UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

Conception et implémentation d'un outil de filtrage de netlist pour un système de prototypage rapide

MÉMOIRE PRÉSENTÉ

COMME EXIGENCE PARTIELLE

DU PROGRAMME DE MAITRISE EN SCIENCES ET TECHNOLOGIES DE

L'INFORMATION

PAR

BARATLI KARIM

Décembre 2012 (17/12/2012)

Jury d'évaluation

Président du Jury :	Dr Marek Zaremba
Membre du Jury :	Dr Mohamed Bougataya
Directeur de recherche :	Dr Ahmed Lakhssassi

Dédicace

À la mémoire de ma mère Najoua.
À mon père Mohamed.
À mes frères Oussama, Moez et Hamdi.
À mon oncle Ridha.
À mes amis Nizar, Hassen et Thomas.

Remerciements

Mes chaleureux remerciements s'adressent à mon directeur de recherche Monsieur Ahmed Lakhssassi et Monsieur Yves Blaquière de l'UQAM pour leurs soutiens, leurs directives et leurs conseils. Je tiens à remercier tous les membres de l'équipe de recherche DreamWafer™ ainsi que le personnel de la compagnie TechnoCap à Montréal. Je veux aussi remercier MITACS pour le soutien financier.

Table des matières

1. Introduction 1

1.1. Mise en contexte 1

1.2. Motivations 2

1.3. Exposé de la problématique 4

2. Introduction aux graphes et à l'isomorphisme de graphes 7

2.1. Introduction aux graphes 7

2.2. Appariement dans un graphe 9

2.3. Isomorphisme de graphes 10

2.3.1. Isomorphisme vs homomorphisme 10

2.3.2. Isomorphisme de sous-graphes 13

3. Le WaferBoard™ 15

3.1. Fonctionnement de la carte WaferBoard™ 15

3.2. Description du Workflow 23

3.3. Exigences et spécifications du système WaferBoard™ 25

4. État de l'art 27

4.1. L'extraction de sous-circuits 27

4.2. L'appariement de graphes 31

4.2.1. Isomorphisme de graphe et de sous-graphe 32

4.2.1.1. Appariement exact 33

a) Les techniques basées sur les arbres de recherche. 34

b) Autres techniques 40

4.2.1.2. L'appariement inexact 42

a) Techniques basées sur les arbres de recherche 43

b) Autres techniques 44

5. L'outil de filtrage de netlist 46

5.1. Choix de l'algorithme d'isomorphisme de sous graphe 46
5.2. Conception de l'outil de filtrage de netlist 48
5.2.1. Diagramme de contexte de l'OFN 48
5.2.2. Diagrammes préliminaires 49
5.2.3. Diagrammes de décomposition 50
5.3. Motifs des sous-circuits 52
5.3.1. Les résistances pull-up et pull-down 52
5.3.2. Les condensateurs de découplage 53
5.3.3. Oscillateur à Quartz 54
5.4. Vue d'ensemble de la norme EDIF 54
5.4.1. Introduction 54
5.4.2. Version 56
5.4.2.1. EDIF 2 0 0 56
5.4.2.2. EDIF 3 0 0 56
5.4.2.3. EDIF 4 0 0 57
5.4.3. Structure 57
5.5. Analyse des résultats 59
6. Conclusion 64
Bibliographie 66
Annexe A 73

Liste des Figures

Figure 2- 1: Exemple de graphe et de sous-graphe 11
Figure 3- 1: Composants CI posés sur la surface de la tranche de silicium 17
Figure 3- 2: Illustration de la carte de prototypage connecté à un ordinateur 18
Figure 3- 3: Illustration de la carte de prototypage, le wafer, réticule, cellule [40] 19
Figure 3- 4: Wafer-scale avec des rangées de cellules programmables contenant 4x4 NanoPads™ [40] 20
Figure 3- 5: Connectivité dans le WaferNet™ [40] 21
Figure 3- 6: Présentation du WaferNet™ connectant des billes du composant CI [40] 22
Figure 3- 7: Détection des billes du composant par le court-circuit [40] 23
Figure 3- 8: Flux de travail proposé pour le système de prototypage [15] 25
Figure 5 - 1: La performance des cinq algorithmes sur des graphes connectés au hasard [17] 47
Figure 5 - 2: Diagramme de contexte de l'OFN 49
Figure 5 - 3: Diagramme de flots de données de niveau 1 de l'OFN 50
Figure 5 - 4: Diagramme de décomposition du module Parsing Netlist 51
Figure 5 - 5: Diagramme de décomposition du module Matching Pattern 51
Figure 5 - 6: Configuration d'une résistance pull-up 53
Figure 5 - 7: Exemples de capacité de découplage 53
Figure 5 - 8: Exemple d'oscillateur à quartz. 54
Figure 5 - 9: Performance en termes de temps d'exécution de l'OFN 63

Liste des abréviations, sigles et acronymes

PCB: Printed Circuit Board

JTAG: Joint Test Action Group

FPGA: Field Programmable Gate Array

CI: Circuit intégré

EDIF: Electronic Design Interchange Format

GRB: Gerber format (images of a PCB)

GUI: Graphic User Interface

VLSI: Very Large Scale Integration

LVS: Layout Vs Schematic

CMOS: Complementary metal–oxide–semiconductor

PR: Pattern Matching

Abstract

The information presented in this master's dissertation describes the studies, research and analyses carried out during the conception and the implementation of a netlist pruning tool for an electronic system prototyping platform.

The increasing complexity of electronic systems makes the recognition of subcircuits in main circuits a desirable feature to several Electronic Design Automation tools (EDA).

The netlist pruning tool is a critical element in the development and functioning of the WaferBoard^{TM}, which is a board, used for rapid prototyping of electronic systems. The use of this prototyping board will allow the reduction of both the cost and time involved in the development of complex electronic systems containing high technology components such as: FPGA, ASIC, processors and memories. Users of the prototyping board can place components on the active upper surface of the board, which will then detect the component pins, power them with the corresponding power and tension, and allow intelligible and programmable connections to be made between components. These connections can be modified at any time, allowing the user to add or remove both connections and components.

Relevance of the topic and the research work presents a hand in the utility of the netlist pruning tool netlist in research and industry, subsequently in the utility and role of the tool in the WaferBoard^{TM} then on the other hand in the originality and new technology introduced by the project. The proposed algorithm first parses the netlist representing the main circuits and the subcircuits to extract. Secondly, it uses a subgraph isomorphism algorithm to extract all the subcircuits. Finally, for each subcircuits found corrective actions are triggered to replace those subcircuits.

Résumé

Le travail présenté dans ce mémoire porte sur les études, les travaux de recherches et les analyses qui ont été effectués durant la conception et l'implémentation d'un outil de filtrage de netlist pour un système de prototypage rapide.

La complexité croissante des systèmes électroniques rend la reconnaissance de sous-circuits dans un circuit principal une fonction souhaitable de plusieurs outils de conception assistée par ordinateur (CAO).

L'outil de filtrage de netlist est un élément important de développement et de fonctionnement du WaferBoard™, carte de prototypage rapide des systèmes électronique. La carte de prototypage rapide a pour objectif de réduire le coût et le temps de conception des systèmes électroniques complexes, elle peut être utilisée soit comme carte de prototypage rapide soit comme une carte d'essais ou simulateur réel et physique d'un système électronique en développement. Elle permet à ses utilisateurs de détecter les pins des composants électroniques encapsulés dans des boitiers à matrice de billes (Ball Grid Array, BGA) par l'intermédiaire de sa surface active, mais aussi de les identifier et d'établir des interconnections entre ces composants d'une façon intelligente est programmable ainsi que d'assurer leurs alimentations par les tensions et les puissances appropriées.

La pertinence du sujet et des travaux de recherches réalisés s'exprime d'une part dans l'utilité de l'outil de filtrage de netlist pour la recherche et l'industrie, dans l'utilité et le rôle de l'outil de filtrage intégré à la carte de prototypage et d'autres parts dans l'originalité et les nouveautés technologiques apportées par le projet. Le rôle de l'outil de filtrage se compose de trois parties: il doit permettre à l'utilisateur d'introduire la netlist (en format EDIF ou bien de l'extraire directement de son outil de CAO) du circuit principal et celle des sous-circuits à extraire ; permettre d'identifier tous les sous-circuits existants et finalement permettre d'éliminer ces

sous-circuits tout en conservant les propriétés du circuit grâce à des actions spécifiques.

1. INTRODUCTION

1.1. Mise en contexte

Dans le but de répondre à la demande grandissante du marché et de rester concurrentielle en termes de coûts et de qualité, l'industrie a rapidement fait évoluer et complexifier les systèmes électroniques. L'intégration de sous-systèmes électronique dans un même système, la miniaturisation, l'amélioration des performances et des rendements ainsi que la réduction des coûts de développement et de fabrication sont donc au cœur des recherches. Cependant ces orientations présentent des problèmes et ont fait émerger des contraintes freinant leurs évolutions rapides, comme:

Intégration: la multifonctionnalité et la complexité des systèmes électroniques actuels veut qu'ils renferment en leurs seins de nombreux sous-systèmes. Ces derniers, qu'ils soient des FPGA, des ASIC, des ASIC ou des microprocesseurs, ils renferment des millions de transistors et de circuits. Or une forte densité de transistor et des circuits dans un système augmentent fortement la puissance dissipée posant des problèmes thermiques.

Miniaturisation: la tendance actuelle est de créer des systèmes électroniques toujours plus petits. Ceci passe donc forcément par une miniaturisation des composants et des supports de composant (PCB) augmentant la densité des contacts et des pins mais aussi la densité des connexions entre les composants. Ceci est permis par la réduction de la taille des pistes et de l'épaisseur des connexions.

Performances et rendements: la performance et le rendement d'un système sont garantis par la sélection de composant de qualité ainsi que par leurs combinaisons efficaces et efficientes. La qualité du réseau d'interconnexions dans le système en termes d'intégrité du signal et de puissance joue également un rôle

essentiel dans la qualité et le rendement d'un système. Ces opérations nécessitent cependant de nombreux tests, analyses et simulations gourmands en temps et en argent.

Réduction du coût: l'amélioration des performances des systèmes électroniques est combiné à une réduction des coûts massives dans l'optique de proposer des produits toujours plus concurrentiels. Le temps de développement est au centre de ces réductions de coût dans la mesure où il détermine en grande partie le coût de revient final du produit. Or la durée de développement d'un produit dépend de sa complexité et du nombre de composants qu'il renferme.

Ces quatre composantes sont au centre de tous les nouveaux projets de développement de nouveaux systèmes électroniques. Les coûts de la conception finale de ces derniers sont déterminés en grande partie par la qualité des travaux de recherches et des essais réalisés lors du prototypage et de la conception. Ainsi la simplification des techniques et des outils de prototypage et de conception jouera un rôle majeur dans la réduction du temps de développement et donc des coûts des projets.

1.2. Motivations

Les graphes étiquetés sont des structures de données connues pour leur capacité à représenter des entités complexes. Dans une représentation se basant sur ces mêmes graphes, les nœuds décrivent des objets alors que les arcs représentent les relations existantes entre les objets. Grâce à l'accroissement des performances des ordinateurs au niveau des calculs, la représentation des objets en structure est devenue de plus en plus utilisée dans de nombreux domaines d'application tels que la biologie, la chimie, l'analyse de capture de document, la reconnaissance de symbole ou la détection de sous-circuit électronique [1, 2, 3, 4, 5]. Une des conséquences de l'usage intensif des représentations à base de graphes est l'intérêt croissant pour les problématiques

scientifiques associées que sont la fouille de graphes, la classification de graphes ou la recherche d'isomorphisme.

Alors qu'une famille de problème est basée uniquement sur un graphe, l'autre s'appuie sur deux ou plusieurs graphes. Les problèmes basés sur plusieurs graphes se concentrent en premier lieu sur la question d'égalité ou de similarité de graphes présents. L'égalité exacte des graphes pose un problème, car chaque sommet d'un graphe donné peut correspondre à n'importe quel sommet d'un autre graphe. La correspondance exacte des graphes est formalisée par le problème d'isomorphisme de graphes.

Notre étude se concentre sur les problèmes d'isomorphisme de graphes, ce ne sont cependant pas les seuls problèmes qui affectent la famille des problèmes à plusieurs graphes. Ainsi on retrouve des problèmes d'identification de sous-structure, d'inclusion de structure dans une autre structure, de recherche des sous-graphes identiques entre deux graphes ou la fouille des bases de graphes pour l'identification des sous-graphes fréquents.

Les problèmes d'isomorphisme de graphes peuvent être différenciés en deux catégories les problèmes exacts et les problèmes inexacts. Les isomorphismes exacts veulent que les graphes étudiés soient exactement isomorphes alors que les isomorphismes inexacts tolèrent des variations d'attribut et/ou de structure. Ces derniers sont souvent formulés comme des problèmes de minimisation de distance alors que les isomorphismes exacts s'apparentent à des problèmes booléens. Les algorithmes utilisés pour résoudre ces deux types de problème sont de natures très différentes. En effet, les algorithmes utilisés pour les problèmes d'isomorphismes exacts permettent de filtrer les sommets, en fonction de leurs caractéristiques propres, dans l'optique de diminuer le nombre d'affectations possibles. Une telle approche, dans le cas des isomorphismes inexacts, exclurait des solutions valides. Ainsi les algorithmes applicables aux problèmes exacts ne le sont, généralement, pas pour les problèmes inexacts. Au contraire les algorithmes résolvant les problèmes inexacts

sont applicables aux problèmes exacts, mais présentent des performances médiocres, car mal adaptés.

1.3. Exposé de la problématique

Le sujet de ce mémoire aborde le problème de la recherche d'isomorphisme exact de sous-graphe et propose une application dans le cadre de la détection de sous-circuit électronique dans un circuit plus large. La reconnaissance de sous-circuits est un enjeu majeur qui contribue à la conception assistée par ordinateur (CAO) des circuits électroniques due à la complexité croissante des systèmes électroniques [6,7]. En exploitant des graphes attribués pour représenter les composants et les nets dans le cadre de la recherche d'isomorphisme de sous graphes, nous présentons l'étude et la conception d'un outil capable de filtrer des netlists des Printed Circuit Board (PCB) de sous circuits motif et ainsi de déclencher pour chacun de ces motifs détectés une série d'actions prédéfinis.

Il s'est avéré que cet outil est d'une grande utilité pour le WaferBoard™, plateforme de prototypage rapide de systèmes électroniques qui réduit d'une manière significative le temps de mise sur le marché d'un système électronique [8]. Il s'agit d'un circuit actif intégré à l'échelle même de la tranche de silicium, appelé WaferIC™. Il peut être reconfiguré, selon les besoins, dans l'objectif de créer des liaisons entre tous les circuits numériques déposés sur la surface utile. Cette dernière est composée d'une matrice de plots conducteurs configurables à souhait, appelés NanoPad™. Ces plots se dénombrent par millions afin de garantir une densité d'interconnexion élevée. Une fois assemblé, le WaferIC™ permet d'interrelier n'importe quel NanoPads™ entre eux en s'appuyant sur un réseau de connexions programmables et tolérant aux fautes, appelées WaferNet™. Chaque NanoPad™ peut être programmé pour être soit flottant, soit une entrée-sortie (E/S) numérique soit une source de tension régulée ou enfin une masse [9].

Un système électronique n'est pas seulement composé de circuits intégrés, il renferme différents composants passifs comme les résistances, les capacités, les bobines (inductances), les LEDs et les connecteurs. Le WaferBoard™ n'a pas à supporter ce genre de composantes. En effet, tous les composants passifs peuvent être simulés en configurant les NanoPads™ qui sont en contact avec les billes des circuits intégrés déposées sur la surface du WaferIC™.

Une méthode qu'utilisent les concepteurs de carte électronique pour décrire leurs dessins est l'utilisation des netlist. Une netlist décrit un circuit imprimé. Elle est le résultat de la synthèse logique de la description logique du circuit imprimé. Celle-ci contient une description détaillée de tous les composants dans le dessin et leurs interconnexions. Il existe plusieurs formats de fichiers netlist, en fonction de l'outil de CAO que le concepteur utilise. Dans la suite, nous allons utiliser un format standard pour l'échange d'informations entre les outils de CAO qui est le format d'échange de conception électronique (EDIF). Les informations de conception stockées dans le fichier EDIF forment une structure imbriquée, qui est agencée selon un plan hiérarchique. En effet, la hiérarchie permet de réduire la taille et la complexité d'un schéma électronique.

Pour permettre à l'utilisateur de tester son système électronique avec le WaferBoard™, une nouvelle netlist adaptée aux spécifications du WaferBoard™ et gardant les fonctionnalités du circuit désirées par l'utilisateur doit être générée. Ainsi un algorithme capable d'élaguer la netlist des circuits passifs comme les résistances de pull-up, les condensateurs de découplage, les circuits résonnants et des connecteurs doit être conçu. De plus, ce même algorithme doit générer une nouvelle netlist contenant seulement des circuits intégrés et pour chaque motif trouvé des actions appropriées doivent être déclenchées afin de simuler ces sous-circuits taillés.

Une fois le fichier EDIF fourni par l'utilisateur, un analyseur EDIF va agir sur ce fichier pour extraire les informations utiles et nécessaires pour l'outil d'élagage. Ces informations seront sauvegardées dans une structure de données qui décrit chaque

composant dans les schémas et ses interconnexions avec d'autres composants. La structure hiérarchique devrait être conservée pour offrir à l'utilisateur la possibilité d'utiliser le design pour d'autres objectifs.

Dans la deuxième section de ce rapport, une introduction aux graphes et à l'isomorphisme de graphes est présentée. Le fonctionnement du WaferBoard™ et la problématique sont exposées dans le troisième chapitre. La quatrième section présente l'état de l'art relatif aux algorithmes d'extraction de sous-circuits et d'isomorphisme de sous graphes. La cinquième section retrace les étapes du développement, la conception et les tests de performance réalisés de l'outil de filtrage. Finalement, nous proposons une conclusion à la section six.

2. INTRODUCTION AUX GRAPHES ET À L'ISOMORPHISME DE GRAPHES

2.1. Introduction aux graphes

Dans cette partie, nous caractérisons d'abord les différentes classes de graphes afin d'établir la définition formelle du problème traité dans ce mémoire. Les définitions qui suivent sont reproduites principalement du livre *Graph Theory and its applications* [16].

Définition 1: Un graphe $G = (V,\ E)$ est constitué d'un ensemble de nœuds V et d'un ensemble d'arcs $E \subseteq V \times V$, où un arc (u, v) est un couple de nœuds (les nœuds u et v représentent les extrémités de l'arc (u, v)).

Des graphes permettent la modélisation des relations entre des objets. Formellement, il n'y a pas de contraintes sur l'ensemble des arêtes. Il est notamment possible que E contiennent des arêtes multiples entre deux sommets (et donc identiques) ainsi que des boucles $(e = (v, v))$ qui relient un sommet v à lui-même. En général, on suppose implicitement que des graphes ne contiennent ni boucles ni arêtes multiples. Dans le cas contraire, on le mentionne explicitement. Des graphes contenant des arêtes multiples sont dénommés multigraphes.

La taille d'un graphe n est définie par le nombre de ses sommets. On écrit également $|G|$ et on note $||G||$ le nombre d'arêtes. Chaque arête peut être décrite par les sommets qu'il joint. On dit que les sommets v_1 et v_2 sont incident avec l'arête $e = (v_1, v_2)$. Le nombre total d'arêtes incidentes à un sommet est dénommé son degré. Il existe des graphes orientés et non orientés. Dans les graphes orientés, les

arêtes ont une direction et sont dénommées arcs. À la différence des graphes non orientés, l'arc (v_1, v_2) n'est pas équivalent à l'arc (v_2, v_1).

Il existe diverses classes de graphes ayant des caractéristiques spécifiques. Par exemple, un graphe est appelé k-régulier si le degré de tous ses sommets est une constante k. Un graphe complet est un graphe entièrement connecté. Deux autres classes assez importantes en pratique sont les graphes planaires et les graphes bipartites. Les graphes planaires sont intuitivement des graphes que l'on peut dessiner sans intersections des arêtes.

Définition 2: Les graphes bipartites sont des graphes dont il existe une bipartition de sommets en deux sous-ensembles $(V_1 \cup V_2 = V \; et \; V_1 \cap V_2 = \emptyset)$, telle qu'il n'y ait pas d'arêtes connectant deux sommets appartenant à un seul sous-ensemble. Formellement, une condition suffisante est la suivante:

$$\forall \, (v_1, v_2) \in E: v_1 \in V_1 \Leftrightarrow v_2 \in V_2$$

Définition 3 (Sous-graphe induit par les sommets): Soient $G = (V, E)$ un graphe et $G' = (V', E')$ un sous-graphe de G. Si $\forall \, v_1, v_2 \in V, (v_1, v_2) \in E \implies (v_1 \, , v_2) \in E'$ alors les sommets V induisent le sous-graphe G' dans G.

Dans de nombreuses applications concrètes, les objets, ainsi que les relations, ont des caractéristiques importantes. Afin de les inclure dans le modèle, on affecte des attributs aux nœuds et/ou aux arêtes. Si les attributs sont exclusivement numériques, on obtient un graphe pondéré. Dans le cas plus général, on parle d'un graphe étiqueté ou d'un graphe relationnel valué (ARG).

Définition 4 (Graphe étiqueté): Un graphe étiqueté est un graphe dont les arêtes et/ou les nœuds sont affectés d'étiquettes. Ici, nous empruntons à Horst Bunke la définition formelle d'un graphe étiqueté [30]. Soient L_v et L_E deux ensembles d'étiquettes, respectivement, pour les nœuds et les arêtes. Formellement, un graphe étiqueté est défini par un 4-tuple $G = (V, E, \alpha, \beta)$ où:

- V un ensemble fini de nœuds,
- E un ensemble d'arêtes ; $E \subseteq V \times V$,
- $\alpha: V \rightarrow L_v$ une fonction qui génère une étiquette à chaque nœud, et
- $\beta: E \rightarrow L_E$ une fonction qui génère une étiquette à chaque arête.

Notons ici qu'on peut associer à chaque nœud (respectivement arête) plusieurs étiquettes en définissant L_v (respectivement L_E) comme un ensemble de tuples de valeurs. De plus, les étiquettes peuvent être de différents types (p. ex. lettre, mot, symbole, nombre, vecteur numérique, etc.).

2.2. Appariement dans un graphe

La recherche d'appariements dans un graphe vise à trouver des couples de sommets compatibles dans des graphes généralement bipartites. Ainsi, étant donné un ensemble d'usines et un ensemble de produits, on peut construire un graphe de compatibilité en connectant chaque usine avec les produits qui peuvent y être fabriqués. À chaque arc on attribue le coût de production. On cherche alors à produire un maximum de produits à coût minimum. Un autre exemple classique est la formation des couples à partir d'ensembles de garçons et de filles dont les préférences sont modélisées par les poids des arcs.

Un appariement est un ensemble d'arêtes indépendantes E_i dans un graphe $G = (V, E)$. L'indépendance d'arêtes est définie par l'absence de sommets incidents communs entre elles. Il s'agit d'un appariement de $V_0 \subseteq V$ si tous les sommets de V_0 sont couverts par E_i, c'est-à-dire s'il existe une arête dans E_i qui relie chaque sommet dans V_i à un autre. Une propriété importante est la stabilité d'un appariement. Chaque sommet a une liste de préférences c'est-à-dire les arêtes partant de lui est pondéré et il existe un ordonnancement des poids. L'idée est que l'on ne souhaite pas qu'il y ait deux sommets qui préfèrent échanger leurs affectations

actuelles par le lien direct entre eux (qui n'est pas dans E_i). Un appariement stable existe pour tout ensemble de préférences d'après le théorème du mariage stable.

Un appariement maximal est l'appariement ayant le nombre maximal d'arêtes. Le plus souvent, on cherche des appariements maximaux. Le problème est beaucoup plus facile que le problème d'appariements entre deux graphes. L'algorithme de Hopcroft & Karp [36], par exemple, a une complexité de $o(N^2)$. Dans la littérature, on utilise souvent le terme appariement de graphes bipartites (bipartite graph matching) afin de lever l'ambiguïté avec les appariements entre deux graphes (graph matching). On peut également imaginer des problèmes de recherche d'isomorphismes entre deux graphes bipartites. Dans ce mémoire, nous allons utiliser de préférence le terme d'isomorphisme (voir les sections suivantes) quand cela est possible pour éviter des confusions. Quand nous utilisons le terme appariement, il est utilisé dans le sens du problème d'isomorphisme également.

2.3. Isomorphisme de graphes

2.3.1. Isomorphisme vs homomorphisme

L'isomorphisme de graphes est une relation univoque des nœuds entre deux graphes qui respectent leurs attributs et ceux des arêtes. Dans le cas d'isomorphisme de sous-graphes, on considère deux graphes Get G' de tailles différentes, $|G'| = n' \leq |G| = n$. Afin de déterminer si G' est un sous graphe de G une telle relation est recherché entre les nœuds du graphe G' et tous les sous-graphes du graphe G induits par n' nœuds. De manière générale, on essaie de déterminer si deux graphes donnés sont égaux. Formellement, on définit les notions de l'homomorphisme et de l'isomorphisme.

Définition 3.1.1 (Homomorphisme de graphes): Soient $G = (N, E)$ et $G' = (N',\ E')$ deux graphes.

S'il existe une relation univoque m: $N \longrightarrow N'$ telle que $(v_1,\ v_2) \in E \Longrightarrow \big(m(v_1)\,, m(v_2)\big) \in\ E'$, alors G et G' sont homomorphes.

Intuitivement, et comme le montre la figure 2-1, on cherche à savoir si le graphe G' est isomorphe aux sous-graphes constitués des sommets $\{a, b, c\}$ comme à celui formé par les sommets $\{b, c, d\}$.

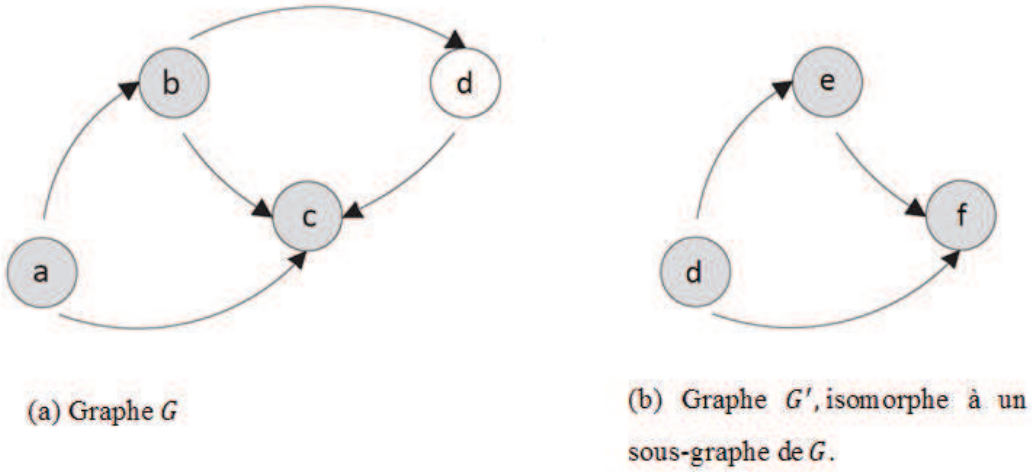


Figure 2- 1: Exemple de graphe et de sous-graphe

Définition 3.1.2: Soit deux graphes $G = (N, E)$ *et* $G' = (N', E')$, et soit les sous-ensembles de sommets $N_s \subseteq N$ et $N'_s \subseteq N'$, tels que $|N| = |N'_s|$. Un monomorphisme de N_s à N'_s est une bijection $f: N_s \to N'_s$ telle que

$$\forall\, u, v\ \in N_s, (u, v) \in E \Rightarrow \big(f(u), f(v)\big) \in E'. \qquad \textbf{(1)}$$

Ainsi, pour chaque paire de sommets de N_s reliée par une arête, la paire de sommets correspondante dans N'_s doit être reliée par une arête. Un isomorphisme

entre N_s et N'_s est une extension du monomorphisme, où on impose la contrainte supplémentaire que si une paire de sommets de N_s n'est pas reliée par une arête et la paire correspondante dans N'_s ne l'est pas non plus, c.-à-d.

$$\forall\, u, v \in N_s, (u, v) \in E \Leftrightarrow \big(f(u), f(v)\big) \in E'. \qquad \textbf{(2)}$$

Les problèmes d'isomorphisme de graphe et de sous-graphe peuvent s'exprimer à l'aide de la matrice d'adjacence. On définit ainsi la matrice d'adjacence M d'un graphe G suivant les arcs qu'il contient en ordonnant ses sommets V. On commence par ordonner les sommets $V = \{v_1, v_2, \dots, v_n\}$ du graphe, puis s'il y a un arc du sommet v_i vers le sommet v_j alors $M_{i,j} = 1$, sinon $M_{i,j} = 0$.

Définition 3.1.3: Soit $G = (V, E)$ un graphe à $n \in \mathbb{N}$ sommets. La matrice d'adjacence M associée à G est la matrice $n * n$ définie par:

$$M_{i,j} = \begin{cases} 1, si\ (i,j) \in E \\ 0, sinon \end{cases} \qquad \textbf{(3)}$$

Le problème de l'isomorphisme peut alors se reformuler. On dit que deux graphes G et G' sont équivalent, noté $G = G'$ si leur matrices d'adjacence A et A' sont identique, c.-à-d. $A = A'$. Par ailleurs, un automorphisme f peut être considéré comme une permutation de l'ordre des sommets dans V produisant une matrice d'adjacence identique. En notant A^f la matrice d'adjacence permutée par f, on a $A^f = A$. Selon le même raisonnement, G et G' sont isomorphes s'il existe une permutation f des sommets de V telle que $A^f = A'$. Si une telle permutation existe, il existe également une permutation f^{-1} telle que $A = (A')^{f^{-1}}$.

De même soit P une matrice $|V| \times |V'|$ tel que:

$$P_{i,j} = \begin{cases} 1, si\ f(v_i) = v'_j \\ 0, sinon \end{cases} \quad \textbf{(4)}$$

Une telle matrice est dite de permutation. Le problème d'isomorphisme de graphe et de sous-graphe peuvent être formulés de la façon suivante: trouver la matrice de permutation P telle que $A = PA'P^T$.

2.3.2. Isomorphisme de sous-graphes

Le problème de l'isomorphisme de sous-graphes consiste à trouver une application partielle entre les sommets de deux graphes, tel que les conditions définies dans la définition 3.2.1 (plus bas) soient satisfaites. Soient G et G' deux graphes avec $n' \leq$ n, on cherche un sous-graphe de G induit par n' sommets. Si l'on considérait des sous-graphes en général au lieu de sous-graphes induits par des sommets, on ne traiterait pas le problème d'isomorphisme, mais celui de l'homomorphisme.

Nous considérons ici des ARG, c'est-à-dire que des valeurs sont associées aux nœuds ainsi qu'aux arêtes. Le problème d'identification d'une correspondance entre deux ARG ainsi que la question de leur isomorphisme sont des tâches centrales de la reconnaissance de formes structurelles. Les ARG constituent une description abstraite des objets réels ayant une structure inhérente. En général, les composants de l'objet sont modélisés ainsi que leurs relations. Les définitions de l'isomorphisme peuvent être facilement étendues à des graphes étiquetés. Dans ce but, il suffit d'assurer que les attributs des nœuds et des arêtes soient respectés.

Définition 3.2.1: Soient $G = (N, E, \alpha, \beta)$ et $G' = (N', E', \alpha', \beta')$ deux graphes tiquetés. Si la relation $m: N \longrightarrow N'$décrit un isomorphisme entre (N, E) et (N', E'), et

$\forall\, n \in N\ \alpha(n) = \alpha'(v)$ ainsi que $\forall\ (n_1, n_2) \in E, \beta(n_1, n_2) = \beta'(m(n_1), m(n_2))$ alors G est isomorphe à G'.

3. LE WAFERBOARD™

Le WaferBoard™ est un système de prototypage rapide pour les circuits numériques capable d'interconnecter rapidement un ensemble de composants électroniques. Le système est conçu pour permettre de tester et valider rapidement des circuits au niveau système. Il a été élaboré pour satisfaire aux besoins croissants de l'industrie et de la recherche dans le domaine de l'électronique. Il est d'une certaine façon un PCB reprogrammable contenant une densité d'interconnexion supérieure aux PCB multicouches conventionnels. Dans cette carte, les composants électroniques sont simplement déposés sur la surface active de la WaferBoard™ qui est une tranche de silicium nommée WaferIC™. Le WaferIC™ est un circuit électronique intégré au niveau de la tranche de silicium. Il permet la transmission de données entre différents réticules d'une même tranche.

3.1. Fonctionnement de la carte WaferBoard™

Avec le WaferBoard™, les composants électroniques sont simplement déposés manuellement sur le WaferIC™ qui a la capacité de détecter les contacts entre elle-même et ces composants. Cela signifie que le WaferIC™ permet un positionnement indifférent à l'alignement au contraire des PCB conventionnels dont les circuits discrets doivent être soudés en respectant soigneusement l'alignement des plots. Le WaferIC™ transmet les informations sur le positionnement des circuits intégrés, permettant ainsi au système d'afficher le résultat obtenu sur un écran d'ordinateur. La netlist détaillant l'interconnexion des pins du système à concevoir est envoyée au WaferIC™, qui les interconnecte pour permettre la transmission des signaux entre les différentes boules. Ces connexions sont programmées à l'aide d'un port JTAG (Joint Test Action Group) d'une manière similaire aux FPGA (Field Programmable Gate Array) [8,10].

La tranche de silicium est au cœur du WaferBoard™ et contient des millions de plots configurables groupés sous forme cellulaire et ayant une dimension de $77\mu m \times 77\mu m$ [11], appelés NanoPads™. Les NanoPads™ sont intégrés à l'échelle de la tranche de silicium afin de garantir une densité d'interconnexion élevée [8]. Ces plots peuvent être configurés de manière à agir comme un plot d'E/S numériques, comme une source de tension régulée, comme masse, ou agir comme un plot flottant (haute impédance). Après avoir détecté l'emplacement de chaque pin de ces composants, la tranche de silicium permet de mettre en place des raccordements de puissance ainsi que des raccordements de signal entre les composants CI (Circuit Intégré). Afin de maintenir l'intégrité du signal, ces mêmes raccordements sont jumelés à des répéteurs [10]. Chaque NanoPad™ est reliée au WaferNet™ «un réseau interne Wafer-Scale d'interconnexion». Le WaferNet™ peut être configuré pour relier un NanoPad™ à un ou plusieurs autres NanoPad™ sans générer de conflit de connexion ou de raccordement. Les circuits et les composants CI peuvent être disposé sur la surface du wafer (Figure 3-1) de façon aléatoire aussi les NanoPads™ sont assez dense pour que chaque bille du composant soit en contact avec plusieurs NanoPads™ [8,10]. Ainsi le rôle des NanoPads™ est la détection des pins des composants et la liaison de ces pins au réseau d'interconnexion. Ils représentent les points d'accès au niveau de l'alimentation ou bien pour la transmission des signaux numériques aux composants.

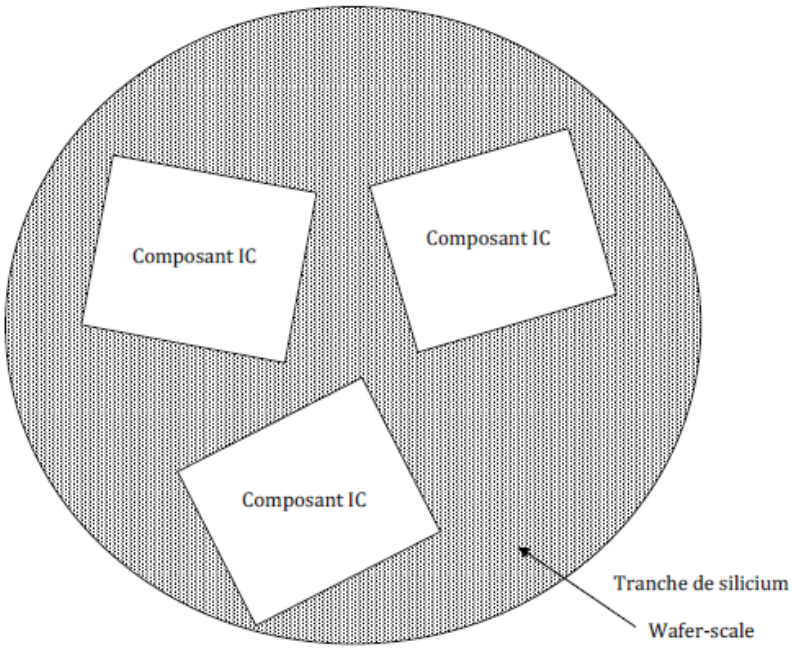


Figure 3- 1: Composants CI posés sur la surface de la tranche de silicium

La notion de la tranche de silicium présente deux attributs majeurs:

- Une haute densité de surface de contact. Des alignements ayant une densité élevée de pads conductrice et ayant une forme carrée et d'une dimension de $200\mu m$ sont contenus dans la surface de contact située dans le haut de la plaquette wafer-scale [10]. Ces pads sont indispensables pour manipuler, supporter et détecter la totalité des pins des composants CI existante dans le marché des composants électroniques packagés de type « BGA, Ball Grid Array » [8].
- Une connectivité interne élevée. Elle est nécessaire pour établir les branchements entre la surface et les boules des composants CI en créant le substrat reconfigurable pour le système de prototypage rapide [12]. Cette

haute connectivité augmentera le nombre des points de contact entre la surface et les boules ainsi que la manipulabilité de ces derniers.

L'utilisateur du WaferBoard™ applique une pression sur le composant [13] pour permettre un bon contact des boules des composants avec la surface de la tranche de silicium (Figure 3-2). Pour afficher l'état de la carte de prototypage et celui des pins des composants et aussi programmer le réseau d'interconnexion à partir de la netlist, une plateforme logicielle est mise à sa disposition depuis un ordinateur.

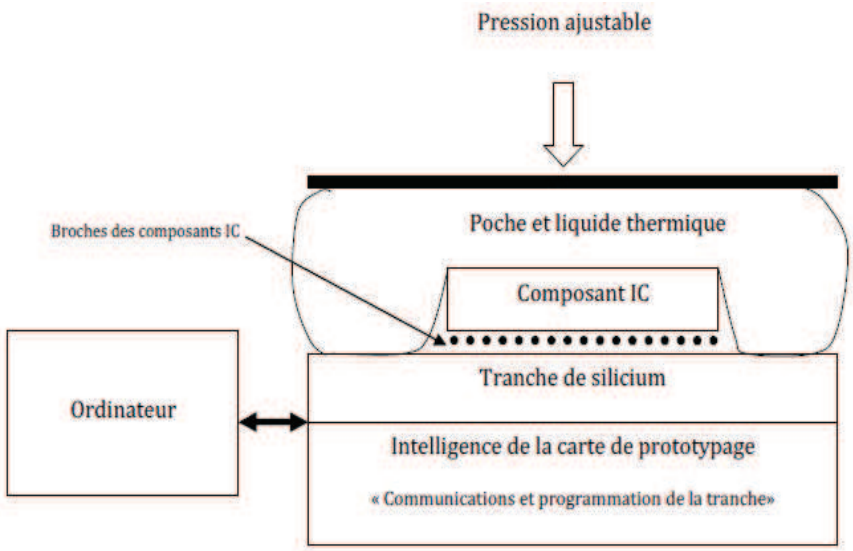
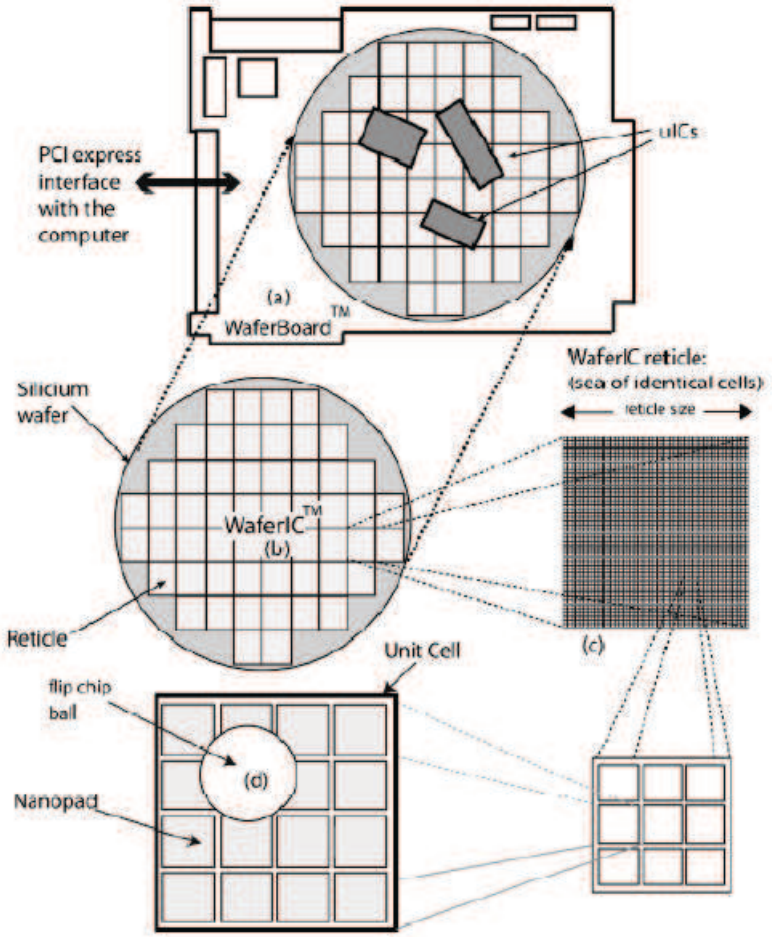


Figure 3- 2: Illustration de la carte de prototypage connecté à un ordinateur

Une architecture cellulaire réduite d'une maniéré considérable la conception du WaferIC™. Le réseau interne est géré par une puce intelligente contenue au niveau de l'unité- cellule. L'unité-cellule renferme aussi un dispositif de régulation de la tension et des NanoPad™ sur la surface. Ainsi, ces NanoPads™ sont contenus dans les unités-cellules qui sont elles-mêmes contenues dans un réticule. La photorépétition de l'unité-cellule tout au long du wafer est la méthode de construction du wafer-scale. Afin d'assurer les liaisons entre les réticules, des techniques d'interréticule ont été

utilisés durant la photorépétition [8,10]. Par conséquent, le wafer-scale est composé d'un ensemble de cellules identiques reliées entre eux par un réseau d'interconnections reconfigurable couplé et relié à une plage de rangés très denses de NanoPads™ (Figure 3-3).



(a) : carte de prototypage, (b) : tranche de silicium « wafer-scale », (c) : réticule qui regroupe un ensemble d'unités-cellules, (d) : bille du composant IC.

Figure 3- 3: Illustration de la carte de prototypage, le wafer, réticule, cellule [40]

La figure 3-4 montre l'étage master qui est au centre de l'unité-cellule. L'étage maître a le rôle de contrôle de l'unité-cellule. Cette figure représente aussi un agrandissement d'un réticule qui contient quatre unités-cellules et qu'elle-même contient 4 × 4 NanoPad™. La surface d'une unité-cellule est de l'ordre millimétrique alors que la surface du réticule est de l'ordre centimétrique [8].

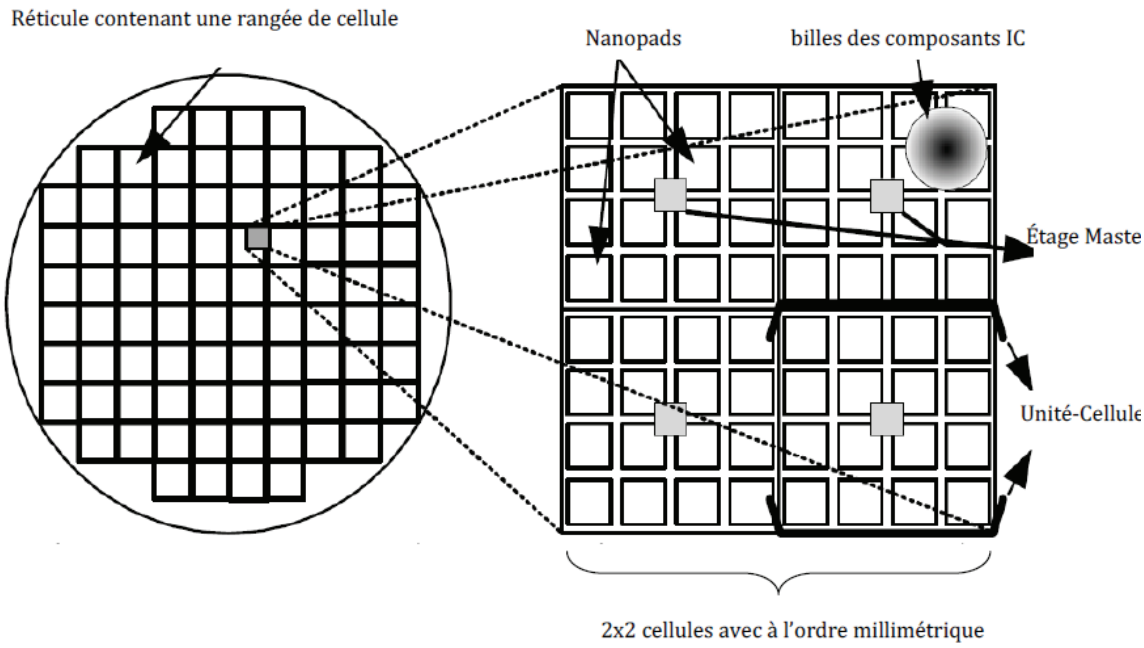


Figure 3- 4: Wafer-scale avec des rangées de cellules programmables contenant 4x4 NanoPads™ [40]

Un NanoPad™ peut être relié à n'importe quel autre ensemble NanoPads™ à l'aide du WaferNet™. Chaque unité-cellule est reliée à sept autres unités-cellules dans chaque direction (Nord-Sud-Est-West) en utilisant une architecture régulière cellulaire de maille [10]. Dans chaque direction, chaque lien a une longueur différente, catégorisée comme suit: 1, 2, 4, 8, 16, 32 et 64. Par exemple, un lien de longueur 2 relie la cellule à la deuxième cellule voisine, suivant les indications du schéma présenté par Norman et coll. [10] (Figure 3-5).

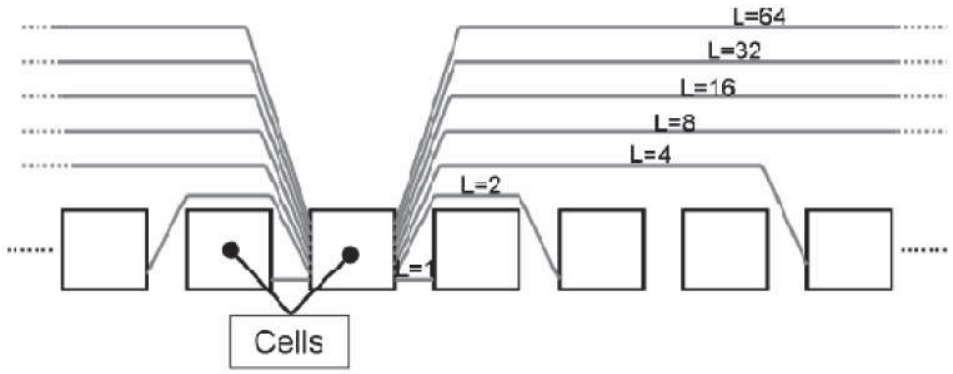


Figure 3- 5: Connectivité dans le WaferNet™ [40]

N'importe quel lien entrant est relié à n'importe quel lien sortant par une grande barre transversale se trouvant dans chaque cellule de la tranche de silicium. Le fonctionnement de la barre transversale est similaire à celui d'un concentrateur « HUB » avec plusieurs ports d'entrées-sorties. Les NanoPads™ sont aussi connectés à cette barre transversale (Figure 3-6).

Les signaux numériques sont efficacement propagés sur les lignes de transmission grâce aux répéteurs inclus dans chaque unité-cellule. La possibilité de connecter une unité-cellule avec une autre unité-cellule plus loin est grandement augmentée.

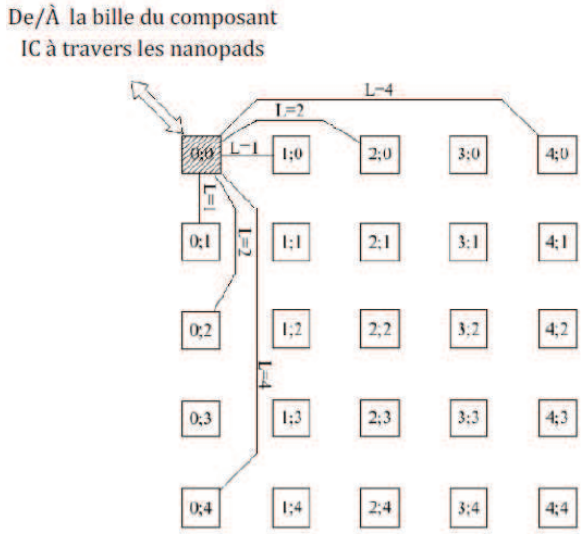


Figure 3- 6: Présentation du WaferNet™ connectant des billes du composant CI [40]

L'une des caractéristiques fondamentales du WaferBoard™ est la détection et la localisation d'anomalies. Le réseau tolérant aux défauts contrôle ces anomalies qui ont la redondance de niveau logique et de cellule.

Chaque cellule intègre des circuits permettant de détecter le contact électrique entre les pins de l'IC et les NanoPads™. Norman et coll. [8] (Figure 3-7) ont proposé une solution pour détecter de tels contacts. Il s'agit ici d'identifier les courts-circuits crées par la bille du composant CI entre un ou plusieurs NanoPads™ ainsi que les minuscules ouvertures dans le NanoPads™. Une autre solution a été proposée par Blaquière et coll. [14] qui consiste à mesurer la capacité de charge e chargement capacitif sur chaque NanoPad™.

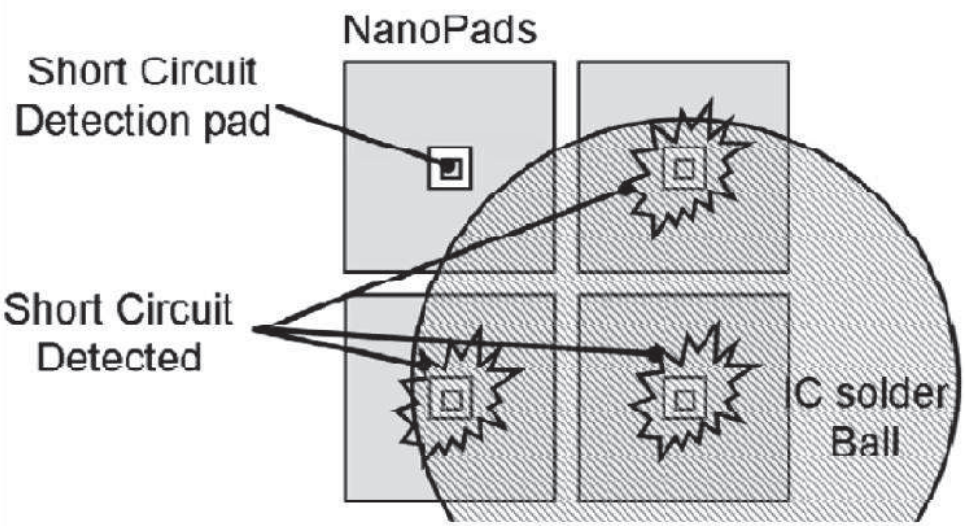


Figure 3- 7: Détection des billes du composant par le court-circuit [40]

La détection des composants dure quelque microseconde [8]. Après cette détection, l'emplacement des billes détectées est transmis par le système afin de créer une cartographie des composants CI. En se basant sur ce plan et la netlist fournie par l'utilisateur, le réseau d'interconnexion est configuré et les raccordements électriques entre les billes des différents composants CI sont créés.

3.2. Description du Workflow

Afin de soutenir le prototypage rapide du système basé sur la carte de circuit configurable, il est essentiel d'avoir un ensemble d'outils et un flux de travail (workflow) [15]. Il s'agit notamment de diagnostiquer les défauts qui peuvent être présents dans le WaferIC™ pour prévenir leurs utilisations. Le système doit également détecter et reconnaître les pins des composants CI placées sur sa surface. Il doit aussi traiter la netlist de l'utilisateur qui comprend une centaine de milliers de nets et les fichiers utilisateurs de définition de contrainte à fin de construire un flux de bits comprenant des mégabits de donnés pour configurer plusieurs millions de

segments d'interconnexion dans le circuit configurable. Ces tâches ne peuvent pas se faire sans l'appui d'outils appropriés et un workflow bien défini.

Le flux de travail inclut huit étapes comme la montre la Figure 3-8. Le processus de démarrage et de diagnostic (1) met automatiquement sous tension le wafer et fait un scan complet du wafer pour extraire le schéma de connexion des NanoPads™. Un NanoPad™ est connecté lorsqu'il est en contact avec l'une des billes d'un des composants placés à la surface du WaferIC™. Une carte de contact est extraite sur la base de court-circuit créé par une bille d'un composant CI et les NanoPads™ adjacents. La carte de contact est alors utilisée par le processus de reconnaissance de package (3). Cet outil est basé sur les algorithmes de filtrages de motifs. Le processus de routage (5) est alors exécuté selon la netlist et les contraintes fournies par l'utilisateur (4). L'utilisateur définit les connexions qui devront être faites entre les ICs déposés sur la surface, qui peuvent être manuellement définies ou lues à partir d'un fichier de netlist standard (p. ex. EDIF (Electronic Design Interchange Format), GRB (gerber), Protel). Le processus de routage consiste à lire les contraintes de performance et d'implantations et puis à calculer une liste d'itinéraires pour chaque net dans la netlist. Ces contraintes ont certaines similarités avec celles utilisées dans le routage des PCB. Le routage présente d'autres contraintes par exemple le temps de latence du bus (contraintes de temps) et la bande passante. Finalement, l'outil génère un flux de bits pour configurer le WaferIC™ et les rapports de routage.

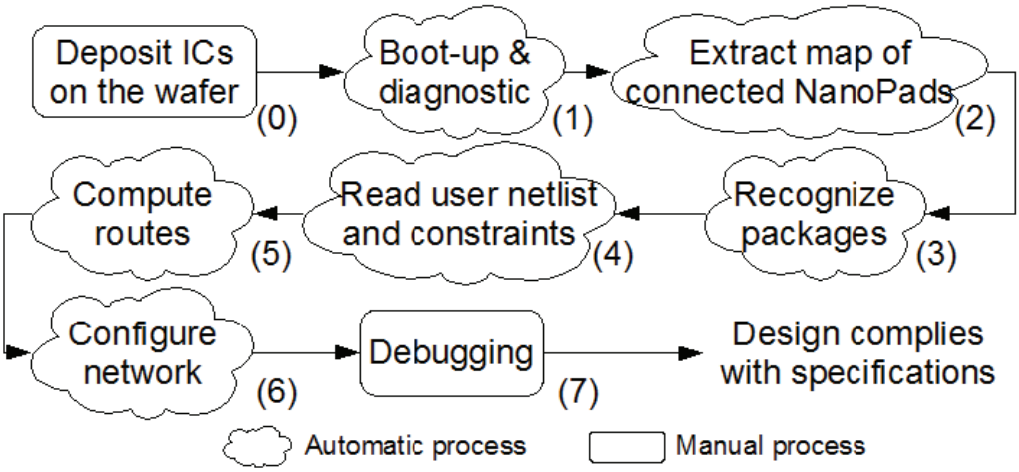


Figure 3- 8: Flux de travail proposé pour le système de prototypage [15]

3.3. Exigences et spécifications du système WaferBoard™

- L'Outil de Filtrage de Netlist (OFN) reçoit la netlist en format EDIF comme entrée.
- L'OFN doit être capable d'analyser la netlist est de la sauvegarder dans une structure de donnée.
- L'OFN doit contenir des méthodes qui lui permettent de lire à travers la structure netlist construite pour extraire l'information dont il a besoin.
- L'OFN doit être capable de reconnaître n'importe quels sous-circuit motif et doit être en mesure de remplacer cette combinaison ou de la supprimer si possible.
- L'OFN doit offrir la possibilité de retirer ou de remplacer n'importe quel composant, ou l'instanciation du composant de la netlist selon des

règles prédéfinies et ainsi maintenir la structure hiérarchique de la netlist.

- Après la fin de l’opération d’élagage, une structure netlist élaguée devrait être construite pour être utilisé par la suite par d’autre module.
- La netlist générée devrait conserver une structure de conception hiérarchique pour d'autres utilisations.

4. ÉTAT DE L'ART

L'extraction de sous-circuit est une application très importante dans de nombreux domaines de la conception assistée par ordinateur, en particulier pour la conception VLSI (Very Large Scale Intergration) avancée. Par exemple, du fait que la conception VLSI se généralise au fil des années, les industrielles du semi-conducteur doivent veiller à ce qu'il n'y ait pas de bugs et que les puces manufacturées sont sans défaut. L'extraction de sous-circuits est l'une des opérations dans le test layout vs Schematic (LVS), qui est un moyen utile pour réduire au minimum les erreurs de masque [6,7]. Par exemple, trouver un sous-circuit dans un circuit principal est une tâche fréquente pour modifier ou supprimer le sous-circuit, qui peut exister partout dans le circuit principal. Ce processus peut être efficacement géré lorsqu'il est appliqué sur des petites netlist, mais devient fastidieux sur les très grandes si elles sont réalisées manuellement. Le problème d'extraction de sous-circuits peut être définie comme suit: étant donné un sous-circuit SC et un circuit C principal bien plus grand que SC. Le problème consiste à détecter si SC existe en C et à modifier SC en C selon des règles d'adaptation. Dans la partie qui va suive nous allons faire une brève revu des publications traitant des principales techniques développées pour résoudre ce genre de problème.

4.1. L'extraction de sous-circuits

Des algorithmes spécialisés ont été conçus pour réaliser l'opération d'extraction des sous-circuits depuis 1983. Les premiers algorithmes s'appuient sur les caractéristiques spécifiques de la technologie des circuits traités. Ils n'ont pas été facilement appliqués à d'autres technologies ou à d'autres circuits, tels que les circuits analogiques. En outre, ces techniques sont fondées sur des hypothèses

concernant les sous-circuits qui sont en train d'être extraits, et ne peuvent pas être généralisé pour permettre l'extraction de sous-circuits arbitraires

A partir, de 1994, plusieurs variétés d'algorithmes avancés ont été inventées sur la base de la théorie des graphes. En traitant l'extraction de sous-circuit comme un problème d'isomorphismes de sous-graphes qui ne font aucune supposition sur les types des circuits en question, des solutions technologiquement indépendantes ont été déterminées. Technologiquement indépendante signifie que le même algorithme peut être utilisé dans de nombreux contextes différents.

Miles Ohlrich et coll. [25] furent les premiers à résoudre le problème d'extraction de sous-circuit basé sur une solution du problème d'isomorphisme de sous-graphe. Leur algorithme a été implémenté dans un logiciel commercial appelé $SubGemini$. En raison de l'exhaustivité de ses résultats expérimentaux et de son temps d'exécution rapide, $SubGemini$ est devenu un algorithme fréquemment utilisé. $SubGemini$ Offre une représentation des circuits en des graphes bipartis: les composants avec des carreaux formant un premier ensemble de sommets, et les nets avec des cercles formant un deuxième ensemble. La connexion entre deux composants est faite pour connecter chaque sommet des composants à un sommet du net. $SubGemini$ Utilise un algorithme de partitionnement basé sur celui développé par Corneil [41], pour trouver un mappage isomorphe entre deux circuits. Le partitionnement est fait de telle sorte que deux sommets équivalents seront dans le même partitionnement. Initialement les invariants des sommets comme les types de composants et le degré des nets sont utilisés pour le partitionnement des sommets. Ce partitionnement est réalisé implicitement via un algorithme d'étiquetage. Les sommets sont étiquetés par l'invariant des sommets et les étiquettes sont utilisées pour classifier les sommets en partition. L'algorithme $SubGemini$ est constitué de deux phases. Dans la phase1, $SubGemini$ identifie tous les emplacements possibles du sous-circuit dans le circuit principal. Il accomplit cette tâche en appliquant un algorithme de partitionnement à la fois au sous-circuit et au circuit principal afin de

choisir une clé de sommets K dans le sous-circuit. Il identifie également tous les sommets possibles dans le circuit principal qui pourraient correspondre à la clé de sommets. Cet ensemble de sommets est appelé le vecteur candidat C_V. La phase 1 agit comme un filtre qui tente de réduire le nombre d'occurrences ayant besoin d'être vérifiées. Dans la phase 2, chaque instance est vérifiée pour déterminer si elle fait partie d'un sous-circuit. Le résultat expérimental fait sur SubGemini montre que la durée typique pour les circuits CMOS (Complementary métal-oxyde-semiconductor) de grande taille est à peu près linaire, par rapport au nombre total de composants à être identifiés des sous-circuits. Cependant, cet algorithme de ré-étiquetage repose sur l'hypothèse que les nets externes ne sont pas court-circuités à d'autres nets externes du même sous-circuit. Cet algorithme a mis 174.3 secondes pour identifier 6664 motifs de sous-circuit dans un circuit comprenant 100 000 transistors.

Huang et coll. [42] a résolu ce problème en utilisant une nouvelle méthode de codage pour les graphes représentant des circuits. Une matrice est générée à partir du graphe de chaque sous-circuit en fonction du fichier de la technologie, puis chaque matrice est partitionnée. Cette procédure de partitionnement donne un ordre unique des nœuds du sous-circuit. Ainsi, un code unique est généré pour chaque circuit. Cet algorithme a été mis en œuvre sur un circuit avec un maximum de 15 000 transistors en 28 secondes.

Vijaykrishnan et coll. [43] en 1996 a présenté une approche appelée $SubGen$ pour modéliser le problème d'extraction de sous-circuit à l'aide de l'algorithme génétique. Il fait usage de la fonction de remise en forme et les opérateurs génétiques de croisement et de mutation pour trouver l'ensemble de candidats avec la remise en forme maximale. $SubGen$ peut identifier différents types de sous-circuits. Mais, il ne peut pas garantir de trouver tous les sous-circuits est c'est parce qu'un algorithme génétique est en lui-même un algorithme d'approximation. Ainsi, il n'y a aucune garantie de trouver la solution optimale.

L'algorithme $DECIDE$ créé par Chang et coll. en 2001 [44] adopte un régime récursif pour réaliser l'opération d'identification. Une fonction attribue un poids à chaque nœud sur la base de ses voisins. Un nœud ayant un type classique (exemple N-type, P-type, un composant, ou un terminal), et ayant des voisins ayant un type classique aussi, présente une valeur de pondération unique. Ainsi, la valeur de pondération est utilisée pour réduire le nombre d'instances qui ont besoin d'être vérifiées. Cet algorithme a été mis en œuvre sur un circuit contenant jusqu'à 100.000 transistors en pp seconds pour identifier 6664 motifs.

En 2002, Wunsh et Zhang [45] ont proposé un algorithme d'extraction de sous-circuits basé sur les réseaux de neurones utilisant une fonction de programmation heuristique dynamique. Cette approche est la première à utiliser les réseaux neuronaux comme approche pour résoudre le problème d'extraction de sous-circuit. Cet algorithme a été testé sur un circuit ayant 100.000 transistors en 85.442 secondes. Le duo a également proposé une approche utilisant les graphes flous et attribués pour résoudre le même problème [46]. Cette approche a été implémentée sur un circuit de 100.00 transistors et elle a mis 18.692 secondes pour extraire tous les sous-circuits.

Zong Ling [23] a proposé une autre méthode de résolution pour le problème d'extraction de sous-circuit. Cette méthode se base sur le concept de la gestion des ressources. Cette approche est intégrée avec un ensemble de politiques et de concept comprenant la criticité pour l'identification du processus et la sélection de la solution, et le concept de l'arbre centré maximal pour la décomposition et le partitionnement du circuit. L'algorithme présenté par Zong est divisé en deux phases: une phase de prétraitement, où le circuit motif est décomposé et le circuit principal est partitionné, et une phase d'appariement guidé par les principes de la planification des ressources limités.

L'approche proposée par cet auteur est efficace pour l'extraction de sous-circuits en adaptant trois stratégies: le prétraitement du circuit motif, la transformation du

problème d'extraction de sous-circuits en un problème de gestion de ressource et en utilisant des arbres centrés maximal pour réduire le nombre de backtrack. Cela permet d'éliminer les sous circuit non isomorphes plus tôt dans la fouille en minimisant ainsi de façon significative le temps perdu dans l'appariement infructueux.

La nouveauté dans cette approche est l'utilisation du principe de la gestion de ressource par contrainte ce qui optimise le temps d'exécution pour obtenir un temps presque linéaire. Cependant, construire le CRP requiert beaucoup de temps additionnel et consomme beaucoup de temps pour des circuits plus larges ou complexe**.** La CRP est une méthodologie heuristique indépendante du domaine. Elle est basée sur six concepts: ressource, tâche, contrainte, solution, criticité et critique. Elle traite un problème de planification composé par plusieurs tâches à réaliser, sous la condition que les ressources sont limitées.

Plusieurs chercheurs ont abordé le problème d'extraction de sous-circuits depuis un plus large. Le point commun entre les algorithmes présentés est la transformation du problème d'extraction de sous-circuit en un problème d'isomorphisme de sous-graphe. Dans cette partie, nous allons présenter une recension des méthodes les plus connues pour l'extraction de sous-circuits et celles pour la résolution du problème d'appariement des graphes.

4.2. L'appariement de graphes

À partir des années soixante-dix, les techniques basées sur la représentation des objets en graphe ont été proposées comme un outil puissant pour la représentation et la classification des patrons (motifs) en structure dans la reconnaissance des formes structurelles « Pattern Matching (PR) ». Après l'enthousiasme provoqué par l'apparente « intelligence » de cette structure de données, les graphes n'ont pratiquement plus été utilisés pendant une longue période de temps. Récemment, l'utilisation des graphes dans les PR est en train de recevoir une attention croissante

de la communauté scientifique. C'est peut-être dû au fait que le coût de calcul des algorithmes à base de graphe, bien que toujours élevé dans la plupart des cas, est en train de devenir compatible avec la puissance de calcul des nouvelles générations d'ordinateurs.

Ici, nous essayons de faire une compilation de la littérature sur les techniques et les algorithmes les plus connus sur l'appariement des graphes. En outre, nous dresserons d'abord quelques définitions et notations importantes relatives aux graphes.

4.2.1. Isomorphisme de graphe et de sous-graphe

Dans de nombreuses applications, une opération cruciale est la comparaison entre deux objets ou entre un objet et un modèle dans lequel l'objet pourrait être lié. Lorsque ces informations structurées sont représentées par des graphes, cette comparaison est effectuée en utilisant une certaine forme d'appariement de graphes. L'appariement de graphe est le processus de trouver une correspondance entre les nœuds et les arêtes de deux graphes qui satisfait certaines contraintes (plus ou moins strictes) veillant à ce que les sous-structures similaires dans un même graphe soient appariées à des sous-structures similaires dans l'autre graphe.

Dans cette section, nous présentons une revue des principaux algorithmes qui ont été proposés et utilisés dans le domaine du PR pour l'appariement de graphes, et les problèmes connexes, telle que la reconnaissance de sous-circuits dans un plus large. La Figure 4-1 présente un tableau des travaux qui seront examinés, organisés selon le type de problème abordé par chaque algorithme et la technique de solution.

En s'inspirant de [47], on a devisé les méthodes d'appariement en deux grandes catégories: la première contient les méthodes d'appariement exactes qui exigent une stricte correspondance entre les deux objets étant appariés ou au moins parmi leurs sous partis. La deuxième catégorie renferme les méthodes d'appariement

inexact, où une correspondance peut se produire même si les deux graphes comparés sont structurellement différents dans une certaine mesure.

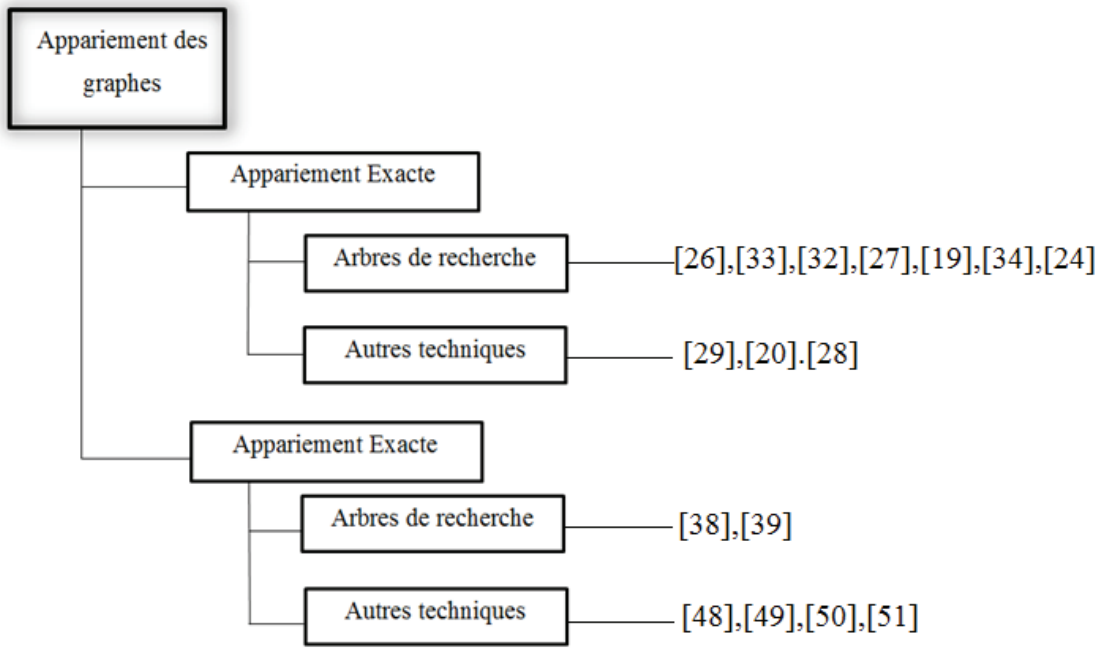


Figure 4 - 1: La taxonomie des algorithmes recensés. Pour chaque algorithme la taxonomie indique la référence bibliographique correspondante.

4.2.1.1. Appariement exact

L'appariement exact des graphes est caractérisé par le fait que l'appariement entre les nœuds de deux graphes doit préserver les arêtes dans le sens que si les deux nœuds dans le premier graphe sont reliés par une arête, ils sont appariés à deux nœuds dans le second graphe relié aussi par une arête. Dans la forme la plus rigoureuse de l'appariement exact des graphes, l'isomorphisme de graphe, cette condition doit tenir dans les deux sens, et le l'appariement doit être bijectif. C'est une correspondance terme à terme qui doit être trouvé entre chaque nœud du premier graphe et chaque nœud du deuxième graphe. Une forme plus faible de l'appariement est l'isomorphisme de sous-graphe, qui exige qu'un isomorphisme doive exister entre

l'un des deux graphes et un sous-graphe de l'autre. En effet, certains auteurs [31] utilisent le terme d'isomorphisme de sous-graphe dans un sens plus faible, passant la condition que l'appariement doit préserver les arêtes dans les deux sens.

Les problèmes d'appariement mentionnés ci-dessus sont tous NP-complet, sauf pour l'isomorphisme de graphe, pour lesquels il n'a pas encore été démontré s'il appartient à cette catégorie ou non. Les algorithmes d'isomorphisme polynomiaux ont été développés pour des types particuliers de graphes (par exemple pour les arbres [35], pour les graphes planaires [36], pour les graphes de valence bornée [37]), mais aucun des algorithmes polynomiaux n'est connu pour le cas général. Par conséquent, l'appariement exact des graphes a une complexité temporelle exponentielle dans le pire des cas. Toutefois, dans les applications de PR le temps de calcul peut être acceptable en raison de deux facteurs: d'abord, les types de graphes rencontrés en pratique sont généralement différents des pires cas pour les algorithmes. Deuxièmement, l'utilisation des nœuds et des arêtes étiquetées peut réduire considérablement le temps de recherche.

Parmi les problèmes mentionnés ci-dessus, l'isomorphisme exact est très rarement utilisé dans les PR, puisque le plus souvent les graphes comparés sont obtenus à la suite d'un processus de description (parfois très complexes) qui est inévitablement soumise à une forme de bruit, et ainsi que des nœuds ou et des arêtes de moins ou de plus peuvent apparaître, ce qui entrave l'isomorphisme. Contrairement, l'isomorphisme de sous-graphes et le monomorphisme, malgré qu'ils soient plus gourmant au niveau temps de computation, peuvent être utilisés dans de nombreux contextes, et plusieurs algorithmes pour ces problèmes ont été proposés.

a) Les techniques basées sur les arbres de recherche.

La plupart des algorithmes d'appariement exact de graphes sont basés sur une certaine forme d'arbre de recherche avec retour sur trace (backtracking). L'idée de base est qu'une correspondance partielle (initialement vide) est itérative et élargie en

y ajoutant des nouvelles paires de nœuds appariés ; la paire est choisie à l'aide de conditions nécessaires qui assurent sa compatibilité avec les contraintes imposées par le type d'appariement et les nœuds appariés jusqu'à présent, et généralement en utilisant également une condition heuristique pour élaguer dès que possible les chemins de recherche infructueuse. Éventuellement, l'algorithme soit qu'il trouve une correspondance complète, soit qu'il atteint un point où l'appariement partiel actuel ne peut être élargi en raison des contraintes d'appariement. Dans ce dernier cas, l'algorithme fait un retour en arrière, c'est-à-dire annule les derniers ajouts jusqu'à ce qu'il trouve une correspondance partielle, pour laquelle une extension alternative est possible. Si toutes les correspondances possibles qui satisfont les contraintes ont déjà été essayées, l'algorithme s'arrête. Différentes stratégies d'implémentation de ce type d'algorithme ont été employées, ayant comme point de différenciation l'ordre dans lequel les correspondances partielles sont visitées. Probablement le plus simple est le parcours en profondeur (depth-first search) qui nécessite moins de mémoire que les autres. Une propriété essentielle de ces algorithmes est qu'ils peuvent être très facilement adaptés pour tenir compte des attributs des nœuds et des arêtes sans aucune limitation sur le type d'attributs qui peuvent être utilisés. Ceci est très important pour les applications de PR, où souvent les attributs jouent un rôle clé dans la réduction du temps de calcul de l'appariement.

L'algorithme Ullmann [26] est le plus populaire pour résoudre le problème de l'isomorphisme de sous-graphe. Malgré qu'il date de 1976, il est encore largement utilisé. Les problèmes abordés par l'algorithme sont l'isomorphisme de graphes, l'isomorphisme de sous-graphes et le monomorphisme. Pour tailler les appariements infructueux, Ullmann propose une procédure dite de raffinement, qui travaille sur une matrice de nœuds possible à appariés dans le futur pour enlever, en se basant sur une condition nécessaire et bien définie, ceux qui ne sont pas compatibles avec le courant appariement partiel.

Un algorithme plus récent à la fois pour l'isomorphisme de graphe et de sous-graphe est l'algorithme VF [33, 32]. Cet algorithme résout avec une approche directe ces problèmes, en utilisant une heuristique qui est basée sur l'analyse des ensembles des nœuds adjacents à ceux déjà pris en compte dans l'appariement partiel. Cette heuristique est rapide à calculer amenant dans de nombreux cas à une amélioration significative à l'algorithme d'Ullmann et bien d'autres [27]. L'algorithme VF reprend à la base les grandes lignes de celui d'Ullmann. Ils ont en commun l'utilisation du backtracking en combinaison de mécanisme de forward checking. Pour résoudre le problème d'isomorphisme de sous-graphe, l'algorithme VF utilise la représentation d'état (State Space Representation SSR), selon laquelle un état s représente une correspondance partielle. Ce processus d'appariement consiste en la détermination d'une fonction d'appariement M qui associe les nœuds du graphe G aux nœuds du graphe G'et vice versa. $M(s)$ est un sous-ensemble de M et la transition entre deux états $s_1$et s_2 correspond à l'ajout d'une nouvelle paire de nœuds appariés (Figure 4-2). Pour chaque état intermédiaire s, l'algorithme considère l'ensemble $P(s)$ des paires de nœuds qui peuvent être ajoutées à l'état actuel. Un ensemble de règles de faisabilité est introduit sur l'algorithme VF afin de vérifier les conditions cohérence, rendant possible seulement la génération d'états cohérents.

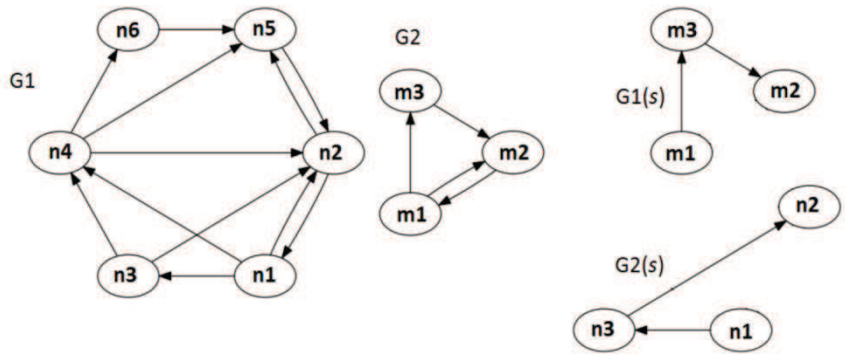


Figure 4 - 2: Deux graphes et leurs graphes partiels relatifs à la fonction d'appariement M(S)={(n1,m1)(n2,m2)(n3,m3)}

Une description de haut niveau de l'algorithme VF est décrite dans la Figure 4-3. À l'état initial s_0, la fonction d'appariement M ne contient aucun élément. Pour chaque état intermédiaire s, l'algorithme calcule l'ensemble $P(s)$ des paires de nœuds qui sont candidat à être ajouté à l'état actuel s. Pour chaque paire p appartenant à $P(s)$, les règles de faisabilité sont évaluées ; si elles sont vérifiées, soit $p = (n, m)$, l'état successeur $s' = s \cup p$ est calculé et l'ensemble du processus s'applique de manière récursive pour s'. Cet algorithme explore le graphe de recherche dans la représentation d'état selon la stratégie de recherche en profondeur. Dans des articles de 2001 [17] et de 2004 [18], les auteurs proposent une modification de l'implémentation de l'algorithme, appelé $VF2$, qui réduit les besoins en mémoire de $O(N^2)$ à $O(N)$, ce qui rend l'algorithme particulièrement intéressant pour travailler avec des graphes de grande taille.

Une autre méthode basée sur les arbres de recherches pour résoudre le problème d'isomorphisme a été proposée par Larroia et Valiente [19]. Dans cet algorithme, les auteurs ont reformulé l'isomorphisme de graphe comme un problème de satisfaction de contrainte (CSP), un problème qui a été profondément étudié dans le cadre de l'optimisation discrète et la recherche opérationnelle. Ainsi, les auteurs appliquent à l'appariement des graphes quelques heuristiques issues de la littérature CSP.

```
Procedure Match(s)
   Input : an intermediate state s ; the initial state s_0 has M(s_0) = ∅
   Output= the mappings between the two graphs.
   If  M(s) covers all the nodes of G_2 THEN
      OUTPUT M(s)
   ELSE
     Compute the set P(s) of the pairs candidate for inclusion in M(s)
   FOREACH (n,m) in P(s)
     IF F(s,n,m) THEN
        Compute the state s' obtained by adding (n,m) to M(s)
       CALL Match(s')
     END IF
    END FOREACH
      Restore the data structure
    END IF
END PROCEDURE
```

Figure 4- 3: Une description de haut niveau de l'algorithme VF2 [17]

Dans [34] Zong Ling propose aussi une méthode pour la résolution du problème d'isomorphisme de sous graphe en reformulant ce problème comme un problème de CSP: le paradigme de la gestion des ressources. Se basant sur un ensemble de stratégies de planification et le concept de l'arbre centré maximal pour la décomposition et le partitionnement du graphe, cette approche réalise moyennement des performances quasi linéaires.

"$Graph\ Explorer$" [24] peut-être répertorié avec les algorithmes d'appariement de graphe qui se repose sur les arbres de recherche. Cet algorithme

repose sur l'appariement des arrêtés. Il convertit le problème d'appariement de graphe en un problème de recherche de chemin dans l'espace de recherche. En utilisant une recherche d'itinéraire optimisé et un backtracking intelligent, cet algorithme évite un très grand nombre d'espaces de recherche invalide et montre une performance linéaire au nombre d'arêtes du sous-circuit motifs. Graph Explorer utilise aussi des graphes étiquetés. Les idées de bases de l'algorithme GE sont:

- L'appariement est basé sur l'appariement des arrêtés plutôt que l'appariement des nœuds comme c'est le cas pour les autres algorithmes TSGM.
- Un algorithme de navigation de graphe convertie le graph motif en une arborescence, qui va être convertie elle-même après en une recherche d'itinéraires.
- Sous la direction de la recherche d'itinéraire, le problème d'appariement des graphes étiqueté est converti en un problème de recherche de chemin dans un espace d'états composé par un espace d'état de sous-graphes partiellement compatibles.
- Les étapes de saut et les étapes de fusion sont soigneusement manipulées pour assurer que la topologie du sous-graphe cible identifié est conforme à la topologie du graphe modèle.
- Une procédure de backtracking intelligent qui fait face à l'état supérieur causant un échec à un état inférieur par un retour arrière direct afin d'éviter les états invalides et inutiles.
- Un arbre intermédiaire est utilisé pour stocker le résultat intermédiaire de chaque étape.

SD [52] est un autre algorithme de backtracking. Il s'appuie sur les informations contenues dans la représentation matricielle de distance des graphes.

Cette représentation est utilisée pour établir le partitionnement initial du graphe. Elle aussi utilisé pendant le backtracking afin de réduire le nombre d'appariements possible dans l'arbre de recherche. Le temps nécessaire pour générer la matrice de distance est $\mathrm{O}(N^3)$ de sorte que la durée d'exécution globale de l'algorithme ne peut pas être inférieur à $\mathrm{O}(N^3)$. Ayant la matrice de distance, le partitionnement initial peut être réalisé en $\mathrm{O}(N^2)$. Le temps requis pour le backtracking dans l'algorithme SD est dans le meilleur des cas $\mathrm{O}(N^2)$ et dans le pire des cas $\mathrm{O}(N.N!)$.

b) Autres techniques

Le plus intéressant algorithme d'appariement de graphe qui n'est pas basé sur les arbres de recherche est $Nauty$[29]. Cet algorithme ne traite que le problème d'isomorphisme, et est considéré par de nombreux auteurs comme l'un des algorithmes les plus rapides. Il est basé sur la théorie des groupes. En particulier, il utilise certains résultats issus de ce cadre théorique pour construire d'une manière efficace le groupe automorphisme de chacun des graphes en entrée. Du groupe automorphe, un étiquetage canonique est dérivé, qui introduit un ordre sur les nœuds qui est uniquement défini pour chaque classe d'équivalence des graphes isomorphes. Ainsi, deux graphes peuvent être vérifiés pour l'isomorphisme simplement en vérifiant l'égalité des matrices d'adjacence de leurs formes canoniques. La vérification de l'égalité peut être calculée en $O(N^2)$, mais la construction de l'étiquetage canonique peut exiger un temps exponentiel dans le pire des cas. Dans [30], il a été vérifié que sous certaines conditions, l'algorithme $Nauty$ peut être surclassé par d'autres algorithmes comme le $VF2$ mentionnés ci-dessus. En outre, $Nauty$ n'exploite pas d'une manière efficace les attributs des nœuds et des arêtes, qui dans de nombreuses applications de PR ces attributs peuvent apporter une contribution précieuse pour réduire le temps d'appariement. Le fait que l'étiquetage canonique doit être calculé séparément pour chaque graphe, indépendamment des autres graphes qui doivent être appariés, peut rendre $Nauty$ vraiment efficace pour

apparier un seul graph avec un plus large ou une base de données fixée de graphes pour lesquels l'étiquetage canonique a été pré-calculé.

Cette propriété est partagée par certains autres algorithmes qui visent spécifiquement à réduire la complexité d'appariement d'un graphe en entrée contre une large librairie de graphes convenablement prétraitée. Parmi, les premiers de ce genre d'algorithme est celui introduit par Bruno Messmer [20]. L'approche proposée est basée une décomposition récursive de chaque graphe de la bibliothèque en plus petits sous-graphes, jusqu'à ce qu'un graphe ayant un seul nœud soit atteint. Le processus d'appariement exploite ensuite le fait que certaines parties communes à plusieurs graphes dans la bibliothèque afin d'éviter de répéter leur comparaison avec le graphe en entrée. De cette façon, le temps d'appariement total a une dépendance linéaire au nombre de graphe existant dans la bibliothèque.

Ming Qiu et coll. [28] ont présenté une approche pour le problème d'isomorphisme de graphe basée sur l'arbre de décision. Contrairement au temps de prétraitement qui est négligé, la complexité du calcul de leur approche est polynomiale le nombre de sommets du graphe d'entrée. De plus la taille de l'arbre de décision est exponentielle ce qui rend l'algorithme non adapté pour la pratique avec des graphes contenant beaucoup de nœuds. Ils ont introduit la notion de l'invariant des nœuds pour le partitionnement de la matrice du graphe en entrée avant la détection de l'isomorphisme. Cela permet de réduire remarquablement la taille de l'arbre de décision et garde la même complexité de l'algorithme. Des expériences pratiques avec des graphes générer aléatoirement sont données pour prouver cela. Ce qui signifie qu'il faut avoir une base de données de graphe, et le travail fréquent est de tester si le nouveau graphe en entrée est un graphe ou sous graphe isomorphe à un graphe motif dans la base de données pour pouvoir ainsi faire du prétraitement.

4.2.1.2. L'appariement inexact

Les contraintes strictes imposées par l'appariement exact sont, dans certaines circonstances trop rigides pour la comparaison de deux graphes. Dans de nombreuses applications, les graphes observés sont soumis à des déformations dues à plusieurs causes: la variabilité intrinsèque des motifs, le bruit dans le processus d'acquisition, la présence d'éléments non déterministes (par exemple les réseaux de neurones) dans les étapes de traitement conduisant à la représentation sous forme de graphe, sont parmi les raisons possibles pour avoir des graphes réels qui diffèrent quelque peu de leurs modèles idéaux. Donc, le processus d'appariement doit être tolérant. Il faut tenir compte des différences dues à la relaxation, dans une certaine mesure, des contraintes qui définissent le type d'appariement. Même en l'absence de déformation attendue, cela peut être utile. Comme nous avons vu, les algorithmes d'appariement exact des graphes (sauf pour les types particuliers des graphes) nécessitent un temps exponentiel dans le pire des cas. Si cela est trop coûteux, il peut être judicieux de se tourner vers des algorithmes qui ne garantissent pas de trouver la meilleure solution, mais qui offre, au moins, une solution approximative dans un délai raisonnable.

Ces deux besoins différents (qui peuvent effectivement être présent) ont conduit à l'élaboration d'algorithmes d'appariement de graphes inexact. Habituellement, dans ces algorithmes l'appariement entre deux nœuds qui ne se satisfont pas aux exigences de préservation des arêtes n'est pas interdit. Au lieu de cela, il est pénalisé par l'assignation d'un coût qui peut prendre en compte d'autres différences (par exemple, parmi les attributs des nœuds ou et arêtes.) Donc, l'algorithme doit trouver un appariement qui minimise le coût d'appariement.

Les algorithmes optimaux d'appariement inexact trouvent toujours une solution qui est le coût minimum global d'appariement. Cela implique que si une solution exacte existe, elle sera trouvée par ces algorithmes. Ils peuvent être considérés donc comme une généralisation des algorithmes d'appariement exact. Les

algorithmes optimaux sont confrontés au problème de la variabilité des graphes et ils ne fournissent pas nécessairement une amélioration du temps de calcul.

Les algorithmes d'appariement approximatifs ou sous-optimaux, au contraire, s'assurent seulement de trouver un minimum local du coût d'appariement. Habituellement, ce minimum n'est pas très loin du minimum global, mais il n'y aucune garantie. Même si une solution exacte existe, ils ne peuvent pas être en mesure de la trouver et pour certaines applications, cela peut ne pas être acceptable. Si c'est acceptable, alors le sou optimalisé de la solution est remboursé abondamment par un temps d'appariement plus court et généralement polynomial.

Un nombre significatif d'algorithmes d'appariement inexact base la définition du coût d'appariement sur un modèle explicite d'erreurs (déformations) qui peuvent se produire (exemple nœuds manquants, etc.), en assignant un coût qui peut être différent pour chaque type d'erreur. Ces algorithmes sont souvent désignés comme correcteur d'erreurs (error-correcting) ou tolérant aux erreurs (error-tolerant). Une autre façon de définir un coût d'appariement est d'introduire un ensemble d'opérations d'édition de graphe (insertion de nœud par exemple, la suppression de nœud, etc.), une fois qu'un coût est assigné à chaque opération, la succession la moins chère des opérations nécessaires pour transformer l'un des deux graphes en l'autre est calculée. Le coût de cette séquence est appelé le coût d'édition du graphe.

a) Techniques basées sur les arbres de recherche

Les arbres de recherche avec retours sur traces peuvent également être utilisés pour l'appariement inexact. Dans ce cas, la fouille est généralement dirigée par le coût de l'adaptation partielle obtenue, et par une estimation heuristique du coût d'appariement pour les nœuds restants. Cette transformation peut être utilisée soit pour élaguer des chemins infructueux, ou encore pour déterminer l'ordre dans lequel l'arbre de recherche doit être traversé, comme dans l'algorithme A^*. Dans ce dernier

cas, si l'heuristique fournit une estimation proche du coût du prochain appariement, l'algorithme trouvera une solution assez rapidement.

Dans [38] et [39], les auteurs nous ont proposé un algorithme basé sur les arbres de recherches. Cet algorithme traite les déformations par un modèle de transformation dans lequel, sous des conditions appropriées, un sous-graphe peut être transformé en un seul nœud. Le modèle de transformation est contextuel, dans le sens où une transformation donnée peut être sélectivement autorisée en fonction des attributs des nœuds et des arêtes voisins.

b) Autres techniques

Dans cette section, nous présenterons brièvement quelques approches pour l'appariement inexact des graphes qui ne sont pas détaillés plus haut. Parmi ces approches on peut citer:

- les méthodes basées sur l'optimisation continue qui visent à transformer le problème d'appariement des graphes, qui est intrinsèquement un problème d'optimisation discret, en un problème d'optimisation continue et non linéaire. Dans différents contextes d'application, cette approche est très attrayante en raison de son coût de calcul extrêmement réduit, qui est habituellement dépendant de façon polynomiale (et avec un exposant faible) de la taille du graphe. Parmi eux on peut citer celle proposée par Van Wyk et coll. [48,49] en 2002. Cette approche est basée sur la théorie de l'espace de Hilbert à noyau reproduisant pour transformer le problème d'appariement de graphes en un problème d'identification de système. Ce dernier est alors résolu par la construction d'un interpolateur RKHS pour approximer la fonction d'appariement inconnue.

- Les méthodes spectrales qui se basent sur le fait que les valeurs propres et les vecteurs propres de la matrice d'adjacence d'un graphe

sont invariantes par rapport aux permutations des nœuds. Parmi les travaux pionniers de cette méthode on peut citer [50].

- Les méthodes de décomposition introduite par Messmer et Bunke pour l'appariement exact des graphes ont été élargies pour le cas d'appariement inexact [51].

5. L'OUTIL DE FILTRAGE DE NETLIST

5.1. Choix de l'algorithme d'isomorphisme de sous graphe

Un ensemble de tests a été fait sur les algorithmes les plus connus pour le problème d'isomorphisme de graphes exact (Ullmann, SD, VF, VF2 et Nauty) [53,54]. Tous ces algorithmes n'imposent aucune restriction concernant la structure du graphe en entrée. S. Voss a mené des études [54] pour explorer les facteurs qui peuvent affecter le temps d'exécution des algorithmes. L'une des théories principales se base sur le fait que l'étape d'étiquetage affecte le temps d'exécution ou non. L'algorithme Nauty a été exclu de cette étude vu qu'il n'est pas assez adapté pour supporter les graphes étiquetés. Dans les deux études citées précédemment, les algorithmes ont été comparés suivant différentes topologies, tailles et étiquetages de nœuds. Les résultats ont montré que VF2 est l'algorithme le plus efficace suivi par VF. Toutefois, ces deux derniers sont touchés par la topologie des graphes en entrée. VF2 été le seul à être affecté par l'étiquetage des nœuds pour toutes les topologies testées, contrairement à VF et Ullmann qui ont montré une variation au niveau de certaines topologies des graphes seulement. L'algorithme d'Ullmann est plus rapide que SD dans le cas ou Ullmann n'est pas affecté par la topologie. L'algorithme SD est le plus cohérent, car il n'est pas affecté ni par la topologie ni par l'étiquetage des nœuds dans tous les cas des tests.

P. Foggia et coll. [53] ont présenté un ensemble de tests pour évaluer les performances de ces algorithmes. La comparaison a été effectuée sur une base de données de graphe générée artificiellement. Les auteurs ont conclu qu'un meilleur algorithme n'existait pas dans l'absolue. En particulier, pour les graphes connectés d'une façon aléatoire, l'algorithme Nauty est le meilleur si les graphes sont assez

denses et/ou de taille assez grande. Pour les graphes les plus petits et très rares, au contraire, VF2 est le plus performant. Dans d'autres cas de graphes réguliers, c'est-à-dire sur des maillages 2D, VF2 est certainement le meilleur algorithme: dans ce cas l'algorithme Nauty n'est même pas capable de trouver une solution pour les graphes ayant plus d'une dizaine de nœuds. Finalement, il est également intéressant de noter que les algorithmes SD, VF et VF2 sont les seuls qui ont toujours été en mesure de trouver une solution au problème d'isomorphisme lors des tests menés par les auteurs, indépendamment de la taille et le type des graphiques à être appariés.

En se basant sur les tests menés dans les travaux de P. Foggia et S. Voss ainsi que les spécifications de l'outil de filtrages et ses besoins. Notre choix de l'algorithme d'isomorphisme de sous-graphe s'est porté sur l'algorithme VF2 de Cordella et col [17,18].

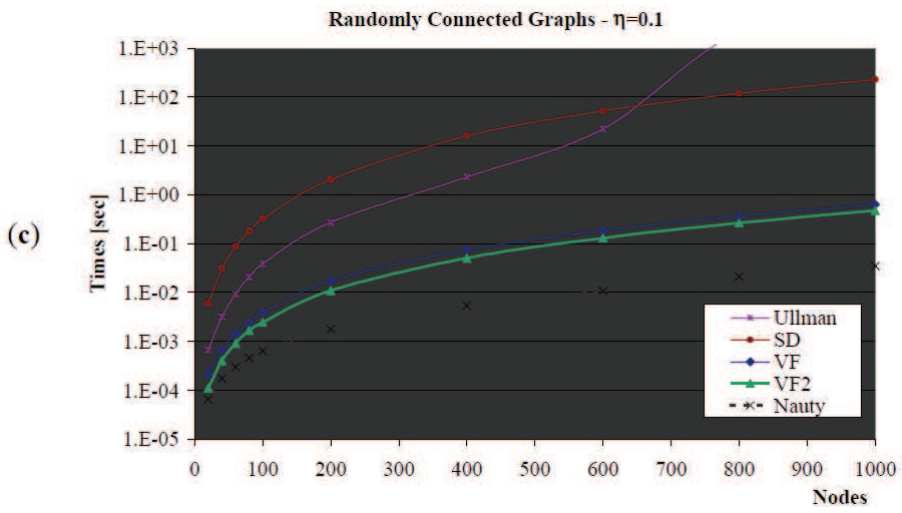


Figure 5 - 1: La performance des cinq algorithmes sur des graphes connectés au hasard [17]

5.2. Conception de l'outil de filtrage de netlist

5.2.1. Diagramme de contexte de l'OFN

La conception de l'outil de filtrage de netlist nécessite l'interconnections avec d'autres module. Le diagramme ci-dessous (Figure 5-2) décrit l'interaction de l'outil de filtrage avec le milieu extérieur:

MentorGraphicsAPI: Ce module fait partie des sous-modules du WaferConnect™. Il permet d'extraire la netlist en format EDIF et/ou XML de l'outil de CAO DxDesigner. Aussi, il contient un ensemble de fonction qui permettent l'interaction avec la schématique du circuit à filtrer en faisant toute sorte de modifications (suppression et/ou ajout de composant et de nets, etc.)

User: Il peut être l'utilisateur du WaferBoard™ ou le concepteur. Dans le cas où on n'utilise pas DxDesigner comme outil de conception de PCB, l'utilisateur doit fournir manuellement la netlist en format EDIF. En plus des patrons de circuit déjà existants dans la base de données, l'utilisateur peut ajouter à n'importe quel moment de nouveaux motifs.

Routing: Ce module fait partie du WaferConnect™ et permet de faire l'opération de routage. Il reçoit de l'OFN la netlist filtrée de tout les composants passifs et la description du footrpint de chaque composant restant.

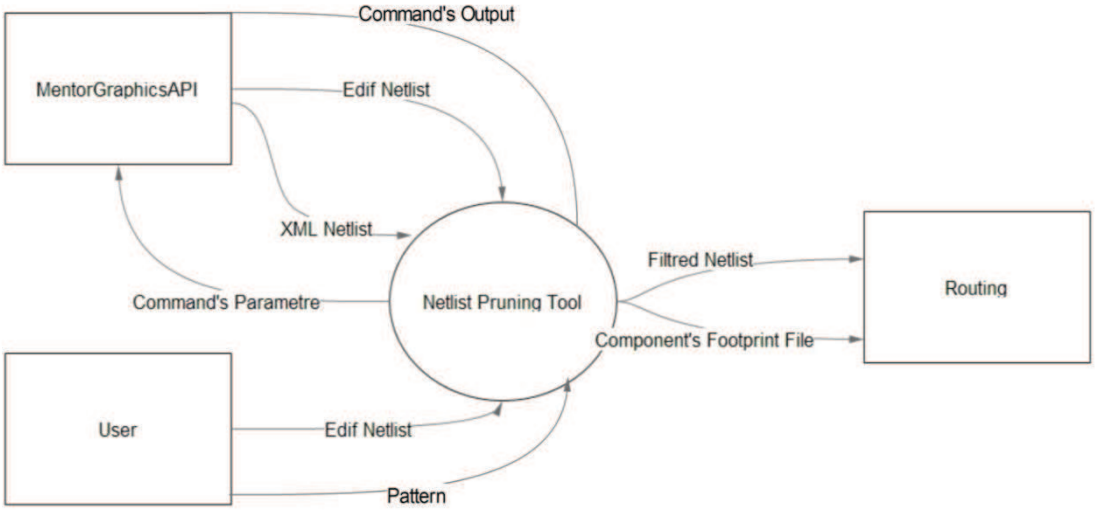


Figure 5 - 2: Diagramme de contexte de l'OFN

5.2.2. Diagrammes préliminaires

Dans cette partie les diagrammes préliminaires de l'OFN seront présentés. La Figure 5-3 décrit la décomposition du L'OFN en plusieurs sous-modules.

Parsing Netlist: Ce module analyse la netlist en entrée afin d'extraire la liste des composants, le type des composants (résistance, capacité, connecteur, etc.), les nets et toutes les informations utiles concernant le schéma de connexion entre les composants.

Matching Pattern: C'est le module principal de l'outil. Il réalise la reconnaissance des motifs de sous-circuit. Ce module fournit comme sortie la liste des composants appariés avec chaque patron de circuit.

Adding Pattern to Database: Ce module permet à l'aide d'une interface graphique de construire un nouveau circuit motif et de l'ajouter à la base de données des sous-circuits motifs.

Generate Output: Dans ce module une nouvelle netlist est générée après le filtrage de la netlist initiale ainsi que le fichier des footprint des composants restants.

Modify DxDesigner Schematic: Chaque modification faite sur la netlist sera traduite directement sur la schématique du circuit en utilisant le logiciel de CAO DxDesigner et l'API de connexion de MentorGraphics.

Modify Netlist: pour chaque motif de sous-circuit trouvé, l'outil effectue des corrections au niveau de la netlist.

Generate Output: À partir de la liste des composants restant après le filtrage ce module genre les fichiers de sorties.

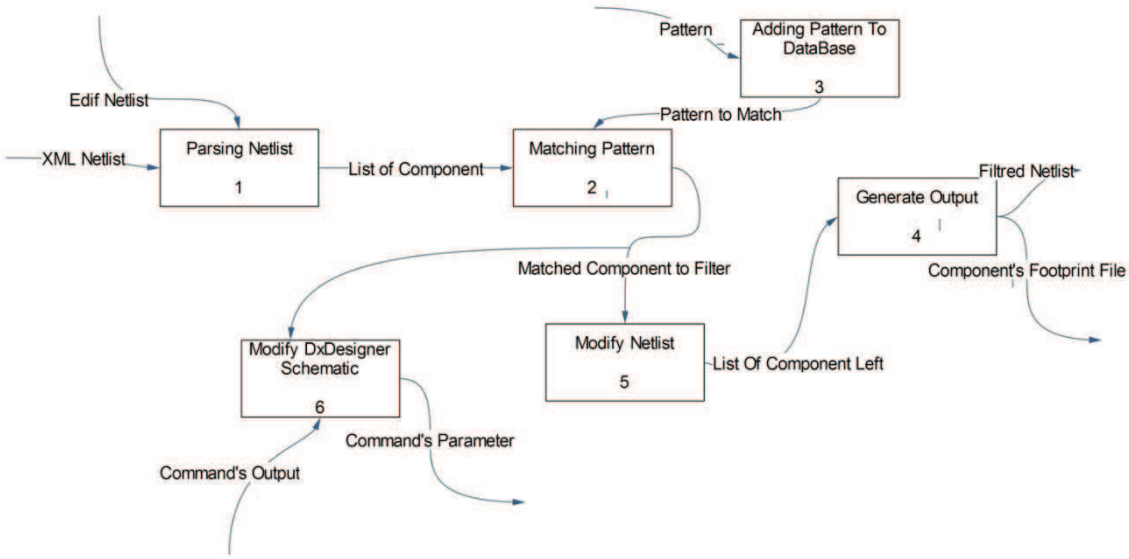


Figure 5 - 3: Diagramme de flots de données de niveau 1 de l'OFN

5.2.3. Diagrammes de décomposition

Dans cette partie les diagrammes de décomposition seront présentés.

La Figure 5-4 illustre les quatre modules suivants:

Edif Parser: Analyse le netlist en format EDIF.

XML Parser: Analyse le netlist en format XML.

Edif DataStructure: Aprés l'analyse de la netlist, toutes les informations récupérées seront sauvegardées dans une structure de données basée sur le format de la netlist.

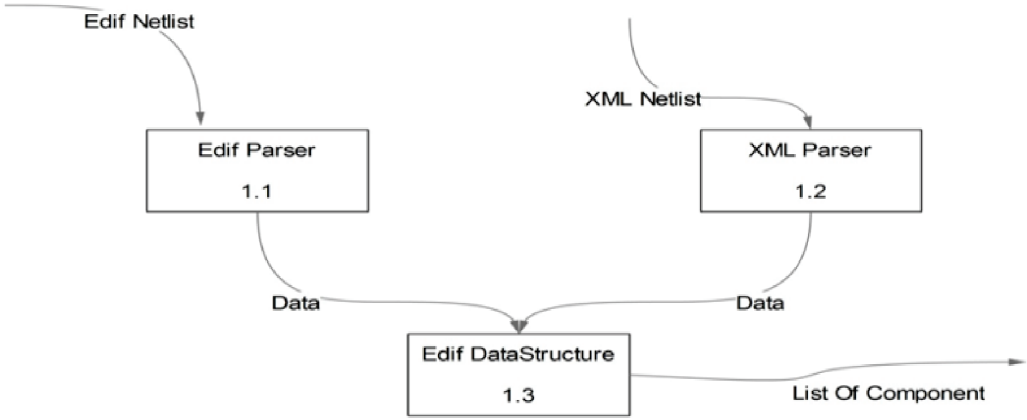


Figure 5 - 4: Diagramme de décomposition du module Parsing Netlist

La décomposition du module Matching Pattern est donnée dans la Figure 5-5:

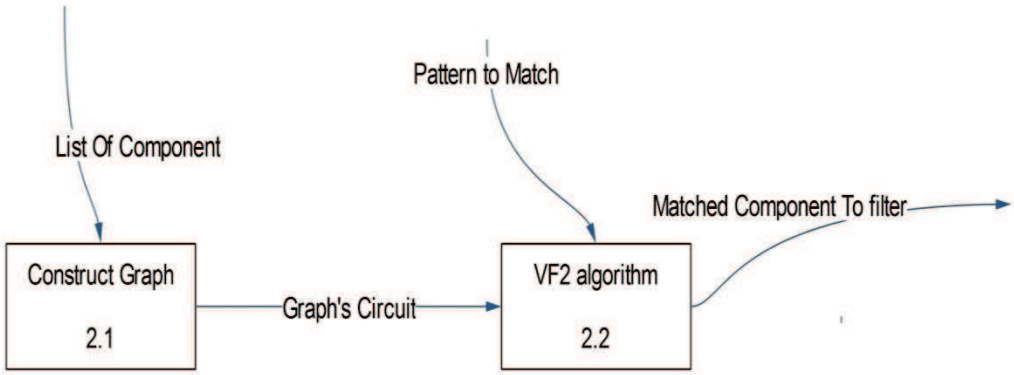


Figure 5 - 5: Diagramme de décomposition du module Matching Pattern

Construct Graph: Ce module utilise la liste des composants extraite à partir de la netlist pour construire le graphe représentant le circuit de l'utilisateur.

VF2 algorithm: L'algorithme VF2 reçoit comme entrée les patrons de circuit à extraire et le graphe représentatif du circuit. Comme sortie, le module génère la liste des composants à filtrer.

5.3. Motifs des sous-circuits

5.3.1. Les résistances pull-up et pull-down

Les résistances pull-up sont utilisées dans des circuits électroniques afin de s'assurer que les entrées pour le système logique s'établissent à un des niveaux logiques attendus si les périphériques externes sont débranchés ou en haute impédance. Ils peuvent également être utilisés comme interface entre deux composants de différents types logiques, éventuellement fonctionnant à différentes tensions d'alimentation. Une résistance pull-down fonctionne de la même manière, mais est reliée à la masse. Il maintient le signal logique proche de zéro volt lorsqu'aucun autre dispositif actif n'est connecté.

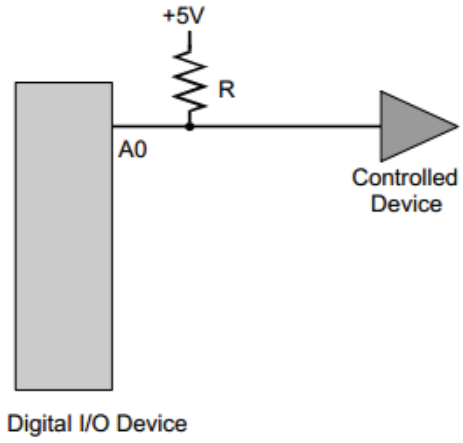


Figure 5 - 6: Configuration d'une résistance pull-up

5.3.2. Les condensateurs de découplage

Un condensateur de découplage est un condensateur utilisé pour découpler une partie d'un réseau électrique (circuit) d'un autre. Le bruit causé par d'autres éléments du circuit est détourné par l'intermédiaire du condensateur, en réduisant l'effet qu'ils ont sur le reste du circuit. Le condensateur de découplage est relié entre la broche d'alimentation d'un composant et la masse du circuit.

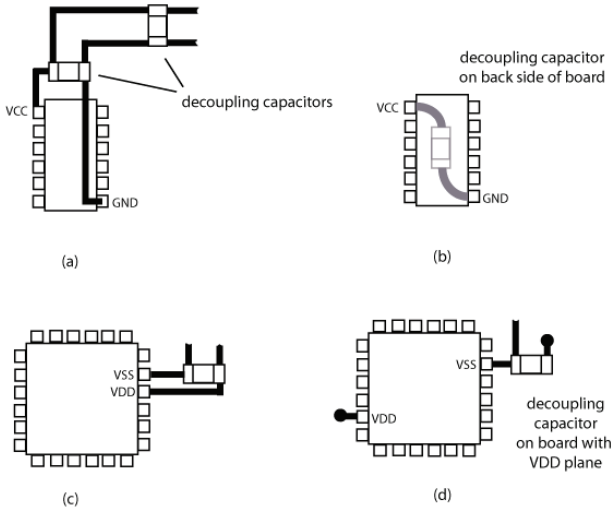


Figure 5 - 7: Exemples de capacité de découplage

5.3.3. Oscillateur à Quartz

Un oscillateur à Quartz est un circuit oscillateur électronique qui utilise la résonnance mécanique d'un cristal de matériau piézoélectrique vibrant pour créer un signal électrique avec une fréquence très précise. Cette fréquence est couramment utilisée pour garder une trace du temps, pour fournir un signal d'horloge stable pour les circuits intégrés numériques, et pour stabiliser les fréquences pour les émetteurs et les récepteurs radio. Le type le plus commun des résonateurs piézoélectriques utilisés est le cristal de quartz, de sorte que les circuits oscillateurs conçus avec sont appelées oscillateurs à quartz.

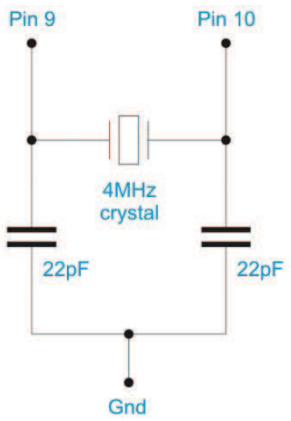


Figure 5 - 8: Exemple d'oscillateur à quartz.

5.4. Vue d'ensemble de la norme EDIF

5.4.1. Introduction

EDIF (Electronic Interchange Format Design) est un format indépendant du fournisseur dans lequel sont stockés des netlists et schématiques électroniques. C'était l'une des premières tentatives pour établir un format d'échange de données neutre pour l'automatisation de la conception électronique (EDA) de l'industrie. L'objectif

était d'établir un format commun à partir duquel les formats propriétaires de systèmes d'EDA pourraient être dérivés. Lorsque les clients nécessitent de transférer des données d'un système à un autre, il est nécessaire d'écrire des traducteurs d'un format à l'autre. Comme le nombre de formats (N) se multiplie, la question du traducteur est devenue un N-squared problème. On s'attendait à ce que, avec EDIF le nombre de traducteurs pourrait être réduits au nombre de systèmes concernés.

Les représentants des entreprises de systèmes EDA: Daisy, Mentor Graphics, Motorola, National Semiconductor, Tektronix, Texas Instruments et l'Université de Californie, Berkeley ont créé le EDIF Steering Committee en Novembre 1983. Plus tard, le professeur Hilary Kahn (1943-2007) de l'Université de Manchester a rejoint l'équipe et a dirigé l'élaboration de la version EDIF 2 0 0 jusqu'à la version finale 4 0 0.

EDIF (Electronic Interchange Format Design) est un format neutre utilisé pour échanger des données de conception électronique. Il existe trois versions de EDIF: EDIF 2 0 0, EDIF 3 0 0 et EDIF 4 0 0. EDIF a été conçu pour aider à résoudre le problème de l'intégration, la migration et l'archivage des données de conception électronique.

Son utilisation principale est comme un format de transfert de données de CAO à CAO. Récemment, il a été amélioré pour inclure tous les aspects de CAD-CAM à un transfert de données. Ce travail a été fait par la US government-funded project appelé ECCE. L'avantage de l'EDIF est que la conception complète peut être représenté dans un fichier qui comprend schématique, Netlist et PCB / MCM

Le format EDIF est une initiative récente permettant de regrouper tous les aspects de la conception VLSI dans une représentation unique. Ceci est utile, non seulement comme un moyen de communication entre le matériel de fabrication, mais aussi comme un format d'échange entre systèmes de CAO, car aucune des

informations de haut niveau n’est perdue. EDIF est conçu pour être à la fois facile à lire par les humains et facile à analyser par des machines.

Fichiers EDIF rassemblent le langage de programmation LISP en raison de l'utilisation de la notation préfixe entre parenthèses. Par exemple, le polygone CIF (Caltech Intermediate Format), Caltech (California Institute of Technology):

P 100 100 150 200 200 100 ;

ressemble à ceci dans EDIF:

(polygon (point 100 100) (point 150 200) (point 200 100))

Toutes les déclarations EDIF se composent d'une parenthèse ouvrante, un mot-clé, certains paramètres, et une parenthèse fermante. Les paramètres peuvent être des d'autres déclarations, ce qui donne à la structure EDIF. En fait, un fichier EDIF contient une seule instruction:

(Paramètres EDIF) où les paramètres sont le circuit décrit.

5.4.2. Version

5.4.2.1. EDIF 2 0 0

La version la plus largement utilisée est EDIF 2 0 0. Il se trouve pratiquement sur toutes les plateformes EDA. EDIF 2 0 0 a été publié en 1987 et est devenu une norme ANSI en 1988. Sa spécification couvre tous les aspects de la conception électronique: schémas, Netlist, PCB, documentation, modèles de simulation, etc. Il a été utilisé avec succès pour échanger des données schématiques et de la connectivité depuis 1987.

5.4.2.2. EDIF 3 0 0

En raison de certaines faiblesses fondamentales dans la version 2 0 0 0 une nouvelle version 3 0 0 0 non compatible avec la précédente a été publié en septembre

1993, compte tenu de la désignation de la norme EIA. Plus tard, il reçoit les certifications ANSI et ISO.

5.4.2.3. EDIF 4 0 0

La version 4 0 0 0 de la norme EDIF reprend la version 3 0 0 0 à laquelle on a rajouté les extensions PCB/MCM. Comme un sur ensemble de EDIF 3 0 0, un fichier EDIF 4 0 0 peut contenir des bibliothèques de symboles, dessins schématiques hiérarchiques et il peut contenir la connectivité complète de la conception. Les extensions de PCB / MCM couvre tous les aspects de bibliothèques physiques de PCB, les règles techniques et les règles de forage des données, etc.

5.4.3. Structure

Un fichier EDIF contient un ensemble de bibliothèques contenant chacune un ensemble de cellules. Chaque cellule peut être décrite avec une ou plusieurs vues qui montrent la cellule sous la forme d'un schéma, mise en page, spécification comportementale, document, et plus encore. Chaque vue possède à la fois une interface et un contenu de sorte qu'il est proprement défini et peut être lié à d'autres vues avec une carte. Les bibliothèques peuvent aussi contenir des informations sur les technologies utilisées. La structure générale d'un fichier EDIF ressemble à ceci:

```
(edif name
   (status information)
   (design where-to-find-them)
   (external reference-libraries)
   (library name
      (technology defaults)
      (cell name
         (viewmap map)
         (view type name
            (interface external)
            (contents internal)
         )
      )     )   )
```

La déclaration « **status** » est utilisée pour suivre les progrès de conception et contient des noms d'auteurs, les dates de modification et les versions du programme. D'autres déclarations « **status** » supplémentaires peuvent apparaître dans chaque déclaration « **library** », « **cell** », et « **view** ». La déclaration « **design** » indique l'endroit où la conception achevée peut être trouvée. Les déclarations « **external** » nomment des bibliothèques qui seront utilisées, mais ne sont pas répertoriées dans ce fichier EDIF. Les blocs « **library** », « **cell** », et « **view** » peuvent être répétés si nécessaire. Il y a aussi une déclaration « **comment** » qui peut être placé à la fin de la plupart des blocs afin d'ajouter de la lisibilité dans le fichier. Voici un exemple d'un fichier EDIF qui illustre davantage le niveau externe:

```
(edif my-design
    (status
        (edifversion 1 0 0)
        (ediflevel 0)
        (written
            (timestamp 1985 4 1 11 16 6)
            (accounting author "Steven Rubin")
            (accounting location "Palo Alto")
            (accounting program "Electric")
            (comment "timestamp contains year, month, day, hour,")
            (comment "                  minute, and second")
        )
    )
    (design hot-dog-chip
        (qualify hot-dog-library top-cell)
        (comment "look for top-cell in hot-dog-library")
    )
    (external pad-library pla-library)
    (library hot-dog-library
        (technology 3-micron-nMOS
            ...
        )
        (cell top-cell
            (viewmap... )
            (view masklayout real-geometry
                (interface ... )
                (contents... )
            )
            (view schematic more-abstract
```

```
            (interface... )
            (contents... )
        )
    )
  )
)
```

La déclaration « **written** » du bloc « **status** » peut-être répétée pour afficher tous les auteurs et les événements de mise à jour.

5.5. Analyse des résultats

L'outil de filtrage de netlist a été créé sous la plateforme [55] Qt Creator et codé en C++ et en utilisant des librairies standard QT. Cette section présente une série d'expériences qui montrent le fonctionnement et les performances de l'outil sur une variété de netlists générés. Ces netlists ont été générés en combinant des netlists réelles fournies par l'outil de conception assistée Altium Designer [56]. Toutes les netlists décrivant à la fois les circuits principaux et les patrons de circuits sont décrites dans le format EDIF 2 0 0. Les résultats expérimentaux sont représentés dans le tableau 5-1. Le temps d'exécution de la reconnaissance est le temps mis par l'outil pour la reconnaissance de sous-circuits sans compter le temps de préparation des fichiers EDIF. Quant au temps d'exécution globale couvre tout le temps mis par tous les sous modules à s'exécuter.

Nombre de composants	Nombre de nets	Temps exécution global	Temps d'exécution pour la reconnaissance
1982	2803	1,01	0,059
1860	3248	1,033	0,067
1985	3313	1,079	0,077
2275	3564	1,195	0,082
2269	3499	1,223	0,087
2309	3120	1,321	0,098
2749	3406	1,334	0,088
2203	3720	1,378	0,104
2322	3505	1,41	0,107

2362	4566	1,424	0,138
2522	3692	1,465	0,113
2712	4254	1,472	0,12
2465	3961	1,515	0,124
2371	3068	1,534	0,104
2915	3515	1,614	0,119
3201	4473	1,648	0,133
3120	4311	1,68	0,138
2973	3903	1,682	0,14
3325	4443	1,733	0,152
2859	3890	1,8	0,142
3157	4032	1,803	0,142
2836	4613	1,807	0,168
3243	3939	1,815	0,139
2938	4610	1,856	0,178
3215	4476	1,901	0,163
3354	4438	1,928	0,159
3500	4495	1,949	0,168
3363	4797	1,961	0,17
3216	5275	2,036	0,207
3543	4991	2,053	0,192
3933	5184	2,06	0,197
3708	4660	2,085	0,192
3821	4537	2,087	0,188
3621	5271	2,13	0,201
3745	5210	2,162	0,199
3618	5287	2,202	0,207
3806	5231	2,208	0,209
3724	5596	2,237	0,24
3759	5457	2,255	0,236
3531	4715	2,258	0,204
4010	5056	2,28	0,223
3963	5292	2,285	0,22
4049	5340	2,31	0,211
4009	5211	2,311	0,223
4092	5603	2,377	0,25
4102	5422	2,387	0,231
3960	5689	2,388	0,268
4027	5747	2,389	0,238

3783	5979	2,398	0,278
3998	6172	2,485	0,278
4231	6217	2,517	0,272
4209	5219	2,542	0,237
4540	5817	2,563	0,254
4034	5799	2,59	0,249
4482	5965	2,596	0,277
4641	5819	2,608	0,279
4423	5565	2,656	0,251
4759	5679	2,662	0,261
4371	6257	2,833	0,329
4444	6220	2,853	0,311
4849	6820	3,135	0,344
4824	6945	3,148	0,372
5518	7717	3,182	0,407
4973	6780	3,197	0,366
5098	6998	3,25	0,384
7089	10 228	4,455	0,668
7634	10 450	4,862	0,73
7830	10 201	4,866	0,718
7778	10 435	4,866	0,672
7547	10 913	4,886	0,811
7493	10 996	4,951	0,802
8112	11 064	5,175	0,812
8220	11 375	5,387	0,859
8787	11 430	5,662	0,872
8892	11 753	5,764	0,913
8927	12 111	5,844	0,951
9097	11 791	5,867	0,952
9208	11 850	5,968	1,005
9045	12 650	6,151	1,061
9254	12 715	6,303	1,132
9614	12 760	6,54	1,076
9920	13 968	6,9	1,256
13 507	17 267	10,54	2,107
14 471	19 342	11,764	2,425
14 696	20 104	11,956	2,527
14 620	20 502	12,004	2,673
13 826	20 820	12,043	2,84

17 297	22 233	14,92	3,242
17 076	23 261	14,969	3,511
17 009	23 290	14,983	3,549
17 710	23 690	15,666	3,662
17 314	24 451	15,842	4,067
17 134	25 538	16,035	4,253
18 165	25 542	16,467	3,968
17 819	25 516	16,684	4,348
18 941	27 391	18,317	4,912
20 109	27 587	19,187	4,801
20 981	28 715	20,625	5,567
21 741	28 349	23,11	5,946
26 698	37 171	32,236	9,019
28 258	38 511	34,503	9,672
28 528	39 672	35,7	10,033
29 752	40 389	37,433	10,834
30 972	41 363	38,473	10,916
33 596	41 491	42,128	12,016
33 720	44 976	45,532	13,383
34 296	47 047	46,536	14,205
36 119	47 925	49,258	15,217
36 258	48 460	52,336	16,289
37 166	50 062	53,193	16,302
40 357	50 103	59,346	17,74
40 854	55 239	66,506	21,169
41 638	56 389	67,935	21,642
41 406	59 577	71,888	24,791
43 956	61 122	75,758	25,349
45 993	61 669	80,186	25,725
46 148	61 939	81,508	26,309
48 642	64 094	89,327	29,301
49 262	64 455	92,653	30,516
51 072	67 750	97,781	33,01
51 233	72 467	101,132	36,102

Table 5 - 1: Runtime used for netlist pruning.

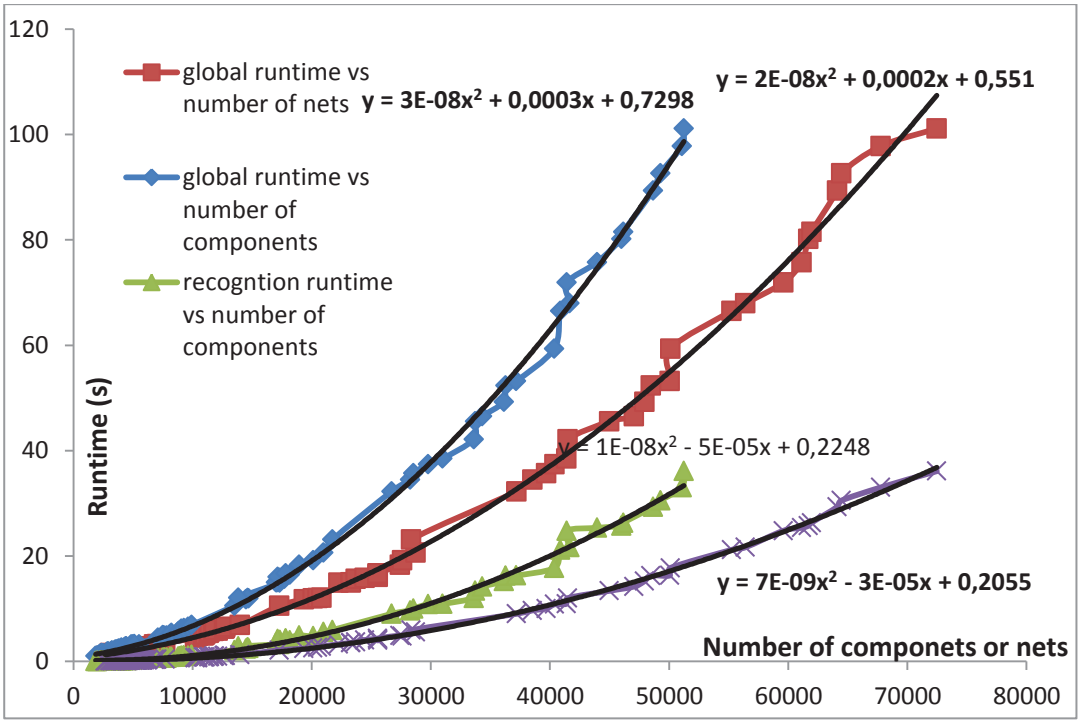


Figure 5 - 9: Performance en termes de temps d'exécution de l'OFN

Comme prévu, l'outil de filtrage de netlist reconnait tous les modèles de sous circuit qu'on voulait retrouver dans toutes les netlist testés. La figure 5-9 montre l'exécution de l'outil en tenant compte du temps mis pour préparer le circuit principal et le modèle. Pour une taille fixe du circuit modèle, le temps de reconnaissance pour des tailles croissantes du circuit principale montre une performance polynomiale. Cette croissance polynomiale est due à la complexité de l'algorithme VF2 qui est $O(N^2)$[18]. La forte croissance du temps d'exécution avec celle de la taille du circuit principal est principalement due à la complexité de la configuration du circuit principale en recueillant les informations relatives aux connexions existantes entre les composantes à partir du fichier EDIF. Comme c'est montré dans le tableau 5-1, il a été mesuré pour le circuit comportant 51 233 composants qu'environ 64 % du temps d'exécution soit consacré à la configuration du circuit principal.

6. CONCLUSION

Le travail présenté dans ce mémoire porte sur les études, les travaux de recherches et les analyses qui ont été effectués durant la conception et l'implémentation d'un outil de filtrage de netlist pour un système de prototypage rapide le WaferBoard™. La pertinence du sujet et des travaux de recherches réalisés s'exprime d'une part dans l'utilité de l'outil de filtrage de netlist pour la recherche et le rôle de l'outil de filtrage intégré à la carte de prototypage et d'autres parts dans l'originalité et les nouveautés technologiques apportées par ce projet de recherche.

Utilisé dans le système de prototypage WaferBoard™, l'outil de filtrage réduit d'une manière considérable le temps que consacre l'utilisateur pour fournir une netlist décrivant les composants déposés sur la surface du WaferIC™. En effet le concepteur du système n'aura plus à repasser à travers les schématiques de son circuit afin d'éliminer les circuits passifs et refaire les connexions entre les composants restant manuellement. Il aura juste à introduire à notre outil sa netlist initiale et les opérations de filtrage seront exécutées automatiquement. En dehors du cadre du projet DreamWafer™, l'outil de filtrage pourra facilement s'interfacer avec des outils de conception de circuits électroniques comme c'est le cas dans notre étude (connexion avec le logiciel DxDesigner de Mentor Graphics). Il sera utilisé pour filtrer des netlist dans le cadre d'autres systèmes ou tout simplement pour retrouver des instances de sous-circuits dans tous systèmes électroniques.

Le défi principal qui a motivé ce mémoire a été de développer un outil pour le filtrage de netlist des circuits électronique se basant sur la notion d'extraction de sous circuits électroniques qui elle-même se base sur la théorie des graphes et l'isomorphisme de graphes. Nous avons amorcé notre recherche par une étude systématique du problème d'extraction des sous-circuits. La méthode la plus utilisée pour résoudre ce problème se base sur la théorie des graphes et plus précisément l'isomorphisme de sous graphes. Ensuite, nous avons développé une manière de

modéliser les circuits électroniques en forme de graphe ou les composants représentent les nœuds des graphes et les connexions entre les composants représentent les liens entre les nœuds (arêtes). La description de ces circuits est tirée d'une netlist écrite dans le format standard EDIF. En nous basant sur ces idées ainsi que sur l'algorithme VF2 nous avons réussi à développer un outil capable de retrouver toutes les instances de sous circuits à l'intérieur d'un autre circuit.

Une revue de littérature portant sur l'extraction des sous circuits électroniques et l'appariement des graphes a été élaborée. Cette étude de l'état de l'art nous a permis de voir les éléments essentiels et nécessaires pour le développement de notre outil. Nous avons présenté quelques approches pour résoudre le problème d'isomorphisme de sous graphes dans la littérature.

Lors des tests de performance réalisés sur notre outil de filtrage, nous avons seulement utilisé des modèles de sous circuits de petite taille contenant quelques trois à quatre composants. Il serait intéressant d'étendre nos tests en considérant des motifs de circuit plus large que celle utilisée dans ce mémoire.

Bibliographie

[1] *D.H. Rouvray, A.T. Balaban. Chemical applications of graph theory, in: R.J. Wilson, L.W. Beineke (Eds.), Applications of Graph Theory, Academic Press, New York,1979, pp. 177-21.*

[2] *Ho-Yon Kim, Jin H. Kim, Hierarchical random graph representation of handwritten characters and its application to Hangul recognition, Pattern Recognition, Volume 34, Issue 2, February 2001, Pages 187-201.*

[3] *H Bunke, G Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognition Letters, Volume 1, Issue 4, May 1983, Pages 245-253.*

[4] *Hartmut Ehrig, Annegret Habel, Hans-Jörg Kreowski, Introduction to graph grammars with applications to semantic networks, Computers & Mathematics with Applications, Volume 23, Issues 6–9, March–May 1992, Pages 557-572*

[5] *E. K. Wong, Three-dimensional object recognition by attributed graphs, in: H. Bunke, A. Sanfeliu (Eds.), Syntactic and Structural Pattern Recognition Theory and Applications, World Scientific, Singapore, 1990, pp. 381-414.*

[6] *Pelz, G. ; Roettcher, U. ; , "Pattern matching and refinement hybrid approach to circuit comparison," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.13, no.2, pp.264-276, Feb 1994*

[7] *Conn, A.R.; Elfadel, I.M. ; Molzen, W.W., Jr.; O'Brien, P.R. ; Strenski, P.N. ; Visweswariah, C. ; Whan, C.B. ; , "Gradient-based optimization of custom circuits using a static-timing formulation," Design Automation Conference, 1999. Proceedings. 36 th , vol., no., pp.452-459, 1999*

[8] *Norman, R. ; Valorge, O. ; Blaquiere, Y. ; Lepercq, E. ; Basile-Bellavance, Y. ; El-Alaoui, Y. ; Prytula, R. ; Savaria, Y. ; , "An active reconfigurable*

circuit board," Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on , vol., no., pp.351-354, 22-25 June 2008

[9] Laflamme-Mayer, N. ; Sawan, M. ; Blaquiere, Y. ; , "A dual-power rail, low-dropout, fast-response linear regulator dedicated to a wafer-scale electronic systems prototyping platform," New Circuits and Systems Conference (NEWCAS), 2011 IEEE 9th International , vol., no., pp.438-441, 26-29 June 2011

[10] Norman, R.; Lepercq, E. ; Blaquiere, Y. ; Valorge, O. ; Basile-Bellavance, Y. ; Prytula, R. ; Savaria, Y. ; , "An interconnection network for a novel reconfigurable circuit board," Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on , vol., no., pp.129-132, 22-25 June 2008.

[11] Laflamme-Mayer, N. ; Valorge, O. ; Blaquiere, Y. ; Sawan, M. ; , "A low-power, small-area voltage reference array for a wafer-scale prototyping platform," NEWCAS Conference (NEWCAS), 2010 8th IEEE International , vol., no., pp.189-192, 20-23 June 2010

[12] Valorge, O.; Nguyen, A.T. ; Blaquiere, Y. ; Norman, R. ; Savaria, Y. ; , "Digital signal propagation on a wafer-scale smart active programmable interconnect," Electronics, Circuits and Systems, 2008. ICECS 2008. 15 th IEEE International Conference on , vol., no., pp.1059-1062, Aug. 31 200-Sept. 3 2008

[13] http://www.dreamwafer.com/WaferBoard.pdf, Décembre 2011

[14] Blaquiere, Y. ; Savaria, Y. ; El Fouladi, J.; , "Digital Measurement Technique for Capacitance Variation Detection on Integrated Circuit I/Os," Electronics, Circuits and Systems, 2007. ICECS 2007. 14 th IEEE International Conference on , vol., no., pp.42-45, 11-14 Dec. 2007

[15] Lepercq, E. ; Blaquiere, Y. ; Norman, R. ; Savaria, Y. ; , "Workflow for an electronic configurable prototyping system," Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on , vol., no., pp.2005-2008, 24-27 May 2009.

[16] J.L. Gross and J. Yellen. Graph Theory and its applications. Chapman and Hall, CRC Press, 2006.

[17] LP Cordella, P Foggia, C Sansone, M Vento, "An improved algorithm for matching large graph", Proceedings of the 3rd IAPR TC-15 Workshop on Graph based ..., 2001

[18] Cordella, L.P.; Foggia, P. ; Sansone, C. ; Vento, M. ; , "A (sub)graph isomorphism algorithm for matching large graphs," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.26, no.10, pp.1367-1372, Oct. 2004

[19] J. Larrosa and G. Valiente, Constraint satisfaction algorithms for graph pattern matching, Math. Struct. Comput. Sci. 12 (2002) 403-422.

[20] Messmer, B.T. ; Bunke, H. ; , "Efficient subgraph isomorphism detection: a decomposition approach ," Knowledge and Data Engineering, IEEE Transactions on , vol.12, no.2, pp.307-323, Mar/Apr 2000.

[21] Marcello Pellilo, Kaleem Siddiqi, Stewen W. Zuker, "Continuous-based Heuristics for Graph and Tree Isomorphisms, with Application to Computer Vision"

[22] Huaijun Qiu, Edwin R. Hancock, Graph matching and clustering using spectral partitions, Pattern Recognition, Volume 39, Issue 1, January 2006, Pages 22-34

[23] Zong Ling; Yun, D.Y.Y. ; , "An efficient subcircuit extraction algorithm by resource management ," ASIC, 1996., 2nd International Conference on , vol., no., pp.9-14, 21-24 Oct 1996

[24] Yin-Tang Dai ; Shi-Yong Zhang; , "A fast labeled graph matching algorithm based on edge matching and guided by search route," Wavelet Analysis and Pattern Recognition, 2009. ICWAPR 2009. International Conference on , vol., no., pp.1-7, 12-15 July 2009

[25] Ohlrich, M. ; Ebeling, C. ; Ginting, E. ; Sather, L. ; , "SubGemini: Identifying SubCircuits using a Fast Subgraph Isomorphism Algorithm," Design Automation, 1993. 30th Conference on , vol., no., pp. 31- 37, 14-18 June 1993

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

[27] Cordella, L.P. ; Foggia, P. ; Sansone, C. ; Vento, M. ; , "Performance evaluation of the VF graph matching algorithm," Image Analysis and Processing, 1999. Proceedings. International Conference on , vol., no., pp.1172-1177, 1999

[28] Ming Qiu ; Haibin Hu ; Qingshan Jiang; Hailong Hu; , "A New Approach of Graph Isomorphism Detection Based on Decision Tree," Education Technology and Computer Science (ETCS), 2010 Second International Workshop on , vol.2, no., pp.32-35, 6-7 March 2010

[29] B.D. McKay, "Practical Graph Isomorphism," Congressus Numerantium, vol. 30, pp. 45-87, 1981.

[30] P. Foggia, C. Sansone and M. Vento, A performance comparison of five algorithms for graph isomorphism, in Proc. 3 rd IAPR-TC15 Workshop Graph-Based Representations in Pattern Recognition, 2001, pp. 188-199.

[31] A. K. C. Wong, M. You and S. C. Chan, An algorithm for graph optimal monomorphism, IEEE Trans. Syst. Man Cybern. 20 (1990) 628-638.

[32] L. P. Cordella, P. Foggia, C. Sansone, F. Tortorella and M. Vento, Graph matching: a fast algorithm and its evaluation, in Proc. 14 th Int. Conf. Pattern Recognition, 1998, pp. 1582-1584.

[33] Cordella, L.P. ; Foggia, P. ; Sansone, C. ; Vento, M. ; , "Fast graph matching for detecting CAD image components," Pattern Recognition, 2000. Proceedings. 15 th International Conference on , vol.2, no., pp.1034-1037 vol.2, 2000

[34] Z, Ling and D. Y. Y. Yun, "Solving Subgraph Isomorphism Problems by an Effective Planning Approach ", was accepted by the Pacific Rim International Conference on Artificial Intelligence (PRICAI), August 26-30, 1996, Cairns, Australia

[35] A. V. Aho, J. E. Hopcroft and J. D. Ullman, The Design and Analysis of Computer Algorithms (Addison Wesley, 1974).

[36] J. E. Hopcroft and J. Wong, Linear time algorithm for isomorphism of planar graphs, in Proc. 6th Annual ACM Symp. Theory of Computing, 1974, pp. 172-184.

[37] Eugene M. Luks, Isomorphism of graphs of bounded valence can be tested in polynomial time, Journal of Computer and System Sciences, Volume 25, Issue 1, August 1982, Pages 42-65.

[38] L. P. Cordella, P. Foggia, C. Sansone and M. Vento, An efficient algorithm for the inexact matching of ARG graphs using a contextual transformational model, in Proc. 13 th Int. Conf. Pattern Recognition, 1996, pp. 180-184.

[39] L. P. Cordella, P. Foggia, C. Sansone and M. Vento, Subgraph transformations for inexact matching of attributed relational graphs, Computing 12(Suppl.) (1997) 43-52.

[40] Hardware Software System Co-Verification of an Active Reconfigurable Board with System CVHDL; Basile-Bellavance et al., IEEE/ICECS International Conference on Electronics, Circuits, and Systems, Malta, Aug 31-Sep 3, 2008

[41] 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

[42] Huang, K.-T. ; Overhauser, D. ; , "A novel graph algorithm for circuit recognition," Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on , vol.3, no., pp.1695-1698 vol.3, 30 Apr-3 May 1995

[43] Vijaykrishnan, N. ; Ranganathan, N. ; , "SUBGEN: a genetic approach for subcircuit extraction," VLSI Design, 1996. Proceedings., Ninth International Conference on , vol., no., pp.343-345, 3-6 Jan 1996

[44] Wei-Hsin Chang ; Shuenn-Der Tzeng; Chen-Yi Lee; , "A novel subcircuit extraction algorithm by recursive identification scheme," Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on , vol.5, no., pp.491-494 vol. 5, 2001

[45] 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.

[46] Nian Zhang ; Wunsch, D.C. ; , "Speeding up VLSI Layout Verification Using Fuzzy Attributed Graphs Approach," Fuzzy Systems, IEEE Transactions on , vol.14, no.6, pp.728-737, Dec. 2006

[47] D. Conte, P. Foggia, C. Sansone, and M. Vento, "Thirty Years Of Graph Matching In Pattern Recognition", presented at IJPRAI, 2004, pp.265-298.

[48] B. J. van Wyk and M. A. van Wyk, Non-Bayesian graph matching without explicit compatibility calculations, in Proc. Joint IAPR Int. Workshops SSPR and SPR, 2002, pp. 74-82

[49] B. J. van Wyk, M. A. van Wyk and H. E. Hanrahan, Successive projection graph matching, in Proc. Joint IAPR Int. Workshops SSPR and SPR, 2002, pp. 263-271.

[50] S. Umeyama, An eigendecomposition approach to weighted graph matching problems, IEEE Trans. Patt. Anal. Mach. Intell. 10 (1988) 695-703.

[51] B. T. Messmer and H. Bunke, A new algorithm for error-tolerant subgraph isomorphism detection, IEEE Trans. Patt. Anal. Mach. Intell. 20 (1998) 493-504.

[52] D.C. Schmidt, L.E. Druffel, "A Fast Backtracking Algorithm to Test Directed Graphs for Isomorphism Using Distance Matrices," Journal of the Association for Computing Machinery, 23, pp. 433-445, 1976.

[53] P. Foggia, C. Sansone, M. Vento, "A Performance Comparison of Five Algorithms for Graph Isomorphism," Proceedings of the 3rd IAPR-TC15 Workshop on Graph based Representation (gbr2001), Italy, 2001.

[54] S. Voss, Jaspal Subholk, "Performance of General Graph Isomorphism Algorithm" Coe College, 2009

[55] Qt Creator and Library, http://qt.nokia.com, mai 2011

[56] http://www.altium.com, mai 2011.

Annexe A

A Netlist Pruning Tool For an Electronic System Prototyping Platform

Karim Baratli
Comp. Sci. & Eng. Dept.
U. du Québec en Outaouais
Gatineau, Canada
bark07@uqo.ca

Ahmed Lakhssassi
Comp. Sci. & Eng. Dept.
U. du Québec en Outaouais
Gatineau, Canada
ahmed.lakhssassi@uqo.ca

Yves Blaquière
Comp. Sci. Dept.
U. du Québec à Montréal,
Montréal, Canada
blaquiere.yves@uqam.ca

***Abstract*— The increasing complexity of electronic systems makes the recognition of subcircuits in a main circuits a desirable feature to several Electronic Design Automation tools (EDA). This paper presents an efficient Netlist Pruning Tool (NPT) for an electronic system prototyping platform. The proposed algorithm first parses any EDIF netlist and represents the circuit into a graph. Then a subgraph isomorphism algorithm is applied for detecting the subcircuits' patterns and for each pattern detected appropriate actions are triggered. NPT was successfully applied to several PCB netlists to prune typical pull-up/pull-down resistors and decoupling capacitors, such as a netlist with up to 51233 components and 70354 nets with a runtime of 101.32 s.**

***Index Terms*— IC, Netlist, Filtering, Pruning, EDA, CAD, EDIF, pattern, subgraph, isomorphism.**

I. INTRODUCTION

The notion of extracting a subcircuit from a large netlist is widely used in simulation, verification and testing of integrated circuit (IC) computer-aided design (CAD) [1] - [2] unlike during electronic circuit design. For example, finding a subcircuit in a main circuit is a frequent task to modify or delete the subcircuit, which may exist everywhere in the main circuit. This process could be manageable when applied on small netlists but becomes tedious on very large ones if it is done manually. The subcircuit finding problem is defined as follows: Given a subcircuit SC and a main circuit C much larger than SC, the problem consists of detecting whether SC exists in C and modifying SC in C according to adaptation rules. The detection problem was extensively studied and many algorithms were suggested [3, 4, 5] and all of them use as input flattened transistor level netlists.

An appropriate approach to represent a circuit is to use a graph such that its nodes and edges represent devices and nets respectively. Thus, subcircuit recognition can be considered as the task of finding the structure of a small model graph representing the subcircuit in a large graph describing the whole circuit. In this way, any effective algorithm for subgraph isomorphism can be applied to solve the subcircuit extraction problem. Subgraph isomorphism is a notion that has been intensively used in different fields such as 3-D object recognition [6] and pattern recognition [7]. Subgraph isomorphism has been proven to be a NP-complete problem. In [8], the authors propose an approach to this problem based on a decision tree. The authors use the vertex invariants of a graph to reduce the tree size, letting them outperform the similar previous approaches [9]. The VF algorithm proposed by Codella et al [10,11] is based on a depth-first search strategy to reduce computational complexity. It is based on the analysis of the sets of nodes adjacent to the ones already considered in the partial mapping. This algorithm is suitable for working with very large graphs. The VF2 algorithm [12,13] is an improved version of the VF algorithm. Based on the performance comparison made between different subgraph isomorphism algorithms [14], a version of VF2 adapted for the needs of the electronic system prototyping platform will be used in our tool for detecting the subcircuits.

This paper presents an Object Oriented Framework tool for pruning electronic system netlists. It is shown that it is efficiently useful for the WaferBoard™, a platform for rapid electronic system prototyping [15] that reduces significantly the time to market of electronic systems. This platform allows interconnecting the user's integrated circuits (uICs), which are manually and randomly deposited on its surface. The active surface includes an interconnection network linking a large number of contact points (called NanoPads). Those NanoPads allow binding the balls of uICs as a PCB. Each of the NanoPads can be configured as a signal I/O or as a programmable power supply [16]. The active surface detects

and maps the component contacts, and the internal interconnection network is then dynamically configured to establish any specified set of connections between the pins/solder balls of those uICs, e.g. according to a user netlist.

An electronic system is not composed solely of ICs but also includes different passive components such as resistors, capacitors, LEDs and connectors. However, the WaferBoard™ does not have to support those. The passive components can effectively be simulated by configuring the NanoPads that are in contact with the uIC pins. In order for the user to debug his electronic system described with a PCB design tool with the WaferBoard™, a new netlist simultaneously adapted to the WaferBoard™ specification and keeping the same functionality of the user's circuit has to be generated. For example, our tool must prune the netlist from passive circuits like pull-up resistors, decoupling capacitors, resonant circuits and connectors and generate a new one made of integrated circuits components and trigger the appropriate actions to simulate those pruned subcircuits.

Independently of its use with the WaferBoard™, the tool presented in this paper could be integrated in any schematic entry tool for circuit pattern search.

Section 2 describes the concept of the netlist pruning tool and its process. The implementation details and some experimental results are detailed in section 3. Finally, section 4 presents some concluding remarks.

II. ALGORITHM DESIGN OF THE NETLIST PRUNING TOOL

The tool takes as input an electronic system netlist, such as an EDIF file generated by a PCB schematic entry tool. The proposed algorithm contains four principle steps: circuit setup, subcircuit pattern setup, identification operation and generating the new netlist in execution order. The following subsections describe them.

A. *Circuit Setup*

In this step all the useful information of the target circuit from the EDIF netlist description are collected to speed up and facilitates the identification operation. It is divided into two passes.

```
...
(Instance C1(Property Description (String "Capacitor")))
(Instance R1(Property Description (String "Resistor" )))
(Instance R2(Property Description (String "Resistor" )))
(Instance R3(Property Description (String "Resistor" )))
(Instance R4(Property Description (String "Resistor" )))
(Instance U1) (Instance U2)
(Net (rename &5V "5V")(Joined (PortRef &2 (InstanceRef
    C1))(PortRef &1 (InstanceRef R1))(PortRef &2
    (InstanceRef R2))))
(Net GND(Joined(PortRef &1 (InstanceRef C1))(PortRef &2
    (InstanceRef R3))(PortRef IO_RST_O (InstanceRef
    U2))(PortRef IO_STB_O (InstanceRef U2))))
(Net NetR1_2(Joined(PortRef &2 (InstanceRef R1))(PortRef
    &39 (InstanceRef U1))))
(Net NetR2_1(Joined(PortRef &1 (InstanceRef R2))(PortRef
    &36 (InstanceRef U1))))...
```

Figure 1. A section of the EDIF file of the netlist in Fig. 2(a)

The first pass analyzes the EDIF file Fig. 1 and extracts circuit information. This operation is done by parsing the netlist and searching for specific keyword, which contains the needed information. Typical keywords are component instance names, the property in which the component type (resistor, capacitor…) is found and net name. The second pass constructs the graph representing the main circuit Fig. 2(b). A node is created for each component instance (with two labels: the instance name and its type) and for power nets, such as GND or VDD. An edge is created for each net.

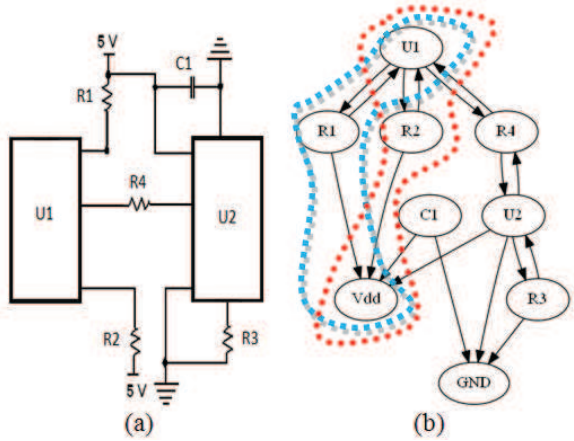


Figure 2. Example of (a) a circuit with (b) its graph.

B. *Subcircuit Pattern Setup*

Same way, the patterns' graphs are created using EDIF netlist. Fig. 3 shows an example of a pull-up resistor pattern and its graph model.

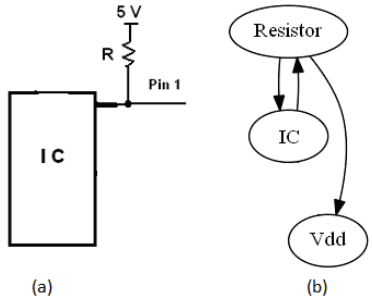


Figure 3. (a) Pull-up resistor pattern and (b) its graph.

C. *Identification Operation*

In this phase, the subgraph matching algorithm is invoked. The subgraph isomorphism problem is defined as follows: Given two graphs, G1=(N1,B1) and G2=(N2,B2), G2 is isomorphic to a subgraph H of G1 if there exists a one to one mapping of the nodes of G2 onto the nodes H such that all the corresponding edge adjacencies are preserved.

The VF2 algorithm was adapted to our application for the matching process. It uses the State Space Representation (SSR), according to which a state *s* represents a partial mapping. This matching process consists of the determination of a mapping M that associates nodes of the graphs G1 to nodes of G2 and vice versa. M(*s*) is a subset of M and the transition between two states *s*1 and *s*2 corresponds to the addition of a new pair of matched nodes Fig. 4. For each intermediate state *s*, the algorithm considers the set P(*s*) of the node pairs that can be added to the current state. A set of feasibility rules is introduced on the VF2 algorithm to verify the consistency conditions, such that only consistent states can be generated.

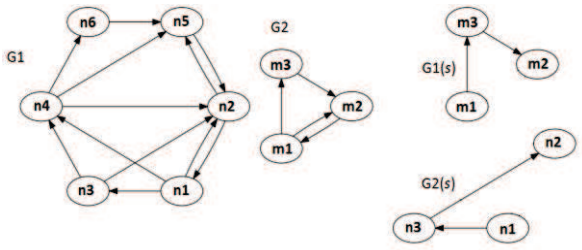


Figure 4. Two graphs and the partial graphs relative to the mapping function M(s)={(n1,m1),(n2,m2),(n3,m3)}.

A high-level description of the adapted VF2 algorithm is outlined in Fig. 5. In the initial state s_0, the mapping function does not contain any components. For each intermediate state *s*, the algorithm computes the set *P(s)* of the node pairs that are candidate to be added to the current state *s*. For each pair *p* belonging to *P(s)*, the feasibility rules are evaluated; if they succeed, being *p=(n,m),* the successor state *s'=s* ∪ *p* is computed and the whole process is recursively applied to *s'*. Notice that the algorithm explores the search graph in the SSR according to a depth-first search strategy.

Procedure Match(*s*)
  Input: an intermediate state *s*; the initial state s_0 has $M(s_0) = \emptyset$
  Output= the mappings between the two graphs.

  If *M(s)* covers all the nodes of *G2* **THEN**
    OUTPUT *M(s)*
  ELSE
    Compute the set *P(s)* of the pairs candidate for inclusion in *M(s)*
    FOREACH *(n, m) in P(s)*
      IF *F(s, n, m)* **THEN**
        Compute the state s' obtained by adding *(n, m)* to *M(s)*
        CALL Match(*s'*)
      END IF
    END FOREACH
    Restore the data structure
  END IF
END PROCEDURE

Figure 5. A high-level description of the VF2 algorithm from [12].

As example, the dotted line in Fig. 2(b) shows the subgraphs found using Fig. 2(b)'s graph as the main graph and the pull-up resistor graph in Fig. 3 as the pattern graph.

D. *Generating the new Netlist*

For each matched subcircuit pattern, adaptation rules are set such as deleting nets/components and adding nets. These rules adapt the main user netlist to the WaferBoard™ requirement. Here are some examples:

- Pull-up/Down resistor pattern: the resistor is going to be deleted and the NanoPad in contact with the uIC's pin will be configured as 1/0.
- Oscillator circuit pattern: the entire component defining this circuit will be deleted and the uIC will receive an external clock signal from the WaferBoard™.
- Connectors: the connector will be deleted and replaced by I/O ports.
- Decoupling Capacitor: the capacitor will be removed and the uIC VDD and ground pins will be powered by the NanoPad configured accordingly [16].

III. EXPERIMENTAL RESULTS

The Netlist Pruning Tool was created under the Qt Creator platform [17] and written in C++ with the Qt standard library objects. This section presents a set of experiments that shows the operation and performance of the NPT over a variety of generated netlists. Those netlist were generated by combining real netlists provided by the computer aided design AltiumDesigner [18]. All netlists for both main circuit and patterns were described in EDIF format. Experimental results are shown in (Table I). Recognition runtime is the time to find

out all the subcircuits. Total runtime includes setup time and subcircuit identification time.

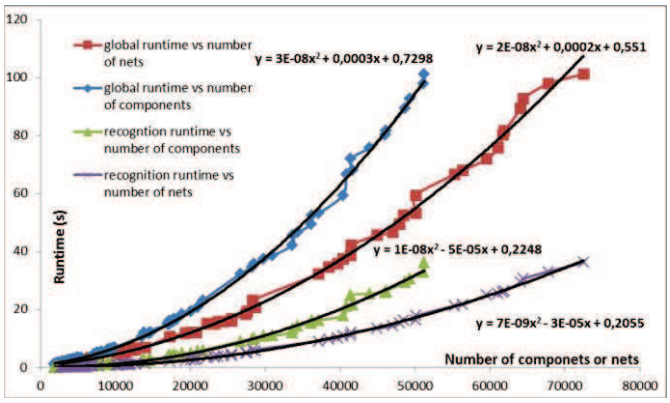


Figure 6. Runtime used for netlist pruning.

As expected, NPT precisely recognized all patterns, defined as pull-up/pull-down resistors and decoupling capacitors with all tested main netlists. Fig. 6 show the runtime of the tool by taking into account whether or not the time of setting up the main circuit and the patterns' circuit versus the number of components and the number of nets. For a fixed-sized patterns model, the matching time for increasingly larger sizes of main circuits shows a polynomial time performance. It's due to the time complexity of the VF2 algorithm which is $O(n^2)$[13]. The big increase in the runtime with the main circuit size is mainly due to the complexity of setting up the main circuit by collecting the connection information between the components from the EDIF file. Like it's shown in (Table I), It was measured for the 51233 component benchmark that 64.36% of the runtime is to set up the main circuit.

TABLE I. NPT RUNTIME FOR SEVERAL MAIN NETLIST SIZES.

Number of component in the main circuit	Number of nets in the main circuit	Recognition runtime in seconds (setup time excluded)	Runtime in seconds
51233	72467	36.102	101.32
41638	56389	21.642	67.935
30972	41363	10.916	38.473
20109	27587	4.801	19.187
13507	17267	2.107	10.54
9097	11791	0.952	5.867
7547	10913	0.811	4.886
2014	2662	0.06	0.976
795	1016	0.022	0.508

IV. CONCLUSION

This paper proposed an Object Oriented Framework tool for pruning electronic system netlists. The tool parses netlists, applies an adapted VF2 algorithm to detect the subcircuit patterns, and for each detected one applies specific modifications to the circuit. The experiments were performed using real circuits and the results show that the process is made in 101.32 seconds in the case of a circuit with 51233 components and 72467 nets. The tool has quadratic time performance and has a very fast runtime when it's used in real cases. It has been successfully integrated into the WaferConnect™ tool suite [19] for the WaferBoard™ rapid prototyping platform for electronic systems. Further research is being made to characterize the tool according to netlist/subcircuit size and to increase the number of subcircuits in the subgraph pattern library.

ACKNOWLEDGMENT

The authors thank the MITACS and Gestion TechnoCap Inc. for their financial support and CMC Microsystems for financial support and providing design tools, technical support and access to technologies, as well as Le Regroupement Stratégique en Microsystèmes du Québec (ReSMIQ) for collaboration.

REFERENCES

[1] Pelz, G.; Roettcher, U.; , "Pattern matching and refinement hybrid approach to circuit comparison," Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on , vol.13, no.2, pp.264-276, Feb 1994

[2] Conn, A.R.; Elfadel, I.M.; Molzen, W.W., Jr.; O'Brien, P.R.; Strenski, P.N.; Visweswariah, C.; Whan, C.B.; , "Gradient-based optimization of custom circuits using a static-timing formulation," Design Automation Conference, 1999. Proceedings. 36th , vol., no., pp.452-459, 1999

[3] Ohlrich, M.; Ebeling, C.; Ginting, E.; Sather, L.; , "SubGemini: Identifying SubCircuits using a Fast Subgraph Isomorphism Algorithm," Design Automation, 1993. 30th Conference on , vol., no., pp. 31- 37, 14-18 June 1993

[4] Chia Shin Ou, R. C. T. Lee; , "Solving the Subcircuit Matching by Using Encoded Matrix Method" The 24th Workshop on Combinitorial Mathematics and Comptation Theory, National Chi Nan University, Puli, Nantou, Taiwan, pp. 355-361, April 27–28, 2007.

[5] Wei-Hsin Chang; Shuenn-Der Tzeng; Chen-Yi Lee; , "A novel subcircuit extraction algorithm by recursive identification scheme," Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on , vol.5, no., pp.491-494 vol. 5, 2001.

[6] E. K. Wong; , "Three-dimensional object recognition by attributed graphs", Syntactic and Structural Pattern Recognition Theory and Applications, World Scientific, Singapore, pp. 381-414, Jan 1990.

[7] Si Wei Lu, Ying Ren, Ching Y. Suen, Hierarchical attributed graph representation and recognition of handwritten chinese characters, Pattern Recognition, Volume 24, Issue 7, Pages 617-632, 1991.

[8] Ming Qiu; Haibin Hu; Qingshan Jiang; Hailong Hu; , "A New Approach of Graph Isomorphism Detection Based on Decision Tree," Education Technology and Computer Science (ETCS), 2010 Second International Workshop on , vol.2, no., pp.32-35, 6-7 March 2010.

[9] B.T. Messmer, H. Bunke, A decision tree approach to graph and subgraph isomorphism detection, Pattern Recognition, Volume 32, Issue 12, Pages 1979-1998, December 1999

[10] Cordella, L.P.; Foggia, P.; Sansone, C.; Vento, M.; , "An efficient algorithm for the inexact matching of ARG graphs using a contextual

transformational model," Pattern Recognition, 1996., Proceedings of the 13th International Conference on , vol.3, no., pp.180-184 vol.3, 25-29 Aug 1996

[11] Cordella, L.P.; Foggia, P.; Sansone, C.; Vento, M.; , "Performance evaluation of the VF graph matching algorithm," Image Analysis and Processing, 1999. Proceedings. International Conference on , vol., no., pp.1172-1177, 1999.

[12] P. Foggia, C. Sansone, M. Vento. "An Improved Algorithm for Matching Large Graphs," the 3rd IAPR-TC15 Workshop on Graph-based Representations, Ischia, 2001.

[13] Cordella, L.P.; Foggia, P.; Sansone, C.; Vento, M.; , "A (sub)graph isomorphism algorithm for matching large graphs," Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.26, no.10, pp.1367-1372, Oct. 2004.

[14] P. Foggia, C. Sansone, M. Vento, "A performance comparison of five algorithms for graph isomorphism", submitted for publication to the Third TC-15International Workshop on Graph Based Representation, Italy, 2001.

[15] Norman, R.; Valorge, O.; Blaquiere, Y.; Lepercq, E.; Basile-Bellavance, Y.; El-Alaoui, Y.; Prytula, R.; Savaria, Y.; , "An active reconfigurable circuit board," Circuits and Systems and TAISA Conference, 2008. NEWCAS-TAISA 2008. 2008 Joint 6th International IEEE Northeast Workshop on , vol., no., pp.351-354, 22-25 June 2008

[16] A Dual-Power Rail, Low-Dropout, Fast-Response Linear Regulator Laflamme-Mayer, IEEE NEWCAS Conference, Bordeaux, France, Jun 26-29, 2011

[17] Qt Creator and Library, http://qt.nokia.com, mai 2011.

[18] http://www.altium.com, mai 2011.

[19] Lepercq, E.; Blaquiere, Y.; Norman, R.; Savaria, Y.; , "Workflow for an electronic configurable prototyping system," Circuits and Systems, 2009. ISCAS 2009. IEEE International Symposium on , vol., no., pp.2005-2008, 24-27 May 2009.