the algorithm is influenced by the parameters of graphs.

Keywords: CDFG; Isomorphism; Subcircuits

1. Introduction

Datapath circuits have lots of recurring subcircuits. Designers often exploit the recurring templates in circuits to achieve layout with a small area and a high performance. Design effort can be reduced by detecting recurring subcircuits, so the productivity of designers being improved. The other use of detecting recurring modules is that the subcircuits that occur frequently among the target applications can be considered for implementation in hardware to increase the chances of hardware reuse.

The problem has been studied for quite some time, and various techniques for identifying sub-circuits have been proposed in the literatures [1, 2, 3, 4, 5, 6, 7, 8, 9]. At present Sub-Gemini algorithm [3] is very popular. Sub-Gemini algorithm is a fast algorithm, and it can identify the module of gate level even higher level, but Sub-Gemini can not recognize of NANDs with short inputs correctly. The algorithm in [4] identifies the meaningful subcircuits using the equivalence problem of Boolean functions. The algorithms in [4, 5, 6] are semantic technique, namely they can identify the functional subcircuits, but they can only identify the high-level component from gate-level netlists. The algorithm in [7] is fast when identifying the subcircuits with tree structure, and the computational complexity will increase when the circuits having circles. The method is presented in [8, 9] for identifying the modules from combinational circuits using polynomials mode. The combinational circuit can be represented by a unique polynomial, so the isomorphic subcircuits can be identified by the comparing of polynomials. It will expend much time when calculating the polynomial coefficients, so the algorithm is invalid when the scale of the circuits is large.

The rest of this paper is organized as follows manner. In section II, we formalize the problem of identifying subcircuits. In section III, the algorithm for identify subcircuits is presented. In section IV, we present experimental results. We conclude in section V.

2. Problem formulation

The problem of identifying sub-circuits is as follows: given a circuits C represented by its Control Data Flow Graph (CDFG) and a series of subcircuits $T_i, 1 \le i \le n$, Find all the sub-circuits of C that are isomorphic to $T_i, 1 \le i \le n$. CDFG includes operands of mathematical operations, the state-space and control information. CDFG of a circuit is a directed labeled graph, so after translating the C code or VHDL code to CDFG, the problem of identifying circuits is essentially the well know subgraph isomorphism problem.

The complexity will be improved if the available algorithms are directly used for identifying subcircuits while ignoring the characters of CDFG, such as exiting indexes with zero in-degree, maximal degree being small, etc. Therefore it is necessary to study the isomorphic algorithm aiming at circuits, although there are some good algorithms for solution the isomorphic problem.

3. Isomorphism algorithm

A heuristic isomorphic algorithm is put forward for the problem of subgraph isomorphism of directed labeled graph, which has a corresponding architecture with circuit. The algorithm can find all the subgraphs

in G_2, which are isomorphic to G_1. The algorithm consists of four phases:

(1) Preprocessing to eliminate obviously the non isomorphic subgraphs situation;

(2) Locating all the possible isomorphic subgraphs;

(3) Decomposing G_2 to reduce the computational complexity;

(4) Detecting all the recurring subgraphs using labeling.

3.1. Preprocessing

Suppose that $G(V,E)$ is a labeled graph. The vector $NV_G = (|N_G^1|, |N_G^2|, \cdots, |N_G^k|)$ is defined as feature vector, where k is the number of types of V according the label of vertexes, $|N_G^i|$, $(1 \le i \le k)$ is the node number with the i-th label.

Theorem 3.1: G_1 and G_2 are two labeled graphs and the vertexes in G_1 and G_2 are divided into k kinds according to the label. Suppose that NV_{G_1} and NV_{G_2} are the feature vectors of G_1 and G_2, and prescribe $|N_{G_j}^i| = 0$, if there are not vertexes with i-th label in G_j. There is not subgraphs in G_2 isomorphic to G_1 if $\max_{1 \le i \le k}(N_{G_1}^i - N_{G_2}^i) > 0$.

3.2. Locating

It is necessary to identify the locations of all the possible isomorphic subgraphs in order to find all the isomorphic sub-graphs. The basic nodes for decomposing in phase 3 and the starting nodes for labeling in phase 4 are determined in this phase.

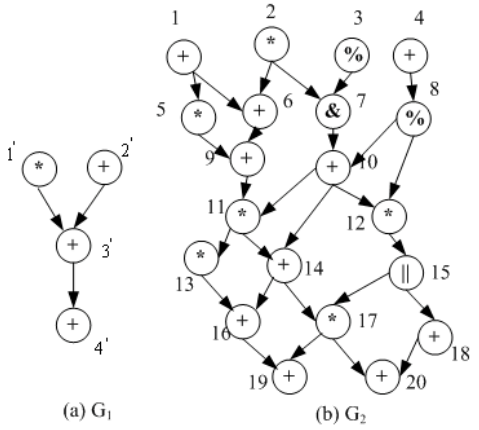

Fig. 1:The Example of graphs for subgrph isomorphism

Definition 3.1: $G(V,E)$ is a directed graph, $v \in V$, $P_v = \{u \mid u \in V,$ and exit directed path $< v,u >$ in $G\}$ .The eccentricity of vertex v represented by $E(v)$ is defined as $\max_{u \in P_v}\{|< v,u >|\}$.

The vertex g in G_2 is defined as the matching node of s in G_1 if vertex s is the corresponding vertex of g under isomorphic meaning. If vertex v is the matching node of u, v should at least satisfy the following conditions:

(1) u and v having the same label;

(2) the in-degree of v in G_2 being not less than that of u;

(3) $E_{G_2}(v) \ge E_{G_1}(u)$.

Every vertex u in G_1 with zero in-degree has a set of vertexes satisfying the above conditions, which are the possible matching nodes of u. In order to simplify the task of subgraph identification in the next phases, the vertex with the least number of the possible matching nodes is selected as the leading node for labeling in phase 4. The vertex is represented by v_0, and the set corresponding with v_0 is represented by V_0. Fig.1 shows the example graph, and $v_0 = v_{1'}$, $V_0 = \{v_2, v_5, v_{11}, v_{12}, v_{13}\}$.

3.3. Decomposing

In this phase, G_2 is divided into a series of sub-graphs containing some nodes of V_0. Searching the isomorphic graphs in the subgraphs will reduce the complexity of labeling in phase 4.

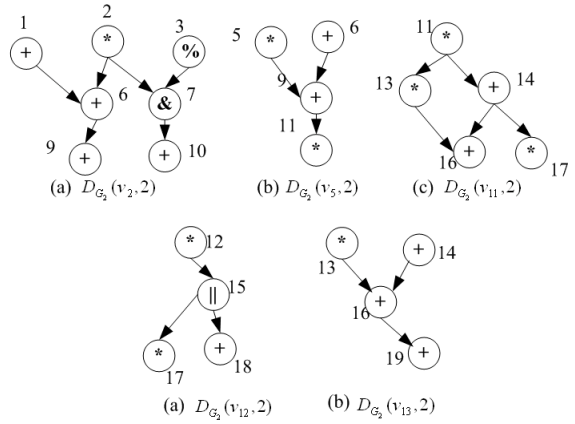

Fig.2: The Directed circles of G_2 in fig. 1

Definition 3.2: $G(V,A)$ is a directed graph, G' is the basic graph of G, $R_v = \{u \mid u \in V,\ d_{G'}(u,v) \le r\}$. The induced graph $G[R_v]$ is defined as a circle with the center v and radius r, and represented by $C_G(v,r)$. The directed graph $C_G(v,r) - V'$ is defined as a directed circle with the center v and radius r, and represented by $D_G(v,r)$, where

$$V' = \{u \mid u \in R_v,\ \exists a = < u,v > \in C_G(v,r)\}.$$

Fig.2 shows the directed circles of G_2 in fig. 1. The radius of the circles is 2 which equal to the eccentricity of v_0.

The matching nodes of v_0 should have zero in-degree and the same eccentricity as v_0 in the isomorphic subgraphs, so the isomorphic subgraphs can be searched in the directed circles. By preprocessing, it can be find that there are not sub-

graphs isomorphic to G_1 in $D_{G_2}(v_5,2)$, $D_{G_2}(v_{11},2)$ and $D_{G_2}(v_{12},2)$.

3.4. Labeling

Upon completion of phase 3, a set of directed circles possibly containing isomorphic subgraphs is chosen, and the set is represented by Ω. In phase 4, all the directed circles in Ω are examined to find the isomorphic subgraphs by labeling. The breadth-first manner is used during labeling for maximizes the use of the information about the structure of G_1.

The relabeling function is defined as follow:

$$p(v) = p(v)' + \sum_{u \in V^-} p(u) - \sum_{u \in V^+} p(u) \quad (1)$$

Where $p(v)'$ is the label of node v after the last time labeling, $V^- = \{u \mid \exists a =< u, v >$, and being relabeled last time$\}$, $V^+ = \{u \mid \exists a =< v, u >$, and being relabeled last time$\}$.

The vertexes lying outside the isomorphic graph may add extra information to the matching vertexes during relabeling. It make that the matching nodes may have the different label. In order to avoid this situation, the vertexes in C having none matching nodes will not be relabeled in next time, and the label of the vertexes are not used to relabel other nodes, where C is a directed circle. The nodes in C having none matching nodes are named rejected nodes, and the remainder nodes in C are named potential nodes. For example, the vertexes in S which is a subset of C are all rejected nodes, if the nodes have the same label and there are not nodes in G_1 with the label. Suppose that S and T are the vertex subsets of C and G_1 individually, and the vertexes in them having the same label. The nodes in S are all potential nodes under the condition of $|S| = |T|$. Especially, the g is the matching node of s, when $S = \{g\}$, $T = \{s\}$. The labels of matching nodes will not change any more during the process of relabeling.

3.5. Algorithm

Algorithm: detecting all the isomorphic subgraphs
Input: G_1, G_2.
Output: all the subgraphs isomorphic to G_1.
Step1: preprocessing.
Step2: locating, so v_0 and V_0 being gained.
Step3: decomposing and preprocessing every directed circle, and then the set represented by Ω is found.
Step4: labeling the every element of Ω to detect all the recurring subgraphs.

Let $S \in \Omega$, and label S as follows until the all the nodes in G_1 find its matching nodes or the remainder nodes except matching nodes are all rejected nodes:

(1) Label v_0 and the center of S with the same label, and mark that they are matching nodes.
(2) Relabel the potential nodes which are adjacent to the nodes labeled last time using the relabeling formula.
(3) Suppose that $v_i \in G_1, v_j \in S$ have the same unique label, mark that they are matching nodes.
(4) Check the condition, if there are no new potential nodes and no unique nodes in both graph with the same label, choose an pair of unmatched nodes which having the same label, and go to (2); else go to(2) directly.

4. Experiment

This section will experimentally investigate the computational performance of the algorithm and demonstrate the capability of the algorithm on solving subcircuits detection problem.

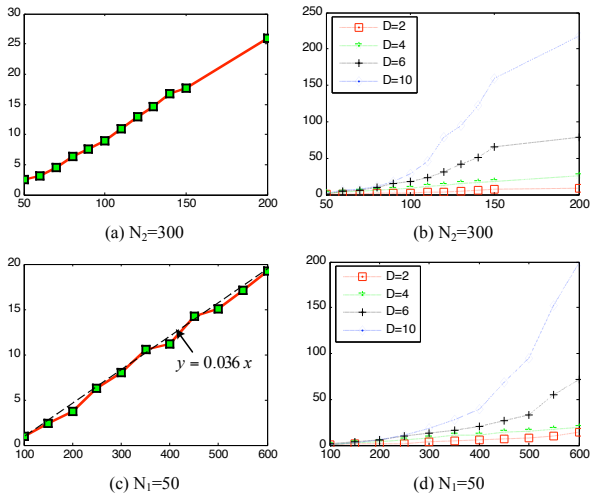

Fig3: The Curves of Runtime Influenced by The Parameters of Graph

In the first experiment, we are interested in measuring the influence of the size of G_1. In the experiment, G_2 has fixed size with $N_2 = 300$, the size of G_1 represented by N_1 was increased from 50 to 200. The curve of execution time varying with N_1 is recorded in fig. 3(a). Fig.3 shows that the runtime increases linearly with the increasing of the size of G_1 when the maximal degree is bounded to 4. As illustrating in fig. 3(b), the rule of runtime G_1 changes when the maximal degree increases. The runtime increases faster with the increasing of maximal degree. In many practical problems the CDFGs of circuits is sparse enough, so the algorithm is a fast in identifying recurring subcircuits.

In the second experiment, we are interested in measuring the influence of the size of G_2. In the experiment, $N_1 = 50$, the N_2 increases from 100 to 600. The results of the experiments are given in fig.

3(c) and 3(d). The rule of runtime varying with the size of G_2 is similar as that of G_1. By comparing all the graphs of fig. 3, the conclusion can be gotten that the runtime increases faster with the increasing of N_1 than that of N_2.

In the third experiment, we are interested in measuring the influence of the number of node types. Fig.4 illustrates the curves, and the runtime decreases with the increasing of types of nodes. Runtime decreases fastest when the number of types is around of half of N_1, and doesn't change almost when the number of types is more than some integer.

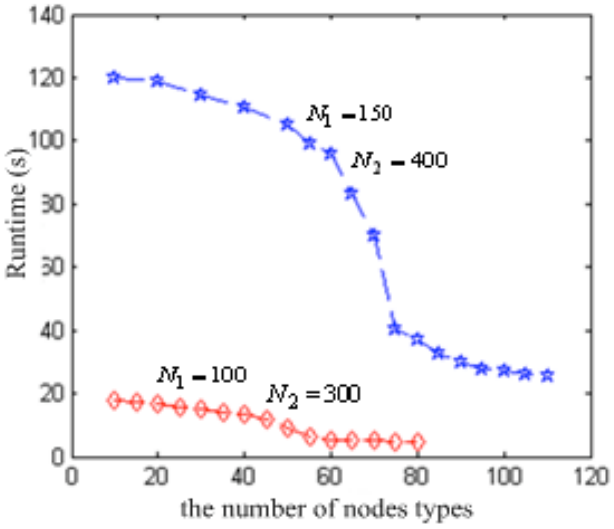

Fig.4: The Curves of Runtime influenced by the types of nodes

5. Conclusion

The major contribution of this paper is the algorithm developed for identifying all the subgraphs isomorphic to a serious of given graphs. The algorithm put forward in this paper can identify any given subcircuits represented by directed graphs in complex circuits. By dividing the process into four phases, and some clever techniques the execution time is saved. Although the algorithm is not polynomial, the run time of the algorithm is influenced by the numbers of vertexes; the numbers of node types, the maximum degree of graph, the run time of the algorithm has linear relationship with the numbers of vertexes when the maximum degree of the graph is not more than 4.

6. References

[1] Michae Boehner. LOGEX-an automatic logic extractor from transistor to gate level for cmos technology [A]. In proceedings of the 30th ACM/IEEE Design Automation Conference[C], 1988, 517-522.

[2] S. Kundu. GateMaker: A transistor to gate level model extractor for simulation, automatic test pattern generation and verification[A]. In proceedings of the IEEE International Test Conference[C], 372-381.

[3] C. E. Miles Ohlrich et al. , Subgemini: Identifying subcircuits using a fast subgraph isomorphism algorithm[A], ACM Press ,31-37, 1993.

[4] Travis E. Doom, Jennifer L. White, Gregory Chisholm, and Anthony S. Wojcik, Identification of functional components in combinational circuits[R], Technical Report ANL/DISfrM-47, Argonne National Laboratory, 1998.

[5] T. Doom, J. White, A. S. Wojcik, and G. Chisholm, Identifying high-level components in combinational Circuits [A], in Proceedings of the IEEE conference of Great Lakes Symposium on VLSI[C], 1998.

[6] J. White, A. S. Wojcik, M. Chung, and T. Doom, Candidate Subcircuits For Functional Module Identification In Logic Circuits[A], Proceedings of the 2000 Great Lakes Symposium on VLSI[C], 2000.

[7] Miguel R. Corazao, Marwan A. Khalaf, Lisa M.Guerra, Miodrag Potkonjak, and Jan M. Rabaey, Performance optimization using templates mapping for datapath intensive high-level synthesis[A]. IEEE Transactions on Computer-Aided Design of Intergrated Circuits and Systems[C], 15(8):877-888., 1996.

[8] J. Smith and G. De Micheli. Polynomial Methods for Allocating Complex Components. DATE [A], Proceedings of the Design, Automation and Test in Europe Conference[C], 217-222. 1999.

[9] Giovanni De Micheli, James Smith, Polynomial Methods for Component Matching and Verification [A], ICCAD'98[C], 678-685, 1998.