SubCircuit Extraction with SubGraph Isomorphism

Zong Ling, Ph. D.

IBM Almaden Research Center — EDA Shape Processing
zling@us.ibm.com

Abstract

The objective of this paper is to demonstrate the practice of the SubGraph Isomorphism (SGI) algorithm on solving the problem of SubCircuit Extraction (SCE). The netlist representation of circuits is discussed first because it has caused the dilemma to the existing SCE approach. Then, based on mapping focus of Edge Unit (EU), an algorithmic solution is exhibited to fulfill the void of circuit comparison in netlist, which is a Hyper Graph structure. In general, by integrating a set of heuristic strategies, a novel planning approach (vs. current searching approaches) for SCE is presented with the advantages of eliminating the non-isomorphic solutions in the early stages and minimizing the number of backtracks on multiple decisions. Therefore, a linear matching time performance of the SCE algorithm is achievable, to the testing circuits that possess EU local distinguishability.

Keywords: SubCircuit Extraction (SCE)
Layout vs. Schematic (LVS)
SubGraph Isomorphism (SGI)
Graph Isomorphism (GI)
Algorithms/Planning
Pattern Matching

1. Introduction

SubCircuit Extraction (SCE) plays an important role in Computer-Aided-Design (CAD) of digital circuits. It sets up the foundation for many practical applications with circuit comparison, such as a special case of SCE problem -- layout vs. schematic verification (LVS), which prevents the circuit designers from leaving the topological design errors until the end of the expensive fabrication process. A more general SCE problem is known as finding the identical copies of a given subcircuit from a larger circuit [E&Z83][OEGS93]. It is often used on determining whether the layout of the circuit geometry corresponds to the specification of the circuit [E&Z83], especially in hierarchical implementation. An common instance, to identify a collection of interconnected primitive devices in a circuit as a single high-level component such as converting a transistor netlist into a gate netlist, involves finding the subcircuits (transistors) and replacing them with the corresponding gates.

An instance is drawn in Figure 1. In (a), the circuit is a NAND gate with four transistors. To identify its isomorphic copy (shadow) from the larger circuit in (b) and replace the copy with the NAND gate, the SCE technique is inevitable.

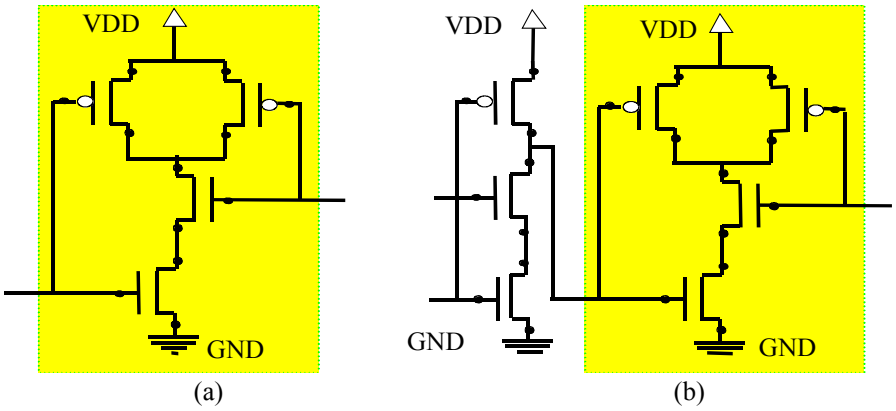


Figure 1 – Problem of SubCircuit Extraction

The existing IC-design tools, such as SubGemini [OEGS93], which is based on Corneil's partition algorithm [C&G70], can currently provide a fast extraction of subcircuits. Nevertheless, there are still some cases that make the tools inefficient and incorrect. For instance, SubGemini is *NOT* able to recognize the shorted-input NAND gates, with the normal NAND as query pattern. When the circuit is very large (more than 96000 devices) and there are highly symmetric structures, SubGemini will spend almost 4 minutes to identify 16000 subcircuits with 6 devices [OEGS93]. Therefore, the more efficient and effective algorithms for SCE are still necessary because the general correctness of SCE is important and the size of circuits will be getting larger and more sophisticate according to the developing trends of VLSI manufacturing technology.

Since the electronic circuits are always represented as graph structures, the SCE problem is naturally an analogy to the problem of SubGraph Isomorphism (SGI). And reasonably, the practice of solving SCE problem should be backed up with the SGI algorithms. Therefore, in the following sections, we present a SGI approach to resolve the SCE problem. Based on the Edge Unit as matching focus, the center and radius for circuit decomposition, the wave front for prefix-propagation, the novel SGI approach can generally solve the SCE problem. It not only fulfills the void of current SCE approaches, but also provides the minimization of the number of backtracks and elimination on the non-isomorphic solutions early. Thus, this SCE approach, guiding by the set of heuristic principles, is expected to enjoy a linear time performance for the input circuit inputs with EU local distinguishability.

In next section, the SCE problem is described and the flaws of current approaches are introduced. In section 3, the algorithmic solution is discussed. Then, in section 4, the novel approach including preprocessing and mapping engine is presented. A set of experimental results is shown in section 5 to demonstrate the correctness and efficiency of the new approach. Finally, in section 6, the there is a summary.

2. Problem Description

As mentioned before, since the electronic circuits can always be represented into graphical structures, the SCE problem is always analogous to that of SGI.

2.1 Problems of SGI and SCE

SubGraph Isomorphism is a well-known concept -- a given graph G_1 is isomorphic to a subgraph H of another given G_2 if there exists a one-to-one mapping of the nodes of G_1 onto the nodes of H such that all corresponding edge adjacencies are preserved. The problem is that of finding an *efficient* algorithm or approach for determining whether one given graph (*smaller graph*, G_1) is an isomorphic subgraph of another graph (*larger graph*, G_2). Similarly, the problem of SubCircuit Extraction can be defined as: determining whether one given *smaller (Guest) circuit* G_1 is isomorphic

to a subcircuit *H* of another given *larger (host) circuit* G_2.

2.2 Existing Approaches

Many SGI algorithms are search-based approaches. They usually take two types of fundamental operations for mapping:

1) *Mapping the nodes one-to-one*;

2) *Preserving the adjacency connections*.

However, in Electronic Design Automation (EDA), some practical restrictions can apply to the existing SGI algorithms.

For an instance, Ullmann's algorithm [Ullm76], which is basically a depth-first search plus refinement strategy, is disable to deal with very large size inputs. The algorithm is actually a brute-force enumeration procedure that is entered after each node in the tree search and the result is generally a reduction in number of successor nodes that must be searched, which yields a reduction in the total computer time required for determining isomorphism. It is not only inefficient due to blindly starting each node for matching but also fall into trouble on space limitation since the refinement needs to occupy huge memory while generating the submatrices. Their published experimental results on subgraph isomorphism did not exceed the graph size with 14 nodes, and even in current implementation platform (such as SGI Indigo workstation with MIPS R4400 processor, 132Mbytes memory), matching between two graphs with 100 nodes is still unfeasible. Obviously, this algorithm is unacceptable to the EDA field where the inputs could be large as the graphs with more than 10k nodes.

Another graph matching algorithm based on search approach is Corneil's algorithm [C&G70], which is a typical breadth-first search during procedure of re-labeling the nodes associated with its neighbor nodes. It originally focused on Graph Isomorphism problem (which supposed two input graphs are size-equivalent, therefore it is a special case of SGI problem) and then worked on SCE problem after practical modification. Now, it is the kernel of SubGemini [OEGS93] -- the SCE software today. In the preprocessing step, this algorithm works on a representative and reordered representation of both input graphs, and the problem is solved based on the transformed graphs. This procedure can only provide an incomplete answer to the isomorphism problem and for some input it can not decide whether the two input graphs are isomorphic [OEGS93][Dorr95]. More seriously, an intractable dilemma for SubGemini is that how to correctly detect the logic gates with shorted inputs. This problem seems unsolvable with Corneil's algorithm since it is created by netlist representation of circuits.

2.3 Netlist of Circuits

Netlist of a Circuit

Electronic circuits can be represented as labeled graphs [B&A89]. A simple model for circuit structures is a specialization of a type of graph known as a *hypergraph* [G&T87], where the nodes correspond to the devices and the connections (between devices) to the nets [Mic94]. In netlist representation, a circuit is an undirected bipartite graph with devices forming one set of nodes and nets (wires) forming another set of nodes. As an example, a normal NAND gate in text file format is shown as follows:

```
SUBCKT NAND OUT AIN BIN
MP1 OUT AIN VDD VDD P
MP2 OUT BIN VDD VDD P
MN1 OUT AIN INT VSS N
MN2 INT BIN VSS VSS N
ENDS NAND
```

In this text file, the first row is the headline, and the last row is the tail line. Between the head and tail, MP_i and MN_i, i = 1, 2, are the titles of two P-channel and two N-channel transistors. Each transistor has four terminals: The drain, gate, source and substrate, which are accordingly represented in each row after each corresponding title of the transistors. Since the pin of substrate only belongs to the CMOS transistor, without losing generality, we only consider the three basic terminals of

transistors here. That is, the substrate pins of the transistors are ignored in the following description.

This NAND gate contains four inter-connected transistors as shown graphically in Figure 2 (a). Symbolically, OUT could be both internal crossing point (net) and the output point of this NAND. AIN and BIN are two nets and the two inputs of the NAND. VDD and VSS are the points of power supply and ground respectively. INT is an internal net.

This NAND could also be equivalently represented as a netlist in Figure 2 (b). In the netlist representation of this NAND gate, the devices (transistors) are represented with the *device nodes* (circles), and the connections or the physical crossing points are treated as a type of special nodes, *nets* (dots). The labels of device nodes are the types of devices based on the device functions such as P-channel or N-channel transistors here. The nets might be the types of cross points such as VDD, GND or CLOCK. The *links* are the connections between devices and nets through device terminals. The link labels are the names of the device terminals, such as drain (d), source (s) and gate (g). By treating the CMOS transistor as a switch, we can label the source and drain with equivalent label "s" the same way that we usually do not distinguish the terminals of a switch.

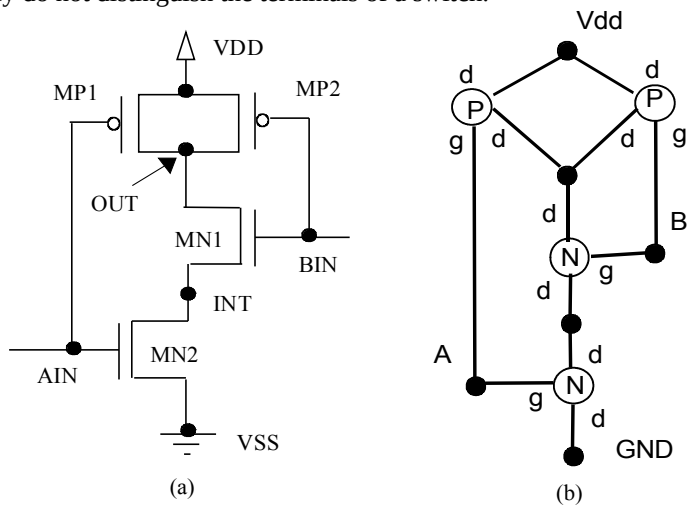
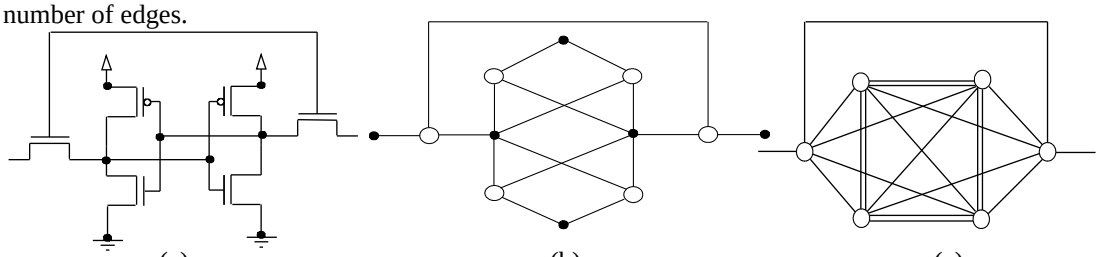


Figure 2. An Circuit and its Netlist Representation

Advantage of Netlist

One of the major benefits of a netlist representation is the reduction of the number of edges, which is the dominant factor for space utilization. This representation is compact, using only O(N) space for a set of N fully connected circuits instead of $O(N^2)$ as required if only devices are represented. In another word, for a fully connected circuit with N device nodes and N(N-1)/2 edges, the inserted net that could represent the connections between devices, will reduce the edge number into N. On the other hand, for a ring-type circuit with N devices and N edges, the inserted nets would increase the "node" number to 2*N and keep 2*N links.

A normal example of RAM Cell with 6 transistors (device nodes) in different representations is shown in Figure 3 (a)(b)(c). Its netlist representation has totally 17 links with 6 nets. However, the graph representation without net nodes would hold 21 edges. As can be seen, the netlist representation not only simplifies graph representation, but also possesses the economic advantage of reducing the number of edges.



(a) A RAM Cell 6 Devices

(b) Netlist Representation 6 Nets & 17 links

(c) Graph Representation 21 Edges

Figure 3 – An Example of RAM Cell

In our daily experience, any benefit would be associated with a cost. Such is the case with netlist representation of circuits. The class of nets in a circuit netlist representation creates a problem for subcircuit extraction with conventional SGI algorithms, since the nets are not the *nodes* in terms of conventional graphical representation and are actually the *connections* among the graphical nodes. In the electronic circuits, they are indeed the edges or connections among devices.

A Dilemma for Conventional SGI Algorithm

Supported by the SGI statement and concepts, in SCE algorithms, it usually takes two types of fundamental operations for circuit mapping:

1) *Mapping the nodes one-to-one*;

2) *Preserving the adjacency connections*.

It would be true that these two operations are correct for SGI mapping based on a general graph representation and any SGI algorithm could be applied on solving SCE problem. However, in the domain of circuit design, there is a significant difference from conventional SGI problem: the graphical structure of circuits is usually the netlist representation, which is actually the *Hyper Graph*.

In fact, the netlist is a bipartite graph. The device nodes can only have edges to nets, and vice versa. Specifically, the nets in netlist has a critical feature: No *connections can bridge nets*. Because of this feature, any connection between *two* nets will merge them into *one*. In another words, the number of nets would be reduced after connecting the nets together.

Let us consider the case that a NAND gate with shorted inputs.

A 2-input NAND gate in circuit and block format is shown in Figure 4. Its two inputs are normally separated and might be connected to other circuits on demand.

As shown in Figure 4 (a), G_2, the NAND gate with shorted inputs, means another external connection is appended onto the normal NAND gate G_1. From the electronic circuit perspective, G_1 is certainly isomorphic to a subcircuit of G_2 since this G_2 is exactly the G_1 *plus* one extra connection -- the shorted link between two inputs. The *set* of *devices* and *connections* in G_1 is clearly the s*ubset* of that in G_2. That is, the normal NAND gate *is* a subcircuit of the NAND with shorted inputs.

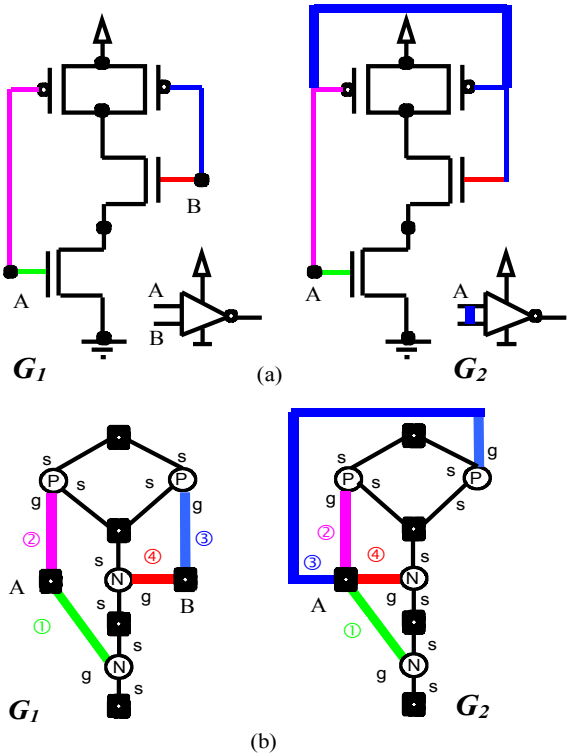


Figure 4 - A NAND with Shorted Inputs

However, in the netlist representation, as illustrated in Figure 4 (b), when two inputs are connected, the *two* originally separate nets (dots) in the normal NAND gate is merged into *one* net. That is, appending this one extra connection decreases the number of nets. The set of *nets* in G_1 is no longer the *subset* of that in G_2.

Consequentially, it is natural that, by treating the nets as *nodes*, the conventional SGI algorithm will inevitably fail in mapping the *nets* one-to-one in this special case. In the example depicted in Figure 4 (b), if G_1 is treated as the query pattern, by mapping G_1 onto G_2 with the conventional SGI algorithm, the *incorrect* SCE solution would be generated according to the node number inconsistency: G_1 is *not* isomorphic to any subcircuit of G_2.

2.4 Flaws in SubGemini

It has been reported that SubGemini would fail on the correct recognition of NANDs with shorted inputs [OEGS93]. SubGemini uses Corneil's partitioning algorithm for subcircuit extraction. This partitioning is done implicitly by assigning a label to each node in the graph and the labeling is done using node invariance, which are properties that are maintained under an isomorphism. The device type is the invariant for devices and the number of connections is the invariant for nets. An initial partitioning is done using these invariants as labels. Then the partitioning is refined by re-labeling each node based on the labels of its neighbors, suitably modified by the terminal classes used to make the connections.

In Corneil's algorithm, the re-labeling procedure needs to check the node consistency between two input graphs. If the *node number* equivalence could *not* be satisfied, the mapping procedure will be terminated with the solution of non-isomorphism. Gemini adopted this algorithm first for *circuit* comparison and then SubGemini used it for *subcircuit* extraction after practical modification. Thus, the merged net nodes make the node number equivalence or node consistency impossible.

Furthermore, it is known that the problem of Graph Isomorphism, which the sizes of two input graphs are supposed to be equivalent to each other, is a sub-problem of SubGraph Isomorphism. General speaking, an algorithm for solving special case in a problem space, such as Corneil's algorithm for graph isomorphism, can not be generally re-planted to solve entire problem (the subgraph isomorphism problem here). Plus the hyper graph – netlist representation of circuits, SubGemini will not be able to recognize the shorted-input NANDs successfully and resolve the SCE problem completely.

Patching SubGemini

Currently, there are two solutions for solving this problem: 1) Construct a pattern library which has exhaustively collected the NAND gates with different types of inputs and then repeatedly try these patterns one by one to extract the corresponding NANDs in the larger circuits. 2) Prefigure the external nets before matching procedure the using Gemini for internal net matching and then specially matching external nets. Both of the above approaches have the drawbacks to extend the application scope of SubGemini.

For the first approach, consider the cases that the N inputs of a NAND are only shorted with each other. Then, the total number of different shorted types would be calculated by

$$\sum_{j=0}^{N} C(N, j) - C(N,1) = 2^N - N$$

where, C (n, k) = n! /[k! *(n-k)!] .

For example, a normal 4-Input NAND would have 11 different types of mutual shorted inputs. That means, to extract the 4-Input NANDs from a larger circuit, a pattern library containing at least 12 pattern NANDs is needed! According to this formula, when the input pin number of NAND gates is increasing, the size of the pattern library would exponentially grow up. If there were some shorted inputs that are connections to VDD or GND, the size of the pattern library would be much larger.

Apparently, this situation is conceptually unacceptable and practically inconvenient.

Then, for the second patching approach, consider the case that the pattern is composed of sophisticated gates, some internal nets inside the guest circuit could be the external nets of some gates, i.e., these internal nets are still possible to be shorted in the host circuit. In another word, the external (possible shorted) nets might not be known before matching. Intuitively, this approach is matching the internal nets with graph isomorphism, which request equal topological property in both guest and host circuits, but specially matching external nets with subgraph isomorphism. It is unpractical to prefigure the external nets in general cases. Therefore, this approach can be only used to extract the simple guest circuit with ***single*** gate or device. It seems hopeless to be generalized on solving SCE problem completely.

Fundamentally, the set of nets in a netlist introduces a transformation on the graph representation of circuits. A circuit with netlist representation has changed its topologic information by treating the physical connections as net "nodes", but this information is very critical for isomorphism and should be preserved for SCE in the special cases. To resolve this problem completely, one potential way is to recover netlist representation into the standard graphic representation, i.e., to remove the net nodes from the netlist representation of circuit graphs. However, in the VLSI design field, the netlist is an industry standard to represent circuits and has the advantage of economical memory utilization. Therefore, an algorithmic approach for the SCE problem on both general-case and special-case has been urgently needed. The EU-focused SGI algorithm, based on the edge unit mapping at each of iterations, could be applied on solving this problem.

3. Algorithmic Solution

As shown in Figure 5 (a), an EU is defined as an *edge* associated with its *two terminal nodes*. Based on the netlist representation, the EU in circuit graphs is the *connection* associated with *two devices* as shown in Figure 5 (b). The net is merely a part of the corresponding EU. This is the key prescription to remedy the headache introduced by the nets of netlist.

Figure 5 - An EU in Netlist Representation of Circuits

To describe the mapping procedure with the EU-based SGI algorithm, let us consider the example shown in Figure 6.

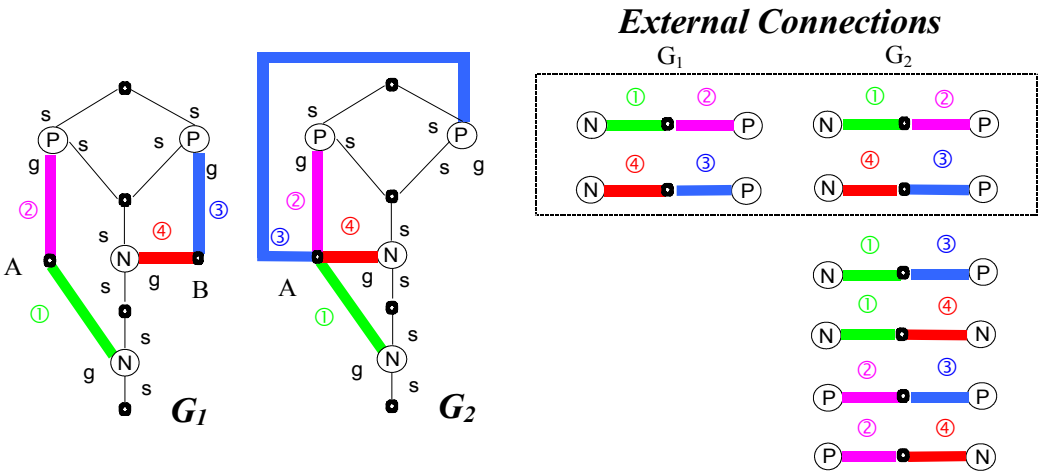


Figure 6 - External EU Mapping

Here, the normal NAND is G_1 and the NAND with shorted inputs is G_2. Our objective is to extract the subcircuit, which is isomorphic to G_1, from G_2. To simplify the description, only the input connections, which contain the nets of inputs, are taken into mapping consideration. Since the internal data structure is still the sets of links and each of the links connects one device and one net, a pair of incident links (sharing one net only) is treated as a complete EU in one single local mapping step. Thus, as listed in the right side of Figure 6, the TEU of links ①+②in G_1 has to be mapped to the SEU of links ①+②in G_2 and so with the mapping between the EU pair of links ③+④in both G_1 and G_2. Due to the shorted connection, there are four more input connections in G_2, links ①+③, ①+④, ②+③and ②+④, but it does not really matter because we only need *subgraph* isomorphism. That is, once the set of input connections in G_1 is a *subset* of that in G_2, we can make the conclusion that G_1 is isomorphic to a subgraph of G_2. The result is expressly concluded that this G_2, which is a NAND with shorted inputs, contains a normal NAND in the type of G_1. Although the number of input nets is different in G_1 and G_2, the actual connections between the corresponding devices are consistent for SGI or SCE.

In this way, the devices have been mapped one-to-one between G_1 and G_2, and the nets are treated as the connections between devices. According to the definition of SGI, the one-to-one device mapping is strictly satisfied and the adjacency connections are consistently preserved as well.

In essence, according to the definition of Edge Unit, only the direct connections between a pair of neighbor devices can be treated as one complete Edge Unit and the nets can merely provide a crossing points of EUs. This exactly restores the original meaning of nets, as the connections among devices in electronic circuits. Moreover, the mapping operations are still based on the netlist representation.

4. Approach for SubCircuit Extraction

Generally, supported by the SGI algorithm [Ling98], an algorithm for SCE is presented in the following sections.

4.1 Terminology

There are some graphic terms we will use in the following sections.

Edge Unit (EU): is an edge associated with its two terminal nodes.

Distance $d(n_1, n_2)$: of two nodes n_1 and n_2 is the length of shortest path (i.e. the number of edges in the shortest path) between them.

Eccentricity E(n): of a node n is the distance from n to the nodes farthest from n in G; that is,

$$E(n) = \max d(n, n_i), \qquad n_i \in G$$

Center: of G is a node with minimum eccentricity in G.

Radius: of the graph is the eccentricity of a center (which is the distance from the center of the graph to the farthest node of the graph).

Path: is a finite alternating sequence of nodes and edges, beginning and ending with nodes, no edge or node appears more than once, such that each edge is incident with the nodes preceding and following it.

4.2 Mapping Focus

Usually, on solving the problem of graph isomorphism, the step-by-step operational procedure is repeatedly composed of two key operations:

1) Map nodes one-to-one,

2) Check adjacency connections between the mapped nodes.

Therefore, in order to store the minimal but sufficient local information of isomorphism, not only node information, but also the connection between the nodes have to be kept. Consequently, in each step of this mapping approach, the mapping focus is an *Edge Unit*.

4.3 General Description

As it's shown in figure 7, we propose to separate the procedure of subcircuit extraction into

two stages: 1) preprocessing and 2) matching engine.

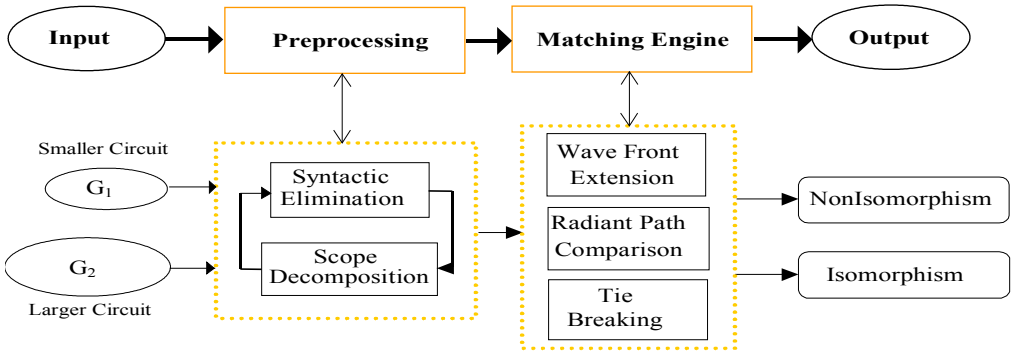


Figure 7. An Approach for SubCircuit Extraction

Here, a set of heuristic strategies is developed to enhance the efficiency and effectiveness in average cases. Conceptually, five strategies are developed here: 1) Syntactical Elimination Model, which is operated as a non-isomorphism detector, so the obvious non-isomorphic situations would be eliminated at the earlier stages; 2) Scope Decomposition, which targets to select the promising starting points and shrink the further search space with the knowledge of input topology and the technique of syntactical elimination; 3) Wave Front Extension, which aims to narrow the search range according to the connected enumeration, so the isomorphic components in both input graphs would be consistently expanded; 4) Radiant Path Comparison, which assists to verify the isomorphism and to exclude the invalid candidates by checking the local and global topological consistency; and 5) Tie Breaking, which provides the global perspective of resource utilization for avoiding the randomness of multiple choice selection by following the principles of Constrained Resource Planning models.

4.4 Preprocessing

Preprocessing procedure is composed of two stages: 1) Extract the starting point in both G_1 and G_2 and 2) Detect non-isomorphism around each of the starting nodes in G_2 and decompose G_2 into a set of smaller areas (subcircuits) $\{H_i$, i = index of the possible starting nodes in $G_2\}$. The succeeding dynamic mapping can be carried out between G_1 and H_i around each corresponding starting node in G_2. If one subcircuit in H_i is isomorphic to G_1, then the mapping procedure would be terminated for single solution SCE problem or would be continued to find remain isomorphic subcircuits until multiple (all) solutions are extracted. If no isomorphic subcircuit is found within this H_i, the next starting node will be tried until each possible H_i is exhaustively visited.

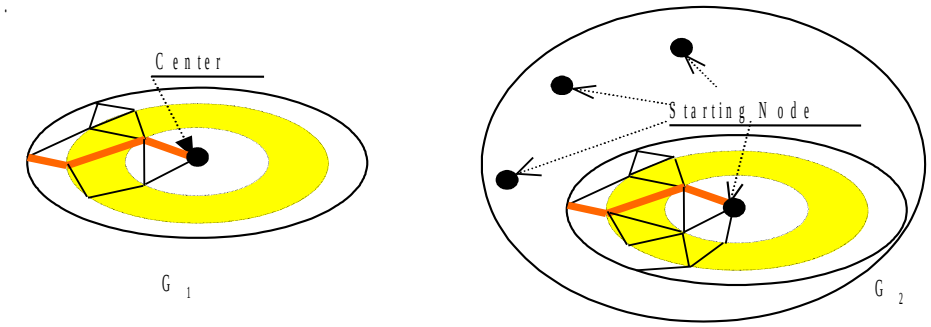


Figure 8 - Preprocessing

4.5 Matching Engine

From the view of operational research, the SCE problem can be treated as a resource management instance. The matching procedure is the process of task scheduling associated with the concern of resource utilization. CRP [Yun92] is a domain independent heuristic methodology. Based on the six concepts: *resource, task, constraint, solution, criticality and cruciality*, it treats a given planning problem as composed of many tasks to be performed, under the condition that resources are constrained.

With the inherent characteristics of incorporating the natural "*demand-pull and supply-push*" economization of resource utilization, CRP model has demonstrated its ability to "achieve high search efficiency" that it is acknowledged to "constitute a good benchmark" [S&F90] [Sadeh90]. Depending on *demand curve*, a unified metric to represent resource contention and load distribution, the planner creates a demand window used for expressing the flexibility of each of the CRP's tasks. Because of always taking a global view of resource contentions and resource utilization during the process, the number of backtracks is minimized.

Terminology

Some of special terms using here will be defined as below.

Resource: all unassigned EUs in host circuit.

Task: any unassigned EU in guest circuit.

Agenda: the set of active tasks.

Solution: an assignment of EUs in host circuit to one EU in guest circuit.

Valid solutions: the EUs in host circuit that each of them

(1) is not assigned yet by any EU in guest circuit after assignment so far;

(2) is of node degree (number of branches) that is equal to or larger than that of task EU;

(3) is of the identical label as the label of task EU.

Solution space: the collection of all valid solutions of a task EU.

Constraints:

(1) Each node (with degree d_1) in guest circuit have to be assigned to the node (with degree d_2) in host circuit such that they have same label and $d_1 \leq d_2$.

(2) Each edge in guest circuit should be identical to the edge in host circuit by label and correspondent to the assigned nodes.

Operational Procedure

Initially, the EUs in host circuit will be treated as resource units, all EUs in guest circuit are considered as active tasks in agenda. Depending on the criticality function, the matching engine will filter out agenda for task identification. Since a task EU in guest circuit with the minimum available candidate solution EUs in host circuit has the least flexibility, it is deemed to be most critical. Then, the set of valid solutions will be scanned over for solution selection according to the cruciality function. The EU in host circuit impacted on other active task EUs in guest circuit at the least will be selected as the least impact solution.

The algorithm can be simply summarized as follows:

0) Preprocessing

1) Task Identification

1.1) Compute criticality;

1.2) Take the task EU with minimum value of criticality function;

2) Solution Selection

2.1) Compute the cruciality;

2.2) Select the solution EU with minimum value of cruciality function;

3) Constraint Propagation

If there is no solution EU matching the task EU, then backtrack;

Otherwise, modify the agenda (remove the selected task) and the availability of resources for other EUs in host circuit and propagate any changes to the candidate solution EU set of each remaining task EUs due to the constraints caused by the current solution EU selection.

4) If some of task EUs can not be assigned to the solution EUs, it is not an isomorphic subcircuit;
If all task EUs are assigned to a set of solution EUs, an isomorphic subcircuit is found;
otherwise go to Step 1).

Policy Functions

1) *Criticality*: {number of valid solutions}

The total number of valid solutions of each task is counted and compared, the task EU corresponding to the smallest number of solutions will be chosen as the most constrained task.

2) *Cruciality*: {number of influenced tasks}

For each solution in the solution space of the most constrained task, count the number of influenced tasks by going through the solution spaces of other tasks, and checking whether any conflicts(*) exist and counting the number of conflict tasks. The smallest number will correspond to the least impact solution.

(*) Meaning of conflict: the other tasks can possibly choose this solution as their own least impacted solution; e.g., this solution also appears on other task's solution space.

3) *Tie-breaking*: (radiant paths)

If there are multiple valid task/solution EUs (tie) in guest/host circuit corresponding to the solution/task EU, the tie will be broken by checking the consistent along the radiant path (shortest path) from the center to both nodes of the EU.

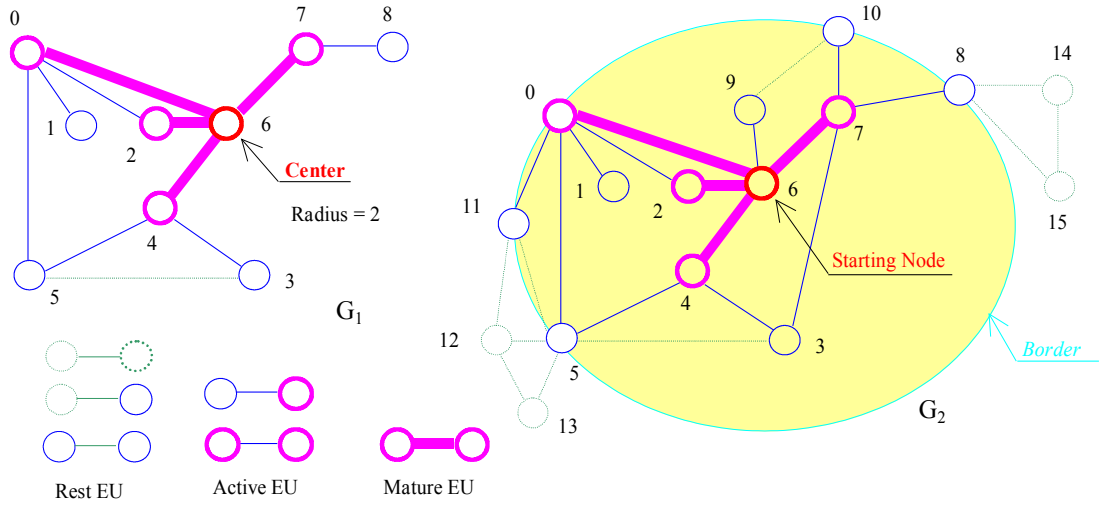


Figure 9 – Snapshot of Matching Procedure

Due to preprocessing and four-corner loop iterations above, the time wasted on non-isomorphic matching can be reduced dramatically by eliminating the non-isomorphic subcircuits early and the number of backtracks can be minimized by identifying most constrained task and selecting least impact solution

5. Experimental Results

This set of simplified experiments is designed to validate the algorithmic solution provided by the presented SGI algorithm on solving the problem of correct recognition of the NAND gates with shorted inputs. The implementation is conducted with C-code on SUN 490 workstation. In the following descriptions, first, the circuits in text file format are transformed into graphs in netlist representation. Then, the set of simple input circuits is introduced, and finally, the test results are discussed.

Data Format Conversion

In logic circuits, the node labels are the device names, such as P-channel and N-channel transistors, and the link labels are the device terminal names, such as the gate/source/drain of the

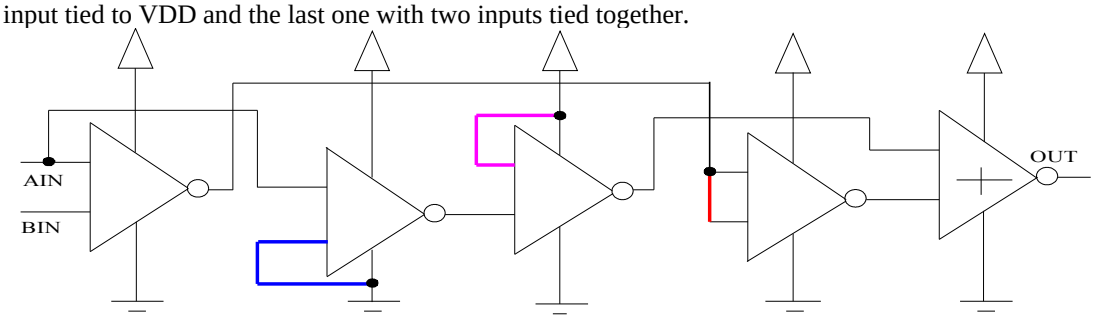
transistors. For a larger logic circuit, which contains many intra-connected gates, the inputs of one gate might be the outputs of other gates. Therefore, the input/output "net nodes" should be labeled the same as the connection "net nodes," i.e., the external net nodes and the internal net nodes are always labeled with same numbers in the following experiments. The special nodes VDD and GND are also labeled the same as connection nodes since they are indeed the connections among the device nodes.

Input Circuits

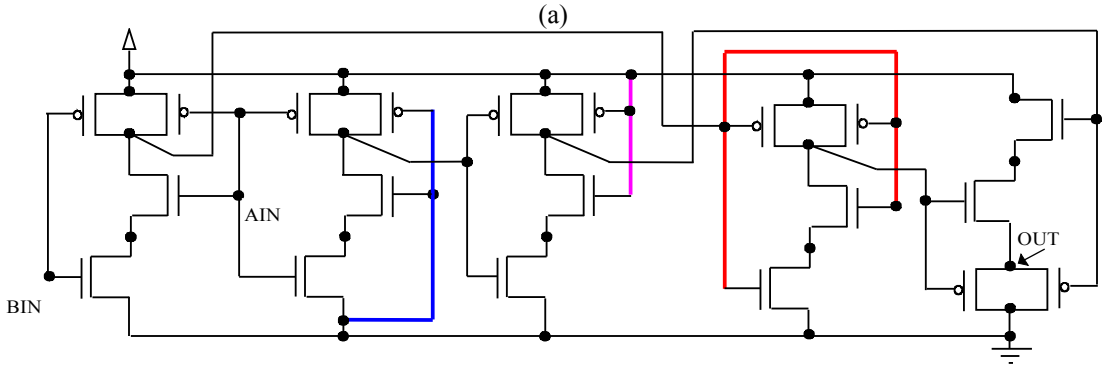
A small set of input circuits is composed of a larger circuit G_2 and a set of smaller pattern circuits $\{G_1\}$. The larger circuit G_2 is named TEST_NAND. Its text file is formatted as follows.

This circuit is drawn in Figure 10. It contains 4 NANDs and 1 NOR, where (a) is the logic block representation and (b) is the circuit representation.

Among these 4 NAND gates, one of them (the left most) is a normal NAND. The remainder three have shorted inputs: The first one from left with an input tied to GND, the second one with an input tied to VDD and the last one with two inputs tied together.



(a)

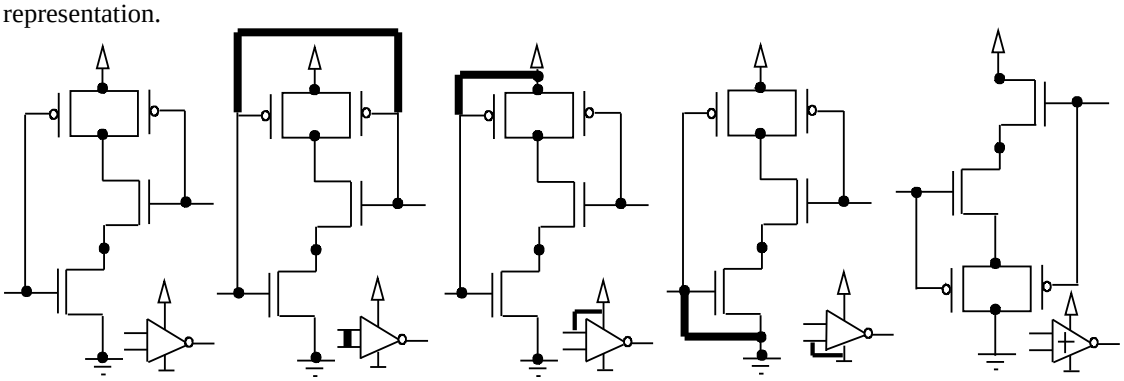


(b)

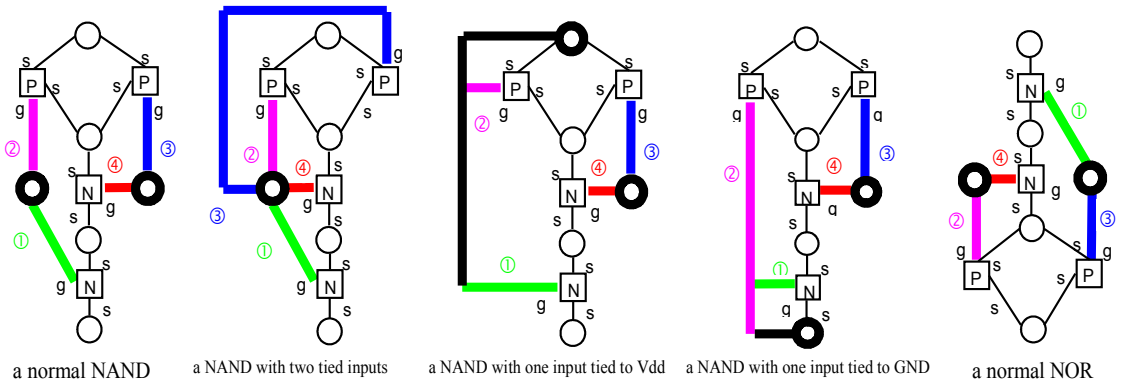
Figure 10 - The Larger Circuit in the Experiment: TEST_NAND

The set of query pattern circuits is composed of the normal NAND, the NANDs with different types of inputs and the NOR.

As shown in Figure 11, (a) is the set of pattern circuits, and (b) is their corresponding netlist representation.



(a)



(b)

Figure 11 - Pattern Circuits

Results

In order to focus on the correctness of shorted NAND recognition, this set of experiments is separated into two groups: 1) The normal NAND as pattern is used to identify both normal NAND and shorted NANDs; 2) The NANDs with different types of shorted inputs as patterns are used to extract the corresponding shorted NANDs. The results should be consistent with the previous analysis.

With the EU-based SGI algorithm, using the normal NAND as pattern, not only the normal NAND in the TEST_NAND is identified, but also the other *three* shorted NAND are located. Therefore, to the question of how many NANDs are in this circuit, the answer of 4 is correctly reported. This is the major improvement from SubGemini, whose answer is *one* in this case. In the execution of SubGemini, those NANDs with shorted inputs are incorrectly missed.

By querying with the individual shorted NANDs, those shorted NANDs, which are in the same type of shorted inputs as the pattern, can be respectively extracted with the presented SGI algorithm. For example, if the pattern is the NAND with one input tied to VDD, then only the NAND with one input tying to VDD in TEST_NAND is recognized. This is similar to the solution of SubGemini.

Additionally, the NOR gate can be only extracted with the NOR pattern. That is, there is no mistake for recognizing NAND and NOR no matter what the input types of NANDs are. SubGemini can also achieve this result.

6. Summary

This paper has presented a solution to the practical issue of subcircuit extraction where the SGI algorithm can be directly applied. The flexible EU-focus possesses the capability of tackling the net nodes in netlist representation, and makes the logic subcircuit extraction successful in both general and special cases. It is significant that not only a long-standing problem of shorted NAND gate recognition is completely resolved, but also a powerful weapon to attack the circuit matching problems caused by netlist representation is uniquely invented. Consequently, the optimization of pattern library in circuit design would be immediately feasible and the automation of VLSI-CAD tool kits would be eventually promising.

In this paper, a new approach for extracting subcircuit (guest circuit) in a larger circuit (host circuit) is introduced. Based on preprocessing guest circuit, transforming subcircuit extraction problem into the resource management paradigm and integrating a set of policy functions, the novel approach could minimize the number of backtracks and eliminate the non-isomorphic subcircuits early so that the time wasted on unsuccessful matching could be reduced dramatically. Because of decomposing pattern circuit topologically, the presented approach can be independent to the circuit structures so that the inefficiency of SubGemini while the circuits are highly symmetric and of some kind of shorted input or output could be avoided. Due to the EU local distinguishability in the input circuits, a near linear time operational performance is achieved. The efficient approach presented here to solve subcircuit extraction problem can be extended to various branches in VLSI design field if the

way to solve the specific problems could be reduced into the matching procedure among graphical structures.

7. References

[B&T88] A. D. Brown and P. R. Thomas, *"Goal-Oriented Subgraph Isomorphism Technique for IC Device Recognition"*, IEE Proceedings, Vol. 135, Pt. 1, No.6, DEC., 1988, 141-150.

[C&G70] D. G. Corneil and C. C. Gotlieb, "*An Efficient Algorithm for Graph Isomorphism*", J. Assoc. Compute. Mach, Vol.17, No. 1, Jan.,1970, pp. 51-64.

[Epps94] D. Eppstein, *"Subgraph Isomorphism in Planar Graphs and Related Problems"*, Tech. Report 94-25, University of California, Irvine, May 31, 1994

[Ebel94] C. Ebeling, "*The Gemini User Guide*", Tech. Report, Department of Computer Science and Engineering, University of Washington, Mar. 15, 1994

[E&Z83] C. Ebeling and O. Zajicek, *"Validating VLSI Circuit Layout by Wirelist Comparison"*, Proceeding of the Conference on Computer Aided design (ICCAD), 172-173, 1983.

[K&Y89] N. P. Keng and D. Y. Y. Yun, "*A planning/Scheduling Methodology for the Constrained Resource Problem*", Proc. 1989 International Joint Conf. on Artificial Intelligence, pp. 20-25, Aug. 1989.

[Ling98] Z. Ling, "*An Algorithm for SubGraph Isomorphism based on Resource Management with Applications*", Ph.D. Dissertation, University of Hawaii, 1998.

[OEGS93] M. Ohlrich, C. Ebeling, E. Ginting, and L. Sathe, *"SubGemini: Identifying SubCircuits using a Fast Subgraph Isomorphism Algorithm"*, 1993, 30th ACM/IEEE Design Automation Conference.

[R&D77] R. C. Read and D. G. Corneil, *"The Graph Isomorphism Disease"*, Journal of Graph Theory, Vol. 1 (1977) 339-363

[Ullm76] J. R. Ullmann, *"An Algorithm for Subgraph Isomorphism"*, J. Assoc. Compute. Mach, 1976, 16, 31-42

[Yun92] D. Y. Y. Yun, *"CRP - an Intelligent Engine for Resource and Production Management"*, an introductory system/tool for solving a large variety of planning and scheduling problems was exhibited at COMDEX '92 during the week of Nov. 16 - 20 and was nominated for a Byte Magazine award.