UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MICROELETRÔNICA

VINÍCIUS DAL BEM

SAT-Based Environment for Logical Capacity Evaluation of Via-Configurable Block Templates

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor in Microelectronics.

Advisor: Prof. Dr. Renato Perez Ribas
Co-advisor: Prof. Dr. André Inácio Reis

Porto Alegre
2016

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Dal Bem, Vinícius

SAT-Based Environment for Logical Capacity Evaluation of Via-Configurable Block Templates / Vinícius Dal Bem. – 2016.

95 f.:il.

Orientador: Renato Perez Ribas; Co-orientador: André Inácio Reis.

Tese (Doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Microeletrônica. Porto Alegre, BR – RS, 2015.

Manufacturing yield. Nanometer technology. Photolithography. Structured ASIC. Regular layout. Via-configurable. CAD. Computer Aided Design. SAT. Satisfiability. Block templates. Logical capacity

1.Manufacturing yield. 2.Nanometer technologies. 3. Photolithography. 4. Structured ASIC. 5.Regular layout. 6. Via-configurable. 7.CAD. 8.Computer Aided Design. 9.SAT. 10. Satisfiability. 11. Block templates. 12. Logical capacity. I. Ribas, Renato Perez. II. Reis, André Inácio. III. SAT-Based Environment for Logical Capacity Evaluation of Via-Configurable Block Templates.

ACKNOWLEDGEMENTS

I hope the readers don’t mind, but I’ll allow myself writing the acknowledgements in Portuguese so my gratitude will be better transmitted.

Dedico esta tese à minha esposa, Daniela, que me apoiou infinitamente, que nos momentos mais difíceis transformou cansaço e angústia em sorrisos, e que fez com que um futuro vazio se transformasse na visão de uma vida feliz. Obrigado, amo você!

Agradeço à minha família, pelo carinho, pelo suporte emocional e financeiro, e pelo exemplo de dedicação ao trabalho.

Agradeço à Renato Perez Ribas e Paulo Francisco Butzen, por serem referências para mim, pessoais e profissionais, durante toda essa jornada acadêmica.

Agradeço aos meus amigos, por compartilharem sua genialidade no meio profissional e pelos momentos de alegria fora dele.

ABSTRACT

Structured ASICs with regular layouts comprise a design-based solution for IC manufacturing yield loss in nanometer technologies caused by photolithography distortions. Via-configurable structured ASICs is even a more restrictive digital IC design method, based on the repetition of a block template comprising all layout layers except the vias one. The choice of such a design strategy impacts greatly the final circuit characteristics, arising the need for specific CAD tools to allow template evaluation and comparison in different aspects. This work presents a SAT-based CAD environment for evaluating the logical capacity aspect of via-configurable block templates. The proposed environment is able to support any user-defined template, and behaves efficiently when applied to block templates presented in related literature.

Keywords: Manufacturing yield. Nanometer technology. Photolithography. Structured ASIC. Regular layout. Via-configurable. CAD. Computer-aided design. SAT. Satisfiability. Block templates. Logical capacity. Digital IC design.

Ambiente SAT para Avaliação da Capacidade Lógica de Padrões de Bloco Configuráveis por Vias

RESUMO

ASICs estruturados com leiautes regulares representam uma das soluções para a perda de rendimento de fabricação de circuitos integrados em tecnologias nanométricas causada pela distorção de fotolitografia. Um método de projeto de circuitos integrados ainda mais restritivo resulta em ASICs estruturados configuráveis apenas pelas camadas de vias, que são compostos pela repetição do mesmo modelo de bloco em todas as camadas do leiaute, exceto as camadas de vias. A escolha do modelo de bloco tem grande influência nas características do circuito final, criando a demanda por novas ferramentas de CAD que possam avaliar e comparar tais modelos em seus diversos aspectos. Esta tese descreve um ambiente de CAD baseado em SAT, capaz de avaliar o aspecto de capacidade lógica em padrões de blocos configuráveis por vias. O ambiente proposto é genérico, podendo tratar quaisquer padrões de bloco definido pelo usuário, e se comporta de maneira eficiente quando aplicado aos principais padrões já publicados na literatura.

Palavras-chave: Rendimento de fabricação. Tecnologias nanométricas. Fotolitografia. ASICs estruturados. Leiaute regular. CAD. SAT. Modelos de bloco. Capacidade lógica. Circuitos integrados digitais.

FIGURES LIST

TABLES LIST

LIST OF ABREVIATIONS

2D	Two-Dimensional
ASIC	Application Specific Integrated Circuit
BDD	Binary Decision Diagram
CAD	Computer Aided Design
CMOS	Complementary Metal-Oxide-Semiconductor
CNF	Conjunctive Normal Format
CSC	Complete State Coding
DFM	Design for Manufacturing
EPE	Edge Placement Error
ESD	Electrostatic Discharge
FPGA	Field-Programmable Gate Array
GND	Ground Supply
HCI	Hot-Carriers Injection
IC	Integrated Circuit
INVA	Inverters Array
NBTI	Negative Bias Temperature Instability
OPC	Optical Proximity Correction
PD	Pull-Down
PDK	Process Design Kit
PU	Pull-Up
PLA	Programmable Logic Array
RAM	Random Access Memory
RET	Resolution Enhancement Techniques
SAT	Satisfiability
SEE	Single Event Effects
SOC	System-on-Chip
STG	Signal Transistion Graphs
TDDB	Time Dependent Dielectric Breakdown
TRL	Transistors Regular Layout
TTSPM	Two-terminal Series-Parallel Multigraph)
VCC	Via-Configurable Cell
VCGA	Via-Configurable Gate Array
VCLB	Via-Configurable Logic Block
VCSA	Via-Configurable Structured ASICs
VCTA	Via-Configurable Transistors Array
VDD	Power supply

SUMMARY

1 INTRODUCTION

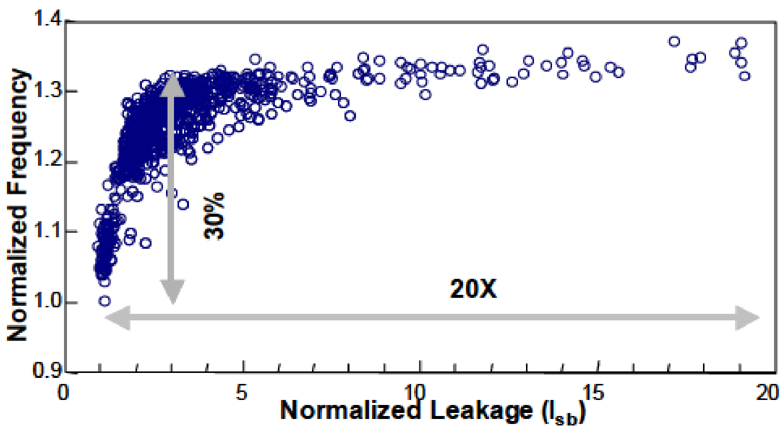
In the last decades, the microelectronics industry advancing has been strongly based on the dimensions reduction of manufactured semiconductor devices. However, as CMOS technology shrinks, a series of previously ignored effects has become significant and new manufacturing and design challenges arise.

Negative-bias temperature instability (NBTI) can be mentioned as a transistors aging effect that became more aggressive in most recent technologies, being treated in massive research work (VATTIKONDA, 2006; BUTZEN, 2010; BUTZEN, 2012).

Among other effects that presenting similar characteristics are single-event effects (SEE), electrostatic discharge (ESD), time-dependent dielectric breakdown (TDDB), hot-carriers injection (HCI), IR-drop, electromigration and, as the main subject of this work: manufacturing variability (BOKAR, 2003; ORSHANSKY, 2008; WONG, 2008)

Manufacturing variability impacts directly the production yield as it may change some die characteristics to beyond the specifications, as illustrated in Figure 1.1. Even when the manufacturing variation is not critical enough to cause pre-sale rejection of dies, it may reduce their operation margins, automatically limiting their lifetime against the mentioned aging effects.

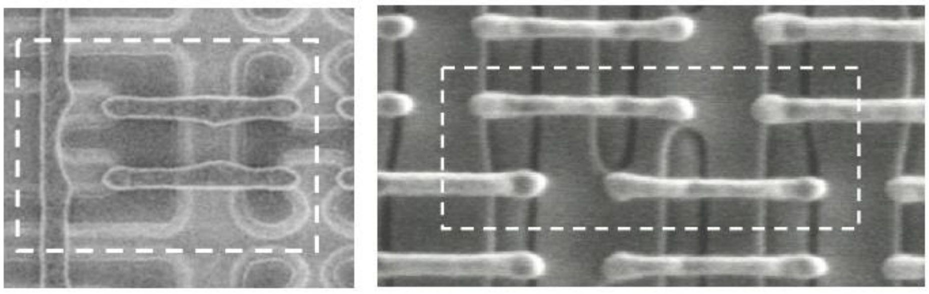
Figure 1.1 - Manufacturing variability changing the electrical characteristics of dies, in this case, frequency and standby leakage current of microprocessors in a wafer.



Source: (BOKAR, 2003)

The manufacturing variability in nanometer technologies is caused by a variety of sources, and can be classified as random, like dopants distribution, and as systematic, like photolithography distortions. This thesis focuses on the last one, illustrated in Figure 1.2.

Figure 1.2 - Discrepancy between drawn layout (assuming perfect rectangle shapes) and manufactured circuit layers by photolithography process.



Source: (KUHN, 2007)

Even more specifically, this work focuses on a single design strategy for reducing the photolithography distortion that is the use of via-configurable regular layouts. Such layouts comprise a small set off patterns, easing the tuning of the manufacturing process.

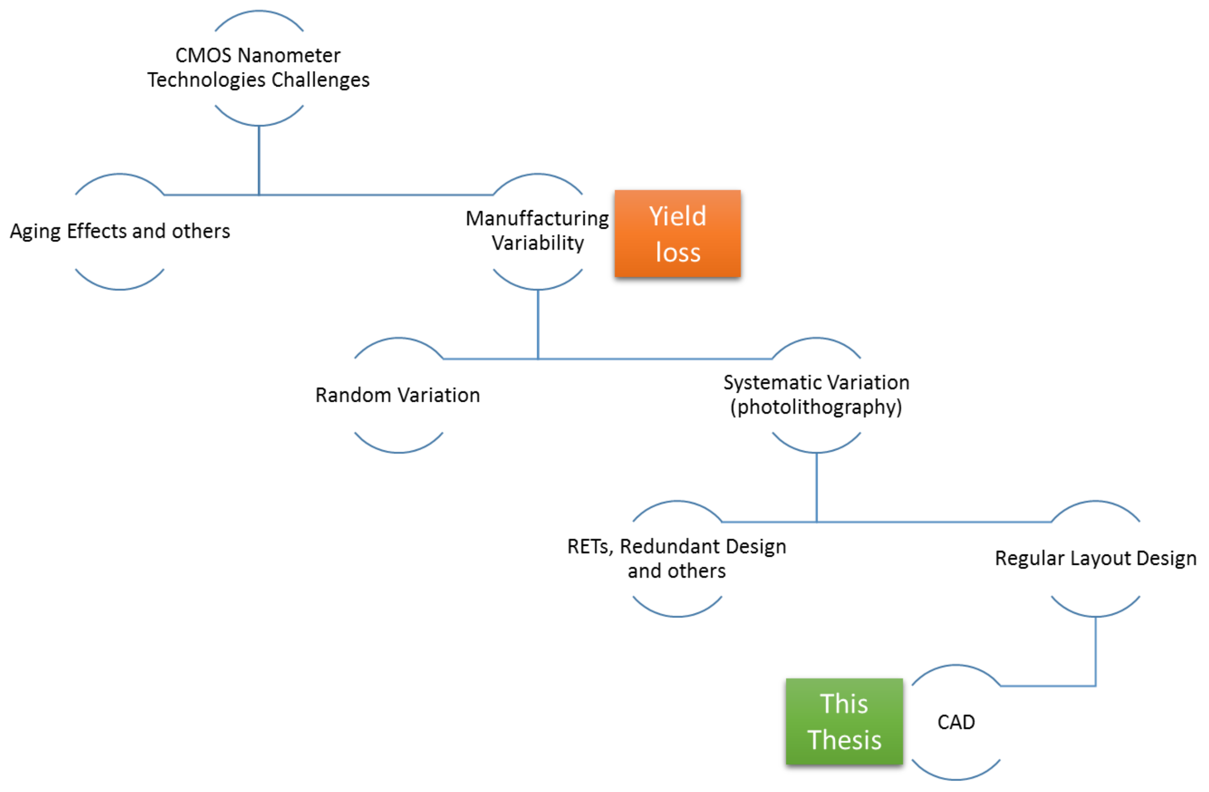
Alternative solutions to photolithography distortion problems include resolution enhancement techniques (RETs) as optical proximity correction (OPC) (CHIANG, 2007). OPC is an automatic layout changing procedure which aims to increase the similarity between the (originally) drawn layout and the produced die. Techniques for increasing design robustness, like inserting logic redundancy or increasing operation margin, do not reduce the suffered variation but mitigates its impact. However, both mentioned solutions do not bring the regular layouts benefits: predictable layout neighboring influence and, in cases where some layers are made by a repeated standard, the reduced production cost and fasten time-to-market. Moreover, all mentioned solutions may be combined.

The main goal of this thesis is to present a novel CAD environment to solve feasibility of a transistors network on a pre-defined layout template. Such CAD environment aims to increase efficiency of design methods based on via-configurable regular layouts, contributing to reduce photolithography distortion, so reducing manufacturing variability and increasing production yield. Figure 1.3 summarizes this proposal and its relationship to the stated problems and alternative solutions.

More specifically, the main contribution of the proposed CAD tool is to be able to answer if a given logic transistors network can be expressed in a given VCSA (via-configurable structured ASICs) layout template.

The main open question investigated herein is whether there is an automated, efficient and exact (not heuristic) method based on Boolean formulation to reach such answer.

Figure 1.3 - Scheme connecting this thesis to the introduced scenario of yield loss and one of its main sources: photolithography process distortions.


CMOS Nanometer Technologies Challenges
Aging Effects and others
Manuffacturing Variability
Yield loss
Random Variation
Systematic Variation (photolithography)
RETs, Redundant Design and others
Regular Layout Design
This Thesis
CAD

Once a general mechanism to reach this answer is built, the logical capacity of any VCSA layout template can be evaluated by repeating the process for several transistor networks. Choosing the VCSA layout template with higher logical capacity is expected to cause a reduction in the final circuit area and, consequently, amortizing manufacturing yield loss.

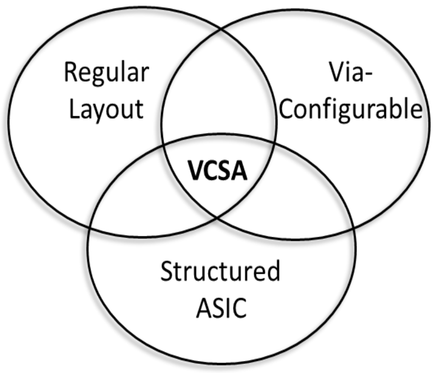
Until the present date, no other author has published a tool or method able to accomplish this goal for any transistor network and any layout template. The tools in related literature are always made for a specific layout template as will be detailed later on this text.

The remaining of this text is organized as follows: Chapter 2 provides the needed background in Via-Configurable Structured ASICs (VCSAs) and the main proposed block templates in related literature; Chapter 3 contains an introduction about computer-aided design (CAD) problems translated to satisfiability (SAT) models; Chapter 4 describes the developed SAT model for extracting the logical capacity of VCSA block templates; Chapter 5 brings achieved results, whereas Chapter 6 draws conclusions and insights about future work.

2 VIA-CONFIGURABLE STRUCTURED ASICS (VCSA)

The target design method or layout strategy may intersect to various labels. “Structured ASIC” is often used to refer layouts made by repeating a block template, as in (TSAI, 2012) and in (TUNG, 2012). This expression may be used when the layout is configurable by via layers, as mentioned in the references. However, it is not always restricted to this characteristic. “Regular fabrics” or “regular layout” are also labels used for this kind of design method, as in (RAN, 2004a; RAN, 2004b). Moreover, they are too general, being used in regular cells design methods, as observed in (JHAVERI, 2010), and even to refer some routing strategies (SUTO, 2012). Thus, to avoid misunderstandings, from now on, the default label to refer the studied design method will be the very specific term “Via-Configurable Structured ASICs” (VCSA). This way, it becomes clear that we are focusing on regular layouts (or fabrics) composed by some standard blocks repetition (structured ASIC). However, the only non-repetitive layer used for the circuit functionality configuration are the via layers. Figure 2.4 highlights of the chosen design method and its resulting layout class.

Figure 2.4 - Relationship among regular layouts, via-configurable layouts, structured ASICs and via-configurable structured ASICS (VCSA).



Among the reasons to explore VCSAs are: the reduced production and mainly re-design cost, due to the reduced set of configurable masks; the reduction on layout design costs, since a single template is used for all digital cells; the extremely predictable layout neighboring influence, reducing uncertainty and variations on manufactured circuit, as well as increasing RETs adherence.

Mask-configurable gate arrays (MPGAs) do not represent a new design approach, as demonstrated by Van Noije’s proposal (VAN NOIJE, 1985), and several following optimizations, as Quick Customized Logic (QCL) (DONG, 1993) and Maragata (LIMA, 2000), all customizable by metal layers. However, such previous work did not focus on lithography optimization or manufacturing variability mitigation, but in reducing costs through pre-manufactured layers (possibly shared among several customers) and faster design or prototyping cycles. Therefore, even fitting the concept of mask-programmable gate arrays, a VCSA differ fundamentally from these past approaches as it presents pre-designed layers instead of pre-manufactured ones, and also the concern of a litho-friendly regular layout, even more restrictive than older MPGAs. Other modern regular mask-programmable design approaches, as in (BOBBA, 2015), also differ from VCSAs in the same point: not intended for lithography process, but for other manufacturing process candidates.

It is important to state clearly that the VCSA design method is not proposed herein, it is just a naming convention adopted in this text and absent in related literature using this design method, as each paper seems to adopt a different term.

2.1 VCSA Block Templates

VCSAs are built by placing and connecting blocks as much as digital circuits using standard cell design method are built by placing and connecting cells from a library. However, in VCSAs, it is not possible to observe any difference between the blocks if the via layers are not observed. The set of layers that present identical layout patterns for all VCSA blocks comprises the "block template". Figure 2.5 demonstrates a VCSA block template, and how different functionalities are configured in this template only by different vias placement on them.

Figure 2.5 - Block template example (a), and the vias configuration in this template example to implement an INV (b) and a NOR3 (c) logical functions.

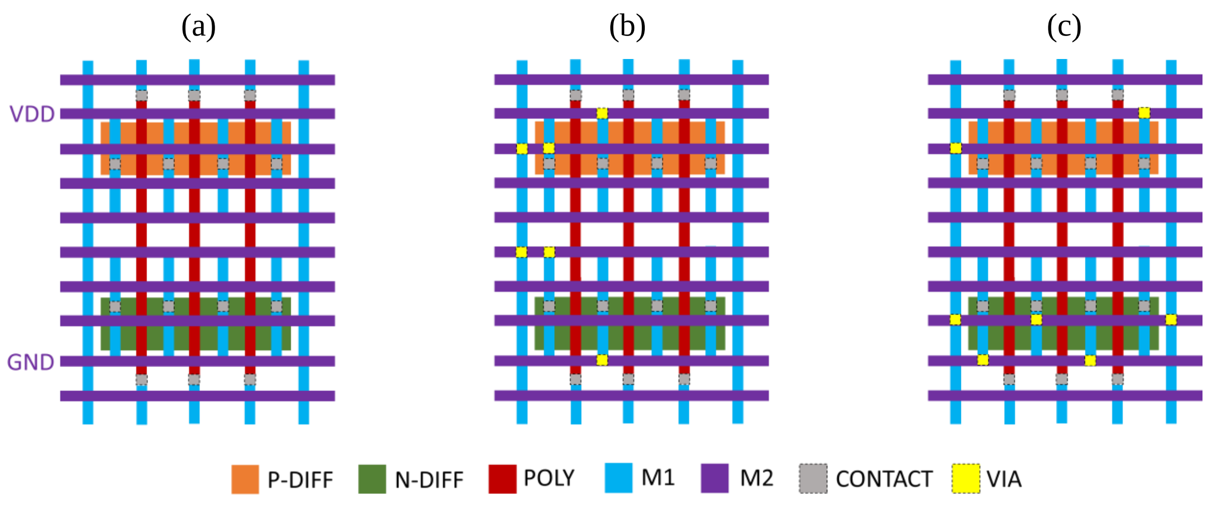
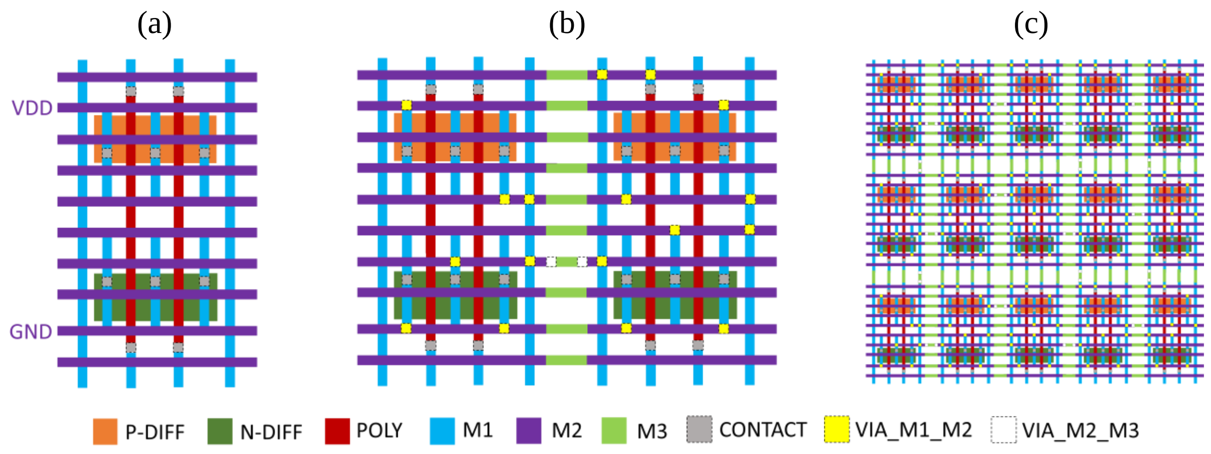


Figure 2.6 depicts another block template. Observe that this template is smaller in terms of resources as the available transistors and metal lines. One single instance of such block is not able to express a logical function as the "NOR3" depicted in the previous template example. However, it can be achieved by correctly configuring and connecting two block instances, as shows Figure 2.6(b). Ultimately, any logic function or digital circuit can be expressed if the number of block instances is high enough, as long as the template is able to express a basic set of logic functions and global routing resources are available.

In this document, the set of realizable logic functions into a single instance of a block template will be labeled in this text as the template "logical capacity", and not as "functional expression capability" or "function implementation capability" or "configurability capability" as in (RAN, 2006a) and in (LI, 2008), even they all presenting the same meaning.

Figure 2.6 - Block template example (a), a NOR3 logical function implemented by configuring two block instances (b) and a circuit designed by VCSAs combination (c).



Logical capacity is not the only attribute that may be evaluated in a VCSA block template. The template area itself is an attribute of interest, and is intuitively correlated to logical capacity: among two templates requiring similar area, the one presenting higher logical capacity is a better choice as will probably result in a smaller final circuit. However, the expected scenario is that templates with higher logical capacity present larger areas. The transistor sizing strategy also may differ among block templates, as can be observed in the previous and subsequent examples, and impacts electrical performance of the final circuit. Even the chosen layout patterns to compound the block template are important on its evaluation, as some patterns (as "L" shapes) will cause higher distortions during the photolithography processes, increasing the discrepancy between drawn layout and manufactured circuit, and decreasing manufacturing yield. This last described attribute is referred as how much “litho-friendly” a block template is or if it is “lithography-aware”.

The mentioned set of block template attributes highlights that, even when working with an extremely regular class of layouts, still there is a wide range of design choices. Therefore, several different block template proposals arise in related literature, and the main ones are discussed in the next subsection.

2.2 VCSA Block Templates in Literature

From now on, all explanations about layouts assume the conventional CMOS technology, p-type and n-type diffusions, transistor gate formed by polysilicon (or simply “poly”), and connections by metal lines (in which “M1” or “metal1” means the metal level closer to diffusions). However, all discussion and models herein are still valid if any of these materials are replaced by the ones in newer technologies.

2.2.1 VCC

“Via-configurable cell” (VCC) is a VCSA block template proposed by Y. Ran *et al.* in (RAN, 2004a). VCC related work progresses in (RAN, 2004b; RAN, 2004c), culminating in (RAN, 2006a) and in (RAN, 2006b). Figure 2.7 depicts a VCC template or, more specifically, the 5-VCC, which is the VCC variation containing 5 transistors in each plane. Figure 2.7 also shows another template called “inverters array” (INVA), which serves as auxiliary logic to VCC. Both VCC and INVA are combined in the proposed placement and routing scheme compose a larger block template named VCGA (via-configurable gate array), shown in Figure 2.8.

Observe in Figure 2.7 that every crossing between M1 and M2 is a potential place for a via, while all contacts are already placed. The global routing strategy is detailed in (RAN, 2006b) and makes the entire circuit configurable only through the via layers customization.

The main characteristics of the VCC template are:

- Poly matching between p-type and n-type transistors.
- No diffusion breaks
- Pre-defined metal lines for supply (VDD and GND)
- Only two metal lines able to connect both p and n planes

The main differences of the INVA template to the VCC one are:

- Present some diffusion breaks
- Diffusion breaks aligned in both p and n planes
- Several metal lines able to connect both p and n planes

Figure 2.7 - 5-VCC block template (a) and INVA block template (b).

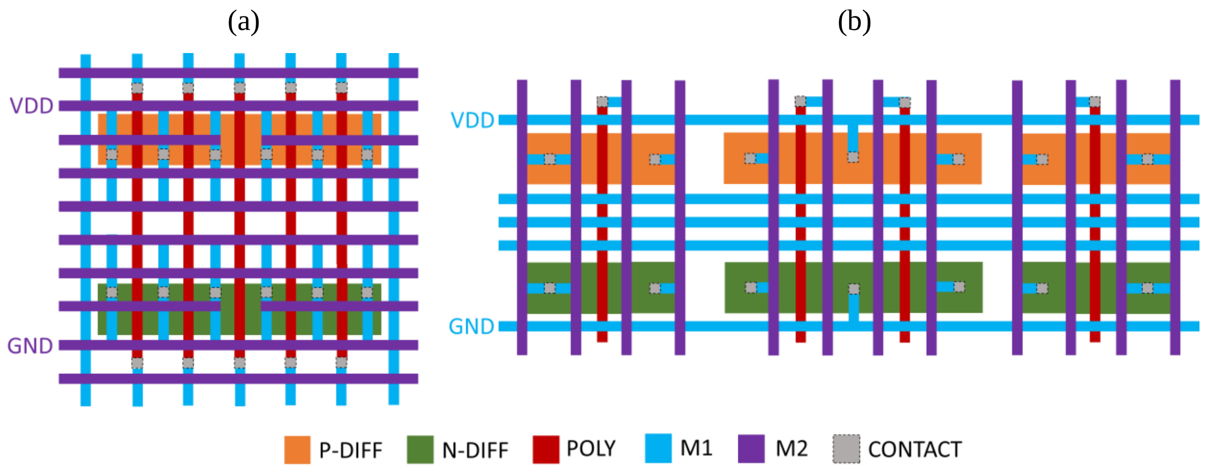
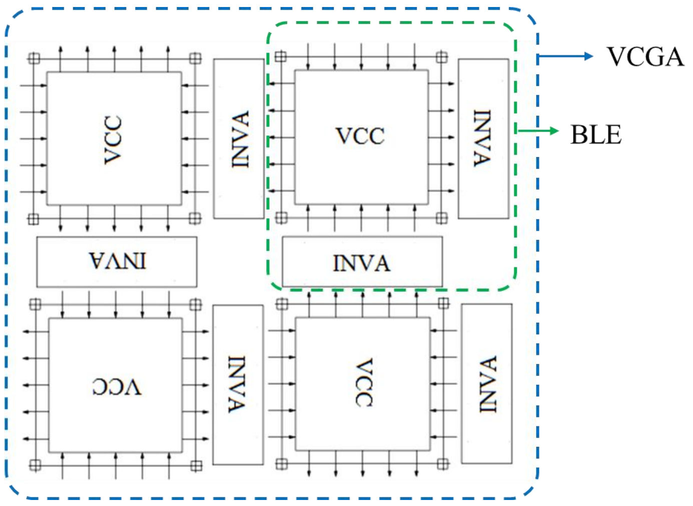


Figure 2.8 - VCGA block built by the association of VCC and INVA blocks.

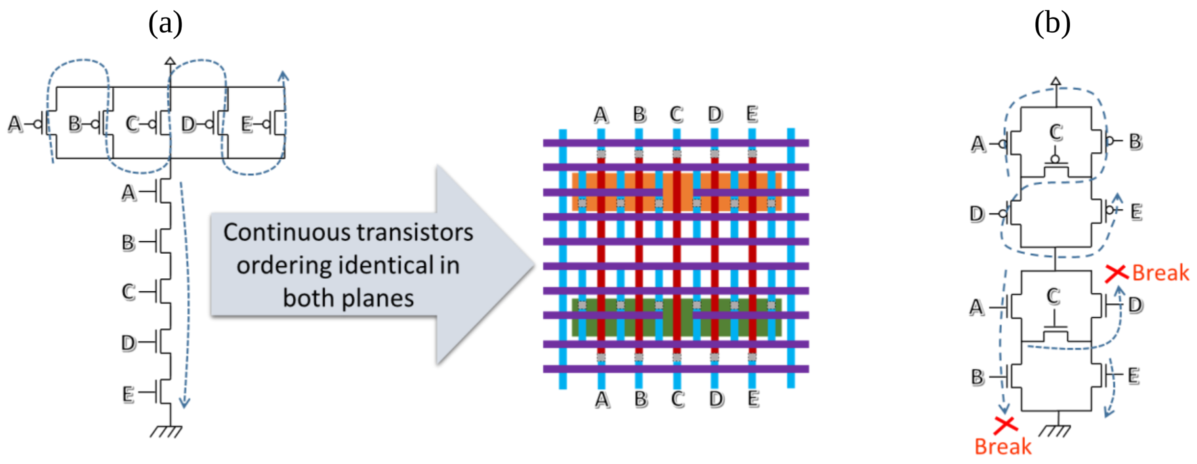


Source: (RAN, 2006)

These templates present connections possibilities restricted by the poly matching and the diffusion abutments. However, these characteristics are very efficient in optimizing the template area. In VCC case, there is also no diffusion breaks, meaning that any transistor network expressed in this template must present the same control ordering in both planes, forming a continuous path through all transistors.

Figure 2.9 illustrates transistor networks with and without such characteristic, and their relationship to the VCC template.

Figure 2.9 - Example of transistor networks that can (a) and cannot (b) be expressed in a 5-VCC.



If two transistor networks present Eulerian paths starting from supply nodes, they can be connected at this node, creating a single continuous path. This fact, together with the presence of only two metal lines in VCC capable of connecting p and n planes, induces the conclusion that this template is intentionally designed to support at most two simultaneous logic functions (or transistor networks).

2.2.2 VCTA

"Via-configurable transistors array" (VCTA) is a VCSA block template proposed by M. Pons *et al.* in (PONS, 2010). Another study about the same template is presented in (PONS, 2011). Figure 2.10 depicts a VCTA template or, more specifically, the VCTA-6T: the VCTA variation comprising 6 functional transistors and 2 dummy transistors in each plane. Observe in Figure 2.10 that every crossing between M1 and M2, or M2 and M3, is a potential place for a via, while each M1 above diffusions is a potential place for a contact. The proposed placement and routing scheme for this template, shown in Figure 2.11, relies exclusively on VCTA blocks with alternated orientation. It is possible to observe in Figure 2.11 that the global routing strategy requires customization of metal layers, even with pre-defined fixed metal layers pattern inside the VCTA blocks.

The main characteristics of the VCTA template are:

- Independent poly stripes in p-type and n-type transistors;
- No diffusion breaks;
- Pre-defined metal lines for supply (VDD and GND);
- Several metal lines able to connect both p and n planes;

- High maximum number of vias;
- Contacts layer is not pre-defined (both contacts and vias configurable).

Figure 2.10 - VCTA-6T block template.

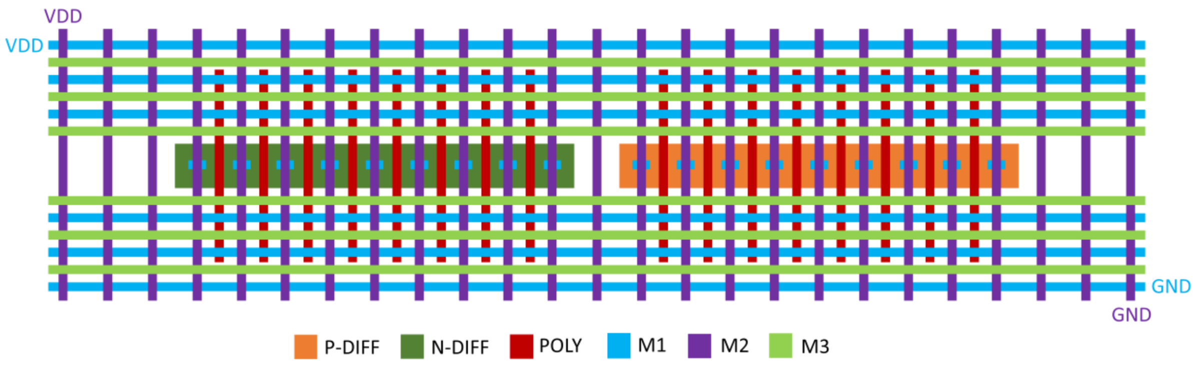
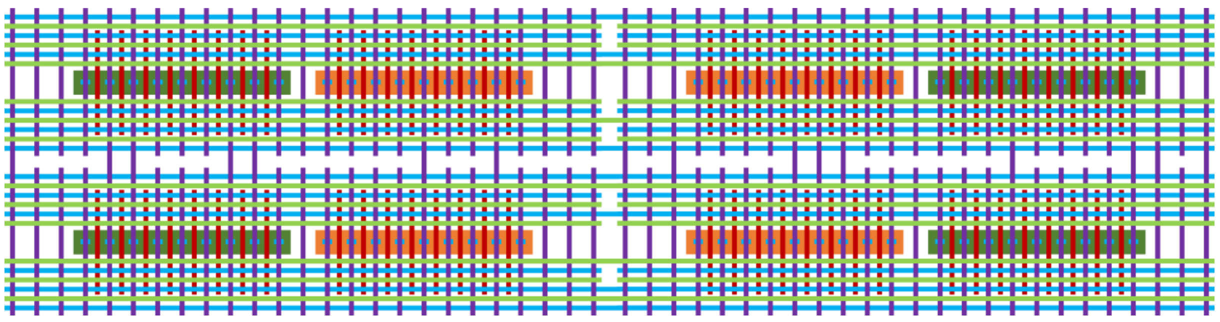


Figure 2.11 - VCTA design method for blocks placement (with mirrored diffusions) and global routing (with metal line extensions).



VCTA presents higher flexibility for transistor network configuration, as the transistor gates at the template layout are not matched in both p and n planes. This flexibility impacts in template area. However, area does not seem to be the focus of this template anyway, as this is the only template to add dummy poly stripes for making lithography process even more predictable. Another very particular characteristic of this template is the need of modifying the most of layout layers in a circuit re-design (increasing considerably the production cost) because not only vias but also contacts are used to configure the blocks logic functions, and global routing does not present a standard fabric, requiring changes in metal layers to build the connections.

2.2.3 VCLB

"Via-configurable logic block" (VCLB) is a not a single template but a set of VCSA block templates detailed by H. Tung *et al.* in (TUNG, 2012), as a result of several previous work exploring the subject (LI, 2008; TUNG, 2009; CHEN, 2010). Figure 2.12 depicts a VCLB template, or more specifically its most explored variation, the SLVC5P template. Observe in Figure 2.12 that every crossing between M1 and M2 is a potential place for a via. The proposed placement and routing scheme for this template, shown in

Figure 2.13, consists basically of side-by-side block instances, making the entire circuit configurable only through the via layers personalization.

Figure 2.12 - VCLB-SLVC5P block template.

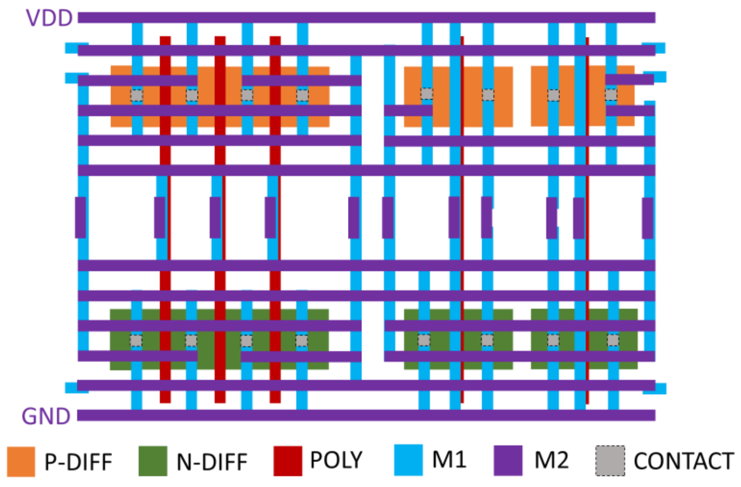
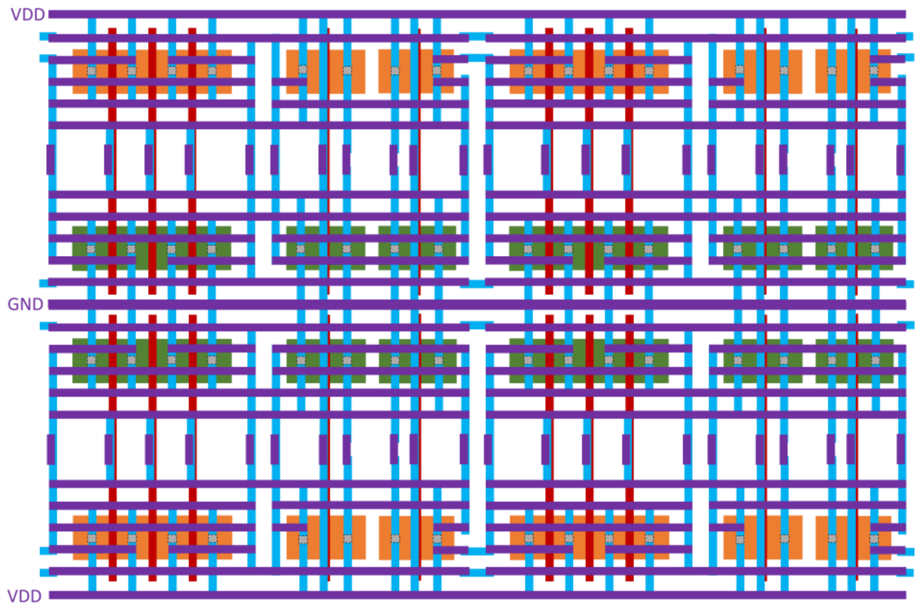


Figure 2.13 - VCLB design method for blocks placement and global routing.



The main characteristics of the VCLB template are:

- Poly matching between p-type and n-type transistors;

- Present some diffusion breaks;
- Diffusion breaks aligned in both p and n planes;
- Pre-defined metal lines for supply (VDD and GND);
- Several metal lines able to connect both p and n planes.

VCLBs are middle-term alternatives to VCC and VCTA. They are not as area-optimized as VCC and not as regular or lithography-aware as VCTA, but still configures circuit functionality only by the vias mask and present increased flexibility on expressing transistor networks due to the diffusion breaks and metal lines crossing both planes.

2.3 Evaluation Methods of VCSA Block Templates

All exposed block templates present a different set of characteristics that makes each one the most convenient choice in a particular scenario or for a certain set of target circuits. It brings up the need for evaluating and comparing methods to aid the choice of the most appropriate template to be used during VCSA-based layout design.

2.3.1 Lithography-based Evaluation

The analysis of how much a layout is lithography-aware can be either qualitative or quantitative.

On the basis of a qualitative analysis, there is a different expected behavior for each layout layer. Layout layers such as diffusions or higher metal levels are not totally prone to lithography errors that are more likely to occur in their relationship with other layers like contacts and vias. However, layers with smaller feature sizes, like polysilicon and M1, are usually considered more critical, since they are also candidates to errors as shorts and opens, and their electrical characteristics are more influenced by dimensions variation. Polysilicon and M1 masks of layouts based on the VCSA templates, addressed in previous section, are depicted in Figure 2.14 and in Figure 2.15

Figure 2.14 - Polysilicon masks based on VCTA (a), VCC (b), INVA (c), VCLB (d).

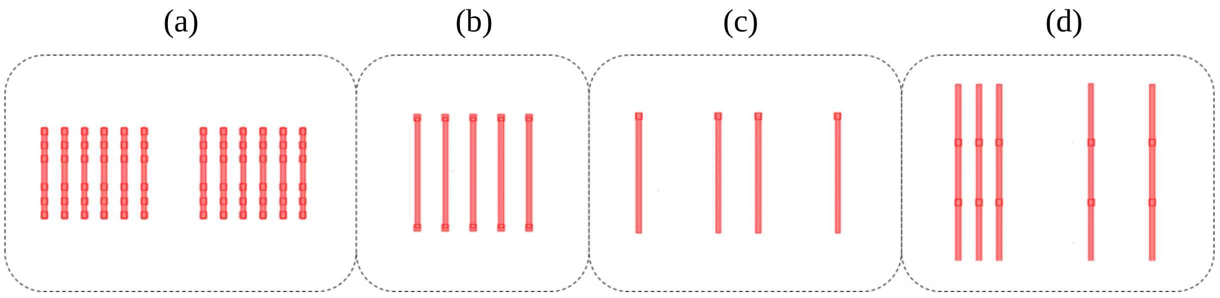
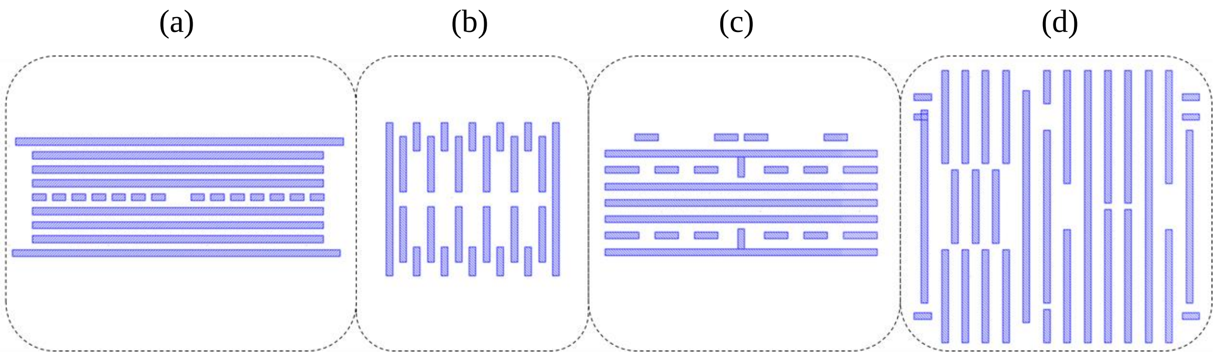


Figure 2.15 - M1 masks based on VCTA (a), VCC (b), INVA (c), VCLB (d).



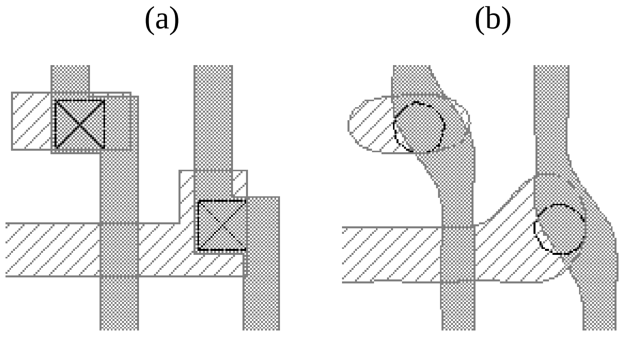
It is possible to analyze such masks according to two aspects, the layout patterns themselves and their neighboring, as detailed below.

a) Layout Patterns

As already stated, designing layouts with smaller amount of patterns, *i.e.*, increasing the circuit regularity, is expected to improve the efficiency of resolution enhancement techniques (RETs) and manufacturing yield. However, some patterns are more difficult to print using photolithography. Hence, two circuits with the same regularity degree can achieve different manufacturing yield, according to their particular layout pattern characteristics.

The rough idea is that the corners of shapes are quite hard to manufacture with high layout fidelity. It makes some patterns, like L-shapes, jogs in lines and end-of-lines, usually not effective in lithography process. Figure 2.16 illustrates such remarks. Thus, from the layout point-of-view, guidelines for lithography-aware design may include the use of unidirectional lines and avoid segmentation.

Figure 2.16 - Comparison between drawn layout (a) and lithography simulation (b) of pattern samples.



Source: (DAL BEM, 2012)

Notice that the lines in Figure 2.14(a) and in Figure 2.15(a) are unidirectional and the segmentation is restricted, whereas the patterns in the two next columns of these figures present some increasing in number of line breaks and two small jogs. Figure 2.14(d) and Figure 2.15(d), in turn, present many different patterns. From the previous statements, it is possible to consider the columns in Figure 2.14 and in Figure 2.15 ordered according to the lithography difficulty, being the pattern in Figure 2.14(a) and in Figure 2.15(a) the most effective to manufacturing process.

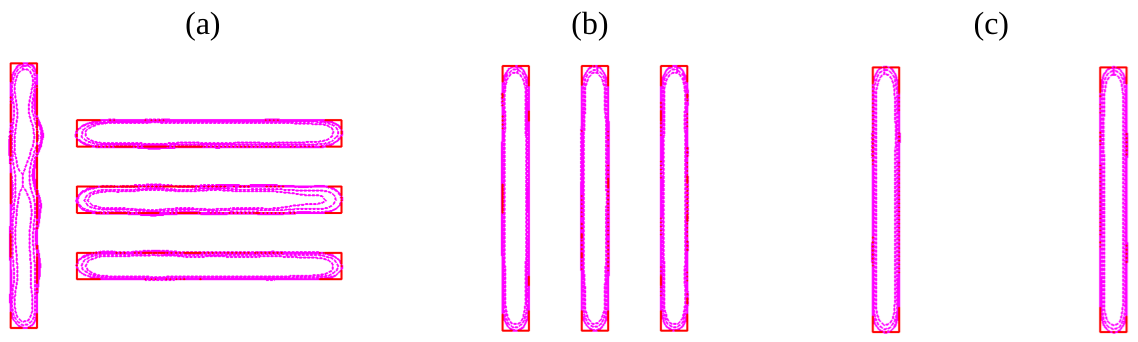
b) Neighboring

The quality of printed patterns does not depend only on their own shape, but also on the influence of near patterns. In lithography-aware designs, the intra-cell shapes and distances are carefully selected, such that neighboring impact is helpful instead of aggressive. Figure 2.17 (a) depicts aggressive neighboring lines influence, whereas Figure 2.17 (b) and Figure 2.17 (c) illustrate a negligible influence becoming a suitable one through the pitch adjustment.

Pitch adjustment and dummy lines insertion are design-for-manufacturing (DFM) techniques for controlling this impact that, again, shows the convenience of unidirectional lines. In the VCTA description (PONS, 2010), the authors are the only one to explicitly mention that the first and the last poly lines in each transistor netlist plane are intended to be dummy, with exclusive purpose of lithography improvement. Its enhanced regularity also simplifies the pitch adjustment. In VCC and INVA approaches, there is no explicit mention about pitch adjustment or dummy lines insertion. However, as all lines respect a common grid, these techniques would be easily applied. In VCLB approach, distances are not exact multiples, as can be seen in Figure 2.14(c) and in Figure 2.15(c). Thus, adjusting lines to a common pitch may imply significant area penalty.

The neighboring influence analysis must consider not only intra-cell shapes but also the inter-cell relationship, which is a very hard task when using conventional standard cell libraries, since there are many possible neighboring patterns. The use of identical transistor arrays, as in VCSA templates, makes such impact very predictable due to the similarity of neighboring layouts. However, it is not ensured that the intra-cell DFM characteristics automatically extend to global scenario. The VCGA architecture, presented in (RAN, 2006a), can be used as example. This architecture comprises VCC blocks surrounded by rotated INVA blocks, making possible aggressions like the one illustrated in Figure 2.17(a), which were not present on the individual analysis of these sub-blocks.

Figure 2.17 - Neighboring influence: aggressive (a), helpful (b) and negligible (c).



Source: (DAL BEM, 2012)

c) Lithography awareness quantification

There are several different methods of quantifying how much a design is lithography-aware. The fast-Fourier-transform 2D approach is not efficient when comparing layouts with similar lithographic behavior (JHAVERI, 2007). The FOCSI method measures regularity of circuits, but does not take into account critical patterns as described in the previous qualitative analysis (PONS, 2012). Edge placement error (EPE), in turn, is a very simple concept, which basically exposes the difference of printed edges when comparing to the original layout (MITRA, 2005). This concept is used for the measurements exposed later in this section.

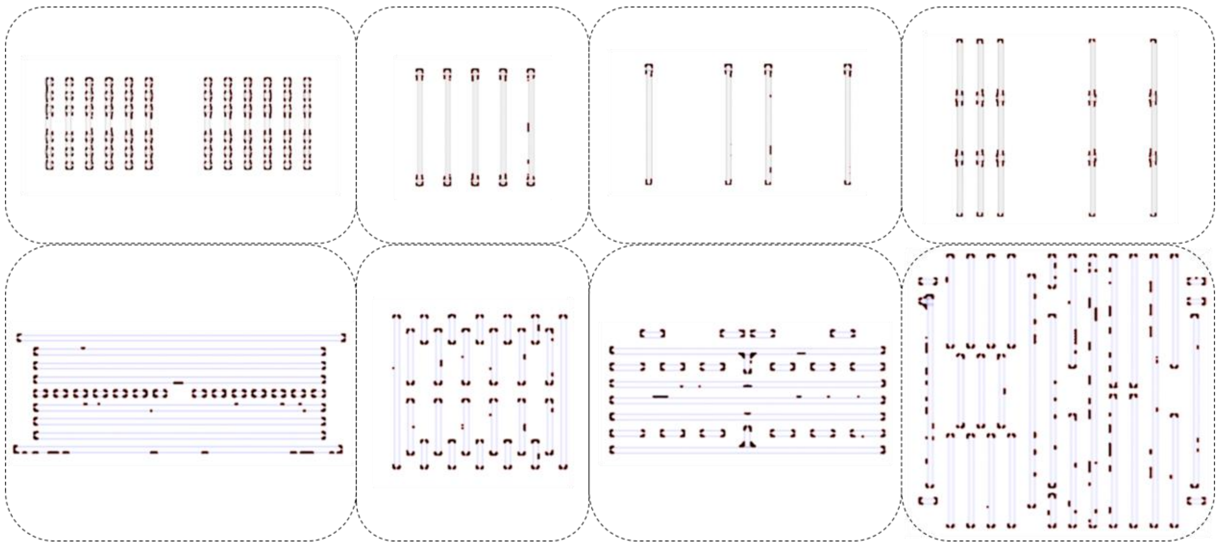
EPE is usually connected to fast lithography simulation techniques in order to estimate the number of hotspots in large circuits. However, the use of fast lithography simulation is not appropriate in context of VCSA design method, since the neighboring influence is very predictable. Thus, simulations can rely only on blocks and not on the entire circuit. It makes detailed lithography simulations much faster and eliminates the need for fast approximated algorithms. Also, as the lithography behavior of templates are not supposed to be very different, the EPE data must be as precise as possible.

In order to achieve the following results on the studied VCSA templates, available in (DAL BEM, 2012), the EPE data is generated using lithography simulations performed using a commercial tool. These simulations are followed by an image inspection algorithm running over pixel precision. This algorithm comprises a list of all found EPE distances in the lithography simulation, composing a very detailed analysis on its data.

The evaluated layouts are the ones already depicted in Figure 2.14. They are built over the predictive FreePDK 45nm technology rules, using the minimum allowed distances (FREEPDK, 2012). Only M1 and polysilicon masks were taken into account.

Figure 2.18 shows the visual representation of EPE achieved data. The highlighted hotspots are placed mainly at end-of-lines and jogs, as previewed in the qualitative analysis. Another critical pattern evidenced is the polysilicon enclosure around contact.

Figure 2.18 - EPE hotspots, achieved from lithography simulation on masks depicted in Figure 2.14 and in Figure 2.15, followed by image inspection algorithm.



Source: (DAL BEM, 2012)

Table 2.1 lists the average and the worst case values of EPE, obtained by image inspection over each template.

Table 2.1 - Average and the worst cases of EPE

Template	**EPE per mask (nm)**			
	M1 (worst case)	**M1 (average)**	**Poly (worst case)**	**Poly (average)**
VCTA	15.714	0.494	5.714	0.995
VCC	14.286	0.818	11.429	0.484
INVA	24.286	0.748	10.000	0.392
VCLB	21.429	0.490	10.000	0.510

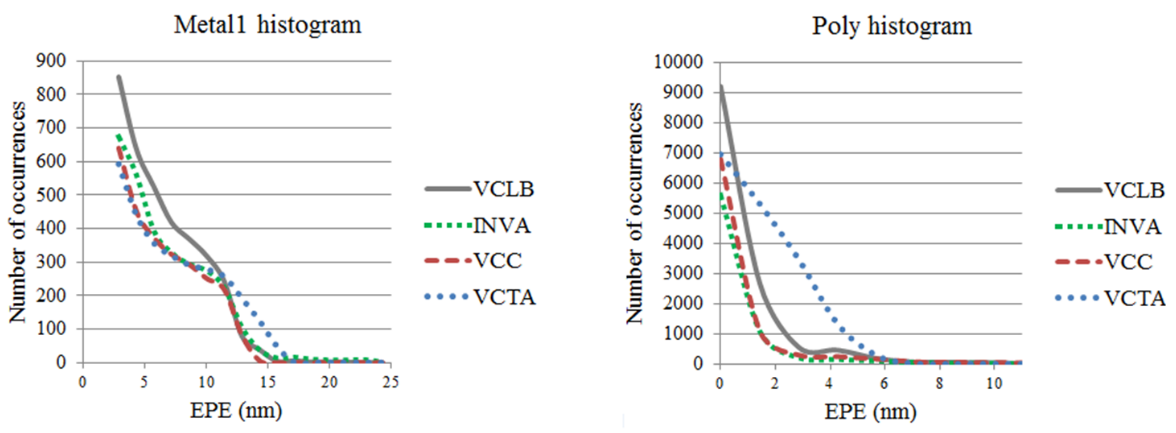
Source: (DAL BEM, 2012)

In M1 masks, it is possible to observe that the worst case values are significantly decreased in templates which follow the unidirectional lines rule, as the VCTA and the VCC ones. On the other hand, average values are dominated by the presence of segmentation, which is high in VCC and INVA approaches.

In polysilicon masks, the worst case values seem to have strong relationship to the chosen pitch, which is narrower in the VCTA. The average values increase mainly due to the increasing of the number of poly contacts, suffering also with end-of-lines distortions. In this case, the strategy of inserting poly contacts only in the poly stripes extremities is advantageous, as adopted in VCC and INVA.

Figure 2.19 shows EPE results through histogram representation. It details the values' deviation, which could not be extracted from the data in Table 2.1. It evidences how the worst case values on M1 of INVA and VCLB do not represent the overall behavior of these templates, corresponding to very specific points, so being easily repaired. It is also possible to observe clearly the superiority on lithography quality of polysilicon lines with fewer contacts.

Figure 2.19 - M1 and polysilicon histograms of edge placement error (EPE) values, obtained by image inspection over lithography simulations.



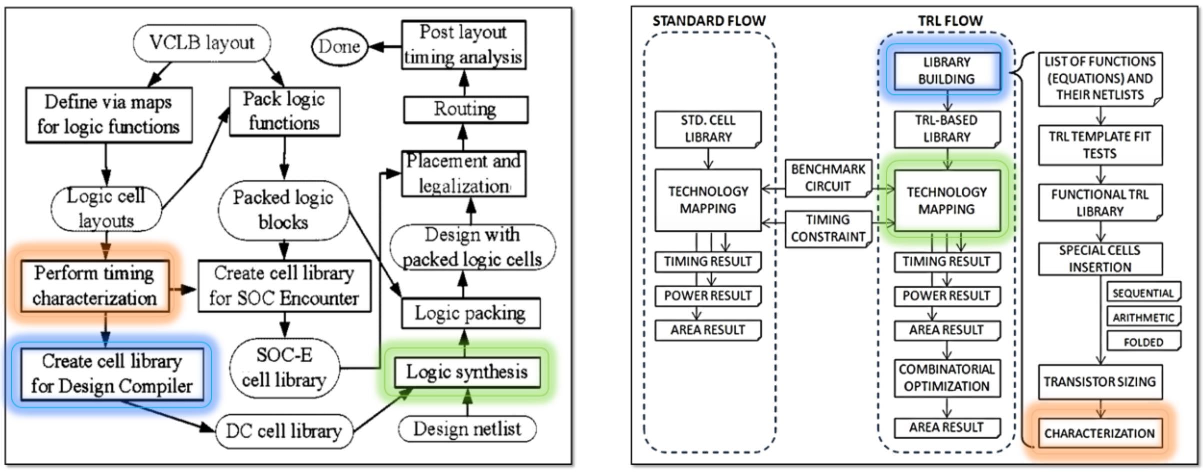
Source: (DAL BEM, 2012)

2.3.2 Electrical Performance Evaluation

Electrical performance can be defined as the speed of signals transitions and the power consumption of a cell, block or circuit. The evaluation of the electrical performance of VCSA-based designs are similar to the ones implemented in standard cells methodology. It is based on extracting a transistor network (and its parasitic elements) from the layout of each configuration/function, and using a characterizer tool to electrically simulate such arcs transitions to obtain timing and power consumption data for that cell. At this point, comparisons between the cells are already possible, but usually it is more appropriate to perform such an analysis at circuit level, using the set of characterized cells as the input library to a circuit mapping tool.

Examples of this approach including library characterization followed by circuits mapping is found in (LI, 2008), (TUNG, 2011), (DAL BEM, 2011b) and in (NOURY, 2012). Figure 2.20 exemplifies some of the design flows on such references, exposing their similarity.

Figure 2.20 - Design flow in (LI, 2008) at left, (DAL BEM, 2011b) at right, evidencing the use of a characterizer to compose a library that is used by a mapping tool.



The electrical performance is influenced by the template transistor sizing strategy (MARRANGHELLO, 2011). In VCSA block templates related literature, different alternatives are employed. In (TUNG, 2011), the transistor sizes are chosen to achieve roughly equal rise and fall output transition times. In (RAN, 2006a) and in (DAL BEM, 2011b), minimum transistor sizes for the target technology are used. Also in (DAL BEM, 2011b), another sizing strategy is presented: transistor widths that achieve good compromise between area and delay, obtained from the performance analysis on ring oscillators and from samples of critical paths in sample circuits.

Besides transistor sizing, another technique for enhancing electrical performance is the transistor folding (or fingering), defined by connecting parallel transistors sharing the same control signal. As mentioned in (PONS, 2010), this technique emulates wider transistors whose width is a multiple of the basic transistor, behaving as a discrete transistor sizing strategy. In (RAN, 2006a), the combination of folding technique and minimum sized transistors is brought as possible solution to meet the circuit timing requirements, however, no deep analysis is performed. This combination is also present in (DAL BEM, 2011b), where the use of the folding for creating alternative P/N ratios is also mentioned. In the VCSA block templates context, the folding technique is more suitable for buffers and inverters, as they usually underuse the available template resources and have great potential for improving circuit's timing performance.

Observing the achieved results on the mentioned references, it is possible to conclude that timing and power constraints commonly are not the bottleneck of using VCSA-based designs, but area. In (TUNG, 2011), the circuits designed using the most efficient proposed templates (SLVC5P variants) presented an increase of 15% to 51% in circuit delay when compared to standard-cell design approach. The increasing in power consumption ranged from 30% to 86%, whereas circuit area overhead varies from 151% to 232%. In (RAN, 2006a), the circuits designed using the proposed VCGA template, when compared to the same circuits designed using standard-cells, an increasing in circuit delay of 33%, 17% in power consumption, and 116% in area overhead. Similarly, when using VCTA, mapped benchmark circuits presented an average augmentation of 4% in power consumption, respecting the same timing constraints, and resulting an area overhead of around 100% (DAL BEM, 2011b).

2.3.3 Logical Capacity Evaluation

The logical capacity of a given VCSA block template is the set of different logical functions that a single instance of such block can express by its vias configuration. This attribute carries an inherent relationship to the block template area: usually, a template with more layout resources (as transistors and metal lines) presents a larger logical capacity but also requires a larger area, composing a "trade-off" scenario. This trade-off is critical, since a mistaken tuning on it can cause prohibitive area penalty, as demonstrates the work presented in (DAL BEM, 2011a) and in (DAL BEM, 2011b), summarized below.

- Impact on circuits area

The area overhead and circuit performance penalty are the most critical drawbacks of regular layout design in comparison to the most popular standard cell design. Complex functions can be used to reduce the overall number of transistors of the design, to mitigate the cost of adopting a more restrictive design space for improving lithography yield.

The extensive investigation presented in this section is based on the VCSA template called VCTA, proposed by Pons *et al.* (PONS, 2010), as such approach is quite interesting in terms of design flexibility due to very fine-grained configurability.

Each VCTA unit has a pre-defined number of metal lines (interconnection wiring) and transistors (PMOS and NMOS). Each amount of the available resources chosen is treated as “template variant”. We will assume that a single design cannot use more than one template variant at the same time.

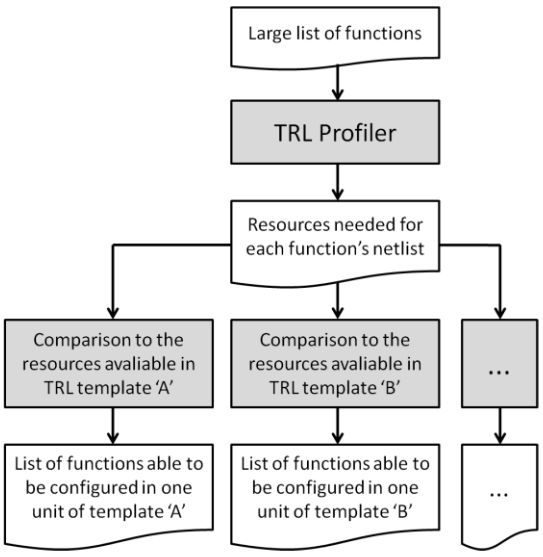
The comparison between VCSA design and standard cell based circuit was performed on the technology mapping step. In summary, a set of benchmark circuits were mapped addressing different cell libraries and the resulting circuit area were compared. Thus, in order to evaluate both design methodologies, the libraries were generated focusing on that, and applied as input for the mapping task.

To build the VCSA based-libraries, we should know which logic functions fit well and do not overflow the target template resources. The first part of this task was to convert the logic functions into transistor networks, and then extracted the information of its minimum required resources for cell implementation. A specific tool, named TRL-profiler, described later on this text, was developed for that purpose.

Once the minimum required layout resources for implementing a list of functions have been defined by our TRL-profiler tool, it could be matched to the resources offered by the different template variants under evaluation. This way, the template acts as a filter to select the functions which fits it. Figure 2.21 illustrates the flow related to the composition of the lists of functions.

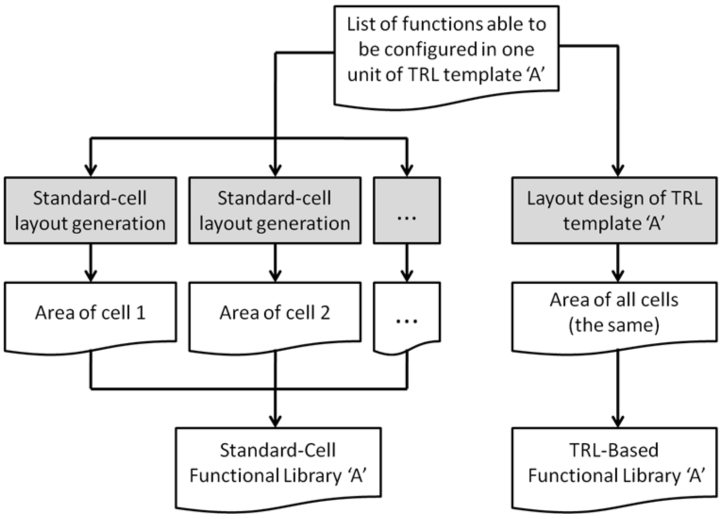
Those lists were the basis of libraries generation. A library description carries a large variety of information. Usually, the most important data for the mapping tool are the functional behavior (logic equation) as well as the timing, power consumption and area characteristics of each cell. In order to balance the timing and power consumption in both approaches the same transistor sizing was applied. In this way, the equivalent VCTA and standard cell libraries had the same logic functions and similar timing and power characteristics. The main difference is actually related to the cell area. In VCTA template variant, each cell had the same area to respect the regularity constraints, whereas in standard cell the library height were optimized to a minimum possible value. Figure 2.22 illustrates the applied flow for composing the set of libraries.

Figure 2.21 - Flow for defining the lists of functions.


Large list of functions
TRL Profiler
Resources needed for each function's netlist
Comparison to the resources avaliable in TRL template 'A'
Comparison to the resources avaliable in TRL template 'B'
...
List of functions able to be configured in one unit of template 'A'
List of functions able to be configured in one unit of template 'B'
...

Source: (DAL BEM, 2011a)

Figure 2.22 - Flow for libraries composition.


List of functions able to be configured in one unit of TRL template 'A'
Standard-cell layout generation
Standard-cell layout generation
...
Layout design of TRL template 'A'
Area of cell 1
Area of cell 2
...
Area of all cells (the same)
Standard-Cell Functional Library 'A'
TRL-Based Functional Library 'A'

Source: (DAL BEM, 2011a)

After building the cell libraries, the mapping process were performed considering a set of benchmark circuits, and applying each one of the libraries generated. Making so, all possible template-benchmark combinations are reached for both VCTA and standard cell design styles.

A set of 21 different VCTA template variants has been tested, comprising all possible combinations among 2, 4 and 6 input M1 lines, and 2 to 8 transistors in each plan (pull-up PMOS and pull-down NMOS). Each of the 21 generated lists of logic functions, corresponding to each evaluated template variant, were considered as basis to compose both VCTA and standard cell libraries, totalizing 42 distinct cell libraries to be used in the mapping of benchmark circuits. A set of 14 combinational ISCAS85 benchmark circuits (BRGLEZ, 1985) was applied in the experimental analysis. Each benchmark was mapped taking into account all the 42 generated cell libraries.

The circuit area obtained for the same benchmark mapped addressing different VCTA template variants had shown an average difference of 52% in respect to the best area attained. In the worst case scenario, comparing always the less efficient tested VCTA template variant against the best one, for all benchmarks, the average area difference goes up to **151%**.

These data demonstrate how the logical capacity versus block area trade-off is critical, and the next step is to discuss the practical issues of its tuning, being the starting point to have available, at least, both template information, the block area and its logical capacity. The block template area is easily obtained by drawing a single block layout respecting the rules of the desired target technology. However, obtaining the block logical capacity is not a trivial task. The current available methods in literature are summarized as follows.

- Logical capacity extraction methods

In (RAN, 2006a), the logical capacity of VCC template is obtained over the dual-Euler trail principle. Consider the graph representation of a transistor network. A trail in a graph is an alternating sequence of nodes and incident edges beginning and ending with nodes, with no edge repeated. A dual trail is a pair of trails, one in pull-up and other in pull-down plane, which consist of the same sequence of edges. A dual-Euler trail is a dual trail which includes all of the edges of a dual TTSPM (two-terminal series-parallel multigraphs). If an *n*-literal function has a single dual-Euler trail, it can be implemented in a *n*-VCC, because of this characteristic single transistor stripe in each plan.

Therefore, all of *n*-VCC realizable functions are enumerated by finding all of the dual-Euler trails with *n* edges. In that work, only non-equivalent functions are taken into account and, when possible, concatenation of two trails is performed, *i.e.*, two functions sharing the same template. A pseudo-code for finding all dual-Euler rails based on pair-wise concatenation is described in (RAN, 2006a), and the obtained logical capacity is summarized in Table 2.2 and in Table 2.3. In Table 2.3, TF stands for Total Functions and RF stands for realizable functions, while FC stands for Functional Coverage.

The method applied for VCC template is not valid for VCTA one, as in this last template, the transistor gate stripe of both pull-up and pull-down planes are not aligned. In the original approach (PONS, 2010), only the implementation of a reduced set of logic functions in VCTA is investigated: an inverter, an XOR, a 2-input NAND, a 4-input NAND, an AND-OR and an OR-AND. In a posterior, analysis in (DAL BEM, 2011a), a specific tool is developed, called TRL-profiler. The TRL-profiler needs a set of transistor networks as input, which will pass through a “fit” test in a target VCTA variant. This approach is not similar as the used in (RAN, 2006a), which is exhaustive, reaching all the functions by construction.

Table 2.2 - *n*-ViaCC realizable non-equivalent functions (RAN, 2006a).

n	$\overline{f} = \overline{f}(x_1, \cdots, x_n)$	
2	x_1x_2	$x_1 + x_2$
3	$x_1x_2x_3$	$x_1 + x_2x_3$
	$x_1(x_2 + x_3)$	$x_1 + x_2 + x_3$
4	$x_1x_2x_3x_4$	$x_1 + x_2x_3x_4$
	$x_1(x_2 + x_3x_4)$	$x_1 + x_2 + x_3x_4$
	$x_1x_2(x_3 + x_4)$	$x_1 + x_2(x_3 + x_4)$
	$x_1(x_2 + x_3 + x_4)$	$x_1 + x_2 + x_3 + x_4$
	$x_1x_2 + x_3x_4$	$(x_1 + x_2)(x_3 + x_4)$
5	$x_1x_2x_3x_4x_5$	$x_1 + x_2x_3x_4x_5$
	$x_1(x_2 + x_3x_4x_5)$	$x_1 + x_2 + x_3x_4x_5$
	$x_1x_2(x_3 + x_4x_5)$	$x_1 + x_2(x_3 + x_4x_5)$
	$x_1(x_2 + x_3 + x_4x_5)$	$x_1 + x_2 + x_3 + x_4x_5$
	$x_1x_2x_3(x_4 + x_5)$	$x_1 + x_2x_3(x_4 + x_5)$
	$x_1(x_2 + x_3(x_4 + x_5))$	$x_1 + x_2 + x_3(x_4 + x_5)$
	$x_1x_2(x_3 + x_4 + x_5)$	$x_1 + x_2(x_3 + x_4 + x_5)$
	$x_1(x_2 + x_3 + x_4 + x_5)$	$x_1 + x_2 + x_3 + x_4 + x_5$
	$x_1x_2 + x_3x_4x_5$	$x_1x_2 + x_3 + x_4x_5$
	$x_1x_2 + x_3(x_4 + x_5)$	$(x_1 + x_2)(x_3 + x_4x_5)$
	$(x_1 + x_2)x_3(x_4 + x_5)$	$(x_1 + x_2)(x_3 + x_4 + x_5)$

Source: (RAN, 2006a)

Table 2.3 - Functional coverage of n-ViaCC (RAN, 2006a).

n	2	3	4	5	6	7	8
TF	2	4	10	24	66	180	522
RF	2	4	10	22	50	112	280
FC(%)	100	100	100	91.7	75.8	62.2	53.6

Source: (RAN, 2006a)

For this reason, the TRL-profiler works extracting the following data for each possible logic gate implementation:

a) number of input inverters, which represents the number of additional M1 lines required for internal connections;

b) presence of output inverter, which represent an additional M1 line required for cell core output signal routing;

c) PU-P and PD-N Euler path lengths (column heights), including breaks (isolated transistors) for each one;

d) number of connections in internal PU-P and PD-N networks, which represents the use of an exclusive M1 for signal routing;

e) verification if PU-P and PD-N Euler paths can begin from the VDD and GND power signals, respectively, because the remaining unused transistors in the column can be used to implement another logic gate whose Euler paths also begin in the same VDD and GND nodes.

Both items (a) and (b) above can be interpreted as a requirement of extra exclusive M1 vertical wires for each distinct input/output signal. If M3 is used for this purpose, then the signal must be connected down to M1, passing through M2, to access the polysilicon gates. Items (c) and (e) provide useful information for estimating the optimized number of switches and to determine the column height, *i.e.*, the number of pre-defined stacked transistors.

After extracting the itemized information for a given transistor network, a simple comparison to the available layout resources of the target VCTA variant is performed.

The following results use as input for the TRL-profiler a list of 4058 functions. This list comprises all possible Boolean functions with up to 4 inputs, excluding the ones equivalent by inputs permutation, named P-class[1] (SASAO, 1999), and the 76 functions of 5 and 6 inputs from the reference list known as Genlib, more specifically ‘genlib_44-6’ (SENTOVICH, 1992). Table 2.4 provides the number of those 4058 functions that are able to be implemented over a single VCTA unit of each template variant evaluated.

[1] In this case, “P” stands for “permutation”. Do not mistake this P-class with the algorithms complexity P-class, in which the “P” stands for “polynomial”, that will be mentioned later on this text.

Table 2.4 - Number of logic functions from the input list which fits each evaluated VCTA template variant, according its resources.

Number of M1 lines	**Number of transistors at each plan**						
	2	**3**	**4**	**5**	**6**	**7**	**8**
6	4	9	19	53	126	151	189
4	4	9	19	33	55	69	81
2	4	6	6	6	8	8	8

Source: (DAL BEM, 2011a)

In (TUNG, 2011), it is presented not a single but several VCLB block templates, and there is no specific tool for logical capacity extraction. It seems to be evaluated manually, performing trials of a pre-determined set of logical functions in each proposed template. Table 2.5 shows a comparison between the amount of functions in a standard cell library (STDL) against the number of realizable functions in one VCLB template variant (SLV5P).

Table 2.5 - Number of cells in standard cell library example and one VCLB variant, called SLV5P.

Inputs	**Number of functions**	
	STDL	**SLV5P**
2	11	8
3	21	12
4	16	12
5	6	4
6	6	0

Source: (TUNG, 2011)

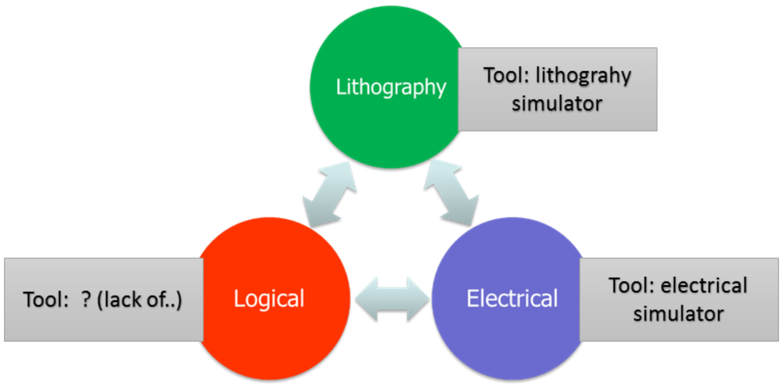
- Thesis' Contribution

As discussed above, the current methods in literature for determining the VCSA templates logical capacity are based on the characteristics of the target template, in a way that changing a feature in its design (*e.g.* removing or adding poly alignment between pull-up and pull-down planes) becomes the method unsuitable. There is a lack for a general and efficient method that could be applied over any VCSA block template.

The lack for a tool to evaluate the logical capacity of any block template is highlighted especially when we observe the larger scenario of evaluating the design quality of a VCSA template. In this scenario, the analysis of the lithography efficiency is covered by lithography simulations, and electrical performance analysis covered by characterization tools, both general methods suitable to any block template, but there is no similar tool for covering logical capacity evaluation, as drawn in Figure 2.23.

This lack is fulfilled in this thesis. The Chapter 4 presents the proposed solution, while the background required to its understanding is exposed in the next chapter.

Figure 2.23 - Aspects to consider when evaluating design quality of a VCSA template.



3 SAT IN CAD

This chapter provides basic background in SAT, beginning with fundamental definitions, followed by a discussion about SAT complexity and a list of applications on CAD problems, including details on a case of special interest.

3.1 Definitions

"Satisfiability" and "SAT" are common abbreviations for Boolean Satisfiability Problem in computer science. It consists in determining if a logic formula is "satisfiable". "Logic formula" and "satisfiable" can be defined as follows.

A Boolean expression, also called logic formula, is composed by variables, operators and parentheses. The single purpose of parentheses is to set precedence on operations. The operators are:

- "AND" or "conjunction" or "∧"
- "OR" or "disjunction" or "∨"
- "NOT" or "negation" or "¬"

A formula is said to be "satisfiable" if it can be made TRUE by assigning appropriate logical values (TRUE or FALSE) to its variables. On the other hand, a logic formula is said to be "unsatisfiable" if it evaluates FALSE identically to all possible assignments to its variables.

A literal is either a variable, then called positive literal, or the negation of a variable, then called negative literal. A clause is a disjunction of literals (or a single literal). A formula is said to be in conjunctive normal form (CNF) if it is a conjunction of clauses (or a single clause).

For instance, "x1" is a positive literal, "¬x2" is a negative literal, "x1 ∨ ¬x2" is a clause, and "(x1 ∨ ¬x2) ∧ (¬x1 ∨ x2) ∧ ¬x1" is a formula in conjunctive normal form. This formula is satisfiable by choosing x1=FALSE, x2=FALSE, since (FALSE ∨ ¬FALSE) ∧ (¬FALSE ∨ FALSE) ∧ ¬FALSE evaluates to TRUE ∧ TRUE ∧ TRUE, so resulting TRUE. In contrast, the CNF formula (x1 ∧ ¬x1), consisting of two clauses of one literal, is unsatisfiable, since for the two possibilities x1=TRUE and x1=FALSE it evaluates to TRUE ∧ ¬TRUE and FALSE ∧ ¬FALSE, resulting FALSE in both cases.

Every logic formula can be transformed into an equivalent conjunctive normal form, which may, however, be exponentially longer. For example, transforming the formula 3.1 into conjunctive normal form yields the equation 3.2.

$$(x1 \land y1) \lor (x2 \land y2) \lor ... \lor (xn \land yn) \tag{3.1}$$

$$\begin{aligned}(x1 \lor x2 \lor \ldots \lor xn) \land (y1 \lor x2 \lor \ldots \lor xn) \land (x1 \lor y2 \lor \ldots \lor xn) \land \\ (y1 \lor y2 \lor \ldots \lor xn) \land ... \land (x1 \lor x2 \lor \ldots \lor yn) \land (y1 \lor x2 \lor \ldots \lor yn) \\ \land (x1 \lor y2 \lor \ldots \lor yn) \land (y1 \lor y2 \lor \ldots \lor yn).\end{aligned} \tag{3.2}$$

Observe that equation 3.1 is a disjunction of ‘n’ conjunctions of 2 variables, whereas equation 3.2 comprises of ‘2^n’ clauses of ‘n’ variables.

3.2 SAT Complexity

Complexity classes are groups of computational problems related to the amount of resources (typically execution time) needed to be solved. Extensive explanation on complexity classes can be found in (BOVET, 1994), but a brief discussion is brought herein to better understand the SAT behavior.

P-class comprises problems that can be solved in polynomial time by a deterministic Turing machine. NP–class, in turn, comprises problems that can be solved in polynomial time by a non-deterministic Turing Machine. Clearly, P-class is included into NP-class. However, there is not yet widely accepted proof that P-class is different from NP-class in theory of computing (SRINIVASAN, 2011). Experimentally, some NP problems seems to be harder than P ones, being treated by algorithms that require super-polynomial time to find a solution.

NP-complete is a subclass of NP, and every NP problem can be reduced to a NP-complete problem in polynomial time. It can be said that NP-complete is the subset containing the hardest problems of NP, and the ones that are most unlikely to be in P class.

SAT is a problem with NP-complete complexity, as proved by Cook (COOK, 1971) and Levid (TRAKHTENBROT, 1974). Thus, there is not a known algorithm that solves SAT efficiently, correctly, and for all possible input instances, and it is usually believed that such an algorithm does not exist. However, many instances of SAT that occur in practice experimentally have been shown to be solved rather efficiently using SAT-solvers.

Depending on the particular structures of the used logic formulas, a special case of the Boolean satisfiability problem is configured. Such special case can be classified as SAT restriction or SAT extension. As expected, the problem complexity on SAT restrictions may be sometimes lower than the unrestricted SAT, but the extensions can only be more complex.

A SAT restriction that deserves special attention here is the one comprising only CNF formulas, sometimes called CNFSAT, which presents complexity in NP-class, like the unrestricted SAT problem.

An extension that has gained significant popularity, since 2003, is satisfiability modulo theories (SMT) that can enrich CNF formulas with linear constraints, arrays, all-different constraints, etc. Such extensions typically remain NP-complete, but very efficient solvers that can handle many such kinds of constraints are available nowadays.

3.3 SAT Application in CAD Problems

In last years, SAT formulation has been used to solve several CAD problems, as illustrated in this non-exhaustive list:

- PLA folding

Quintana *et al.*, in (QUINTANA, 1995), propose an algorithm for optimum PLA folding based on its formulation as a problem of Boolean satisfiability.

- Asynchronous circuits synthesis

In (GU, 1995), it is presented a general and efficient partitioning approach to the synthesis of asynchronous circuits based on complete state coding (CSC) of signal transition graphs (STGs) satisfaction through satisfiability (SAT) model. In (KHOMENKO, 2003), the synthesis of asynchronous circuits is also addressed with SAT, but this time without the construction of the STGs, in a formulation that use only the information about causality and structural conflicts between the events involved in a finite and complete prefix of its unfolding. The same problem is also treated in (KHOMENKO, 2004), with an incremental SAT model.

- FPGA routing

In (WOOD, 1998), it is developed a novel formulation of both signal routing and “routability” estimation that relies on a rendering of the routing constraints as a single large Boolean equation. In (NAM, 1999a), the authors also addresses the problem of detailed FPGA routing using Boolean formulation methods. These two last mentioned papers evolve to

(NAM, 2002). The previous attempts at FPGA routing using Boolean methods were based on binary decision diagrams that limited their scopes because of size limitations, but in (NAM, 2002), it is presented a FPGA detailed routing formulation for a new search-based satisfiability (SAT) that handles simultaneously all channels in an FPGA. In (HE, 2005), the author addresses a new segmented channel routing problem with pin rearrangements in FGPA technology, using an efficient SAT-based approach to solve the problem. The focus on routing FPGA multi-pin nets is clear in the SAT formulation in (MUKHERJEE, 2010). In (NAM, 1999b), the author presents a study on the FPGA routing through SAT, exposing that initial Boolean-based approaches for routing used binary decision diagrams (BDDs) to represent and solve the layout problem. However, BDDs limit the size and complexity of the FPGAs that can be routed, leading these approaches to concentrate only on individual FPGA channels.

- FPGA mapping

In (LING, 2005), the author presents a developed algorithm, based on Boolean satisfiability (SAT), that is able to map a small subcircuit into the smallest possible number of lookup tables (LUTs) needed to realize its functionality. In (SAFARPOUR, 2006), it is proposed a Boolean matching approach for FPGA technology mapping targeting networks of programmable logic blocks, and to do this the work breaks the original SAT problem into easier and smaller ones. In (HU, 2007), the author provides optimizations when addressing the same problem, by exploring function and architectural symmetries.

- Timing analysis

In (ROY, 2007), it is proposed an accurate technique based on SAT for computing critical delay of a circuit under a bounded delay model.

- Partitioning and placement of heterogeneous reconfigurable SoCs

The work in (WILDERMANN, 2011) presents three encoding variants that unify partitioning and placement of heterogeneous reconfigurable SoCs. In particular, their SAT-based approach uses a preprocessing mechanism that identifies partitioning which inevitably lead to unfeasibility and then incorporates this information into the symbolic encoding for calculating feasible placements.

- Standard-cell layout

The work in (IIZUKA, 2004) proposes a cell layout synthesis method via Boolean satisfiability (SAT). However, as stated in (TAYLOR, 2007), its layout style is too much simplistic for several purposes (*e.g.* doglegs are not allowed in routes), and would result in much larger layout areas than those produced by other methods.

- Standard cell routing

The work in (TAYLOR, 2007) will be detailed later on, but for now consider that it is an example of standard-cell routing method based on SAT that exploits the layout regularity. The work in (RYZHENKO, 2012) is another example of using SAT to perform standard-cell routing. In this paper, the authors present a way to make this technique more effective by introducing pruning techniques, and also for further optimize routing quality within the SAT formulation through the use of successive more stringent constraints. More recent work, presented in (CORTADELLA, 2014), represents an evolution of previous basic SAT formulations, using a novel encoding scheme, graph theory to support floating terminals, efficient heuristics to reduce the computational cost, and minimization of the number of unconnected pins in case the cell is unroutable. Even incremental routing is addressed by SAT in (YANG, 2010). In this approach, multiple nets in each hotspot region are simultaneously ripped up and rerouted based on Boolean satisfiability (SAT). The hotspot patterns, which are described and stored in a pre-built library, are forbidden to appear in the reroute through SAT constraints.

- Transistor array layout

The work in (LIN, 2009) presents an automatic placement and routing strategy for a high-density, super-regular, double-gate transistor-array based layout. The placement is performed by a custom algorithm, while the routing is solved by a combination between a custom algorithm and SAT solver.

3.4 Case of interest: standard-cells regular routing using SAT

As an example of special interest in this thesis, a case of standard cells routing with regular layout solved by SAT is presented in more details in this section.

The following formulation is developed in Taylor’s thesis (TAYLOR, 2005) and summarized in a highly referenced paper of the same author (TAYLOR, 2007).

In order to build this formulation, the authors divided the entire routing space in a grid. Each grid segment has an associated Boolean variable which is 1 if and only if that segment is filled in with a routing line. Then, it is stated a set of routing rules that force an assignment to the Boolean variables to correspond to a valid routing solution. Finally, each rule is "enforced" by a set of clauses.

The used variables and rules are summarized below. Their enforcing clauses are not shown because such level of detail is not necessary for now.

Variable SEGMENT(x,i,j): Whether the grid segment at row 'i' and column 'j' of layer 'x' is filled or empty (TRUE or FALSE, respectively).

Variable NET(segment): Each segment has associated with it a bit vector which denotes the net ID of the net it is carrying.

Variable SUBNET(segment): Each segment has associated with it T valid bits. The 'k'th valid bit T vector indicates whether the segment carries the 'k'th subnet of the net whose ID is given by the previous described variable.

Variable IOTERMINAL(i,j): Are TRUE if M2 grid point (i,j) corresponds to a pin terminal of the 'n'th I/O net (whose net ID is not in general 'n')

Variable TERMINAL(i,j): Are TRUE if M2 grid point (i,j) corresponds to a pin terminal of some net.

Rule SEGMENT 1: For each pair of adjacent segments, if both are filled in, they must carry the same net ID (otherwise the net carried by one would be shorted to the net carried by the other).

Rule SEGMENT 2: For each endpoint 'e' of each filled-in segment and for each subnet 's' carried by the segment, if 'e' is not a terminal of 's' (neither a fixed terminal nor a pin terminal), then exactly one of the segments neighbors 'e' carries 's'. Note that only segments in the VIA1 and M2 layers may be incident to pin terminals.

Rule FIXED TERMINAL 1: If grid point gij is a fixed terminal of subnet s(n, k), then exactly one segment incident to gij carries s(n, k).

Rule FIXED TERMINAL 2: If grid point gij is a fixed terminal of subnet s(n, k), then any filled-in segment incident to gij must carry net n. Note that the net ID gij of fixed terminals gij is actually a constant, rather than a b-bit vector of Boolean variables; thus the logic required to enforce the above rules is simpler than is needed for pin terminals.

Rule PIN TERMINAL 1: For each possible pin location -- that is, for each M2 grid point (i,j) -- if that location implements the pin terminal of the ‘n’th I/O net, then exactly one of M2 grid point (i, j) neighbors carries the subnet corresponding to that terminal. Note that the subnet index corresponding to a pin terminal is always the maximum subnet index of that net (if an I/O net ‘n’ has 4 terminals, and thus 3 subnet indices 0, 1, and 2, the subnet index of the pin terminal will be 2).

Rule PIN TERMINAL 2: For every possible pin location (i, j), if that location implements the pin terminal of the ‘n’th I/O net, then by rule PIN TERMINAL 1, one of grid point neighbors carries the pin terminal subnet ‘s’ corresponding to the ‘n’ th I/O net. Since we only want one adjacent segment adjacent to carry ‘s’ (as in rule FIXED TERMINAL 1), we make sure that VIA1(i,j) segment does not carry ‘s’

Rule PIN TERMINAL 3: If a grid point is the pin terminal of net ‘n’, the lth I/O net, then any segment incident to terminal must carry net ‘n’.

Rule PIN TERMINAL 4: Every M2 grid point can be the site of at most one pin terminal.

Rule PIN TERMINAL 5: For every I/O net ‘n’, at least one possible pin terminal location actually implements the pin terminal of net ‘n’.

Rule PIN TERMINAL 6: For every I/O net ‘n’, at most one possible pin terminal location actually implements the pin terminal of net ‘n’.

Rule PIN TERMINAL 7: This rule enforces the definition of the pin variable class. Pin variable Pij is 1 if M2 grid point (i,j) is the location of a pin terminal for some net.

The following rules arise from the particular layout style being used, from DRC correctness, or from DFM considerations. Some rules, which are based on DRC correctness, are shared between both styles; these are listed first. The logic to implement these rules is so simple that an English description of the rules suffices.

Rule DRC 1: Vertically adjacent contacts are not allowed.

Rule DRC 2: Horizontally adjacent vias are disallowed; vertically adjacent vias are allowed only if they share the same net.

Rule DRC 3: A filled-in via segment must be adjacent to at least one filled-in M1 segment (to satisfy minimum metal rules).

Rule FEOL 1: M1 segments horizontally adjacent are not allowed. This rule ensures DRC-correctness in the FEOL style, but is not needed in the BEOL style.

Rule FEOL 2: If two filled-in M2 segments s and s' lie in the same row and are separated by exactly two empty segments, then the horizontal tracks above and below s and s~ must be empty in the columns between and including s and s'. This rule, as well as FEOL Rule 3 and BEOL Rule 1, exist to accommodate hammerhead corrections for OPC.

Rule FEOL 3: If a filled-in M2 segment s lies on the left or right boundary of the brick (column 0 or column 1), then the M2 segments directly above and below s are empty.

Rule BEOL 1: If three filled-in M1 segments lie in row 'r' and occupy columns j, j + 1, and j ÷ 2, then it is not allowed for both Ml(r+l)j and Ml(r+l)(j+l) to be empty; nor is it allowed for both Ml(r+l)(j+l) Ml(r+l)(j+2) to be empty. If three filled-in M1 segments lie in row 1 and occupy columns j, j ÷ 1, j ÷ 2, then it is not allowed for both M10j and M10(j+l) to be empty; nor it is allowed for both M10(j+l) and M10(j+2) to be empty.

It is not necessary to detail the above referenced model with examples because for the further understanding of this thesis it is enough to get its general idea, and keep the variables and rules listed here as reference in later model comparisons.

For now, it is crucial to realize that the above described system is not based on listing all possible paths between two layout terminals as variables. Instead, it is based on modeling the possible grid points as variables, which is very convenient as it represents, in this scenario, a much smaller set of possibilities. Systems based on the possible paths are useful for more limited cases, sometimes used in FPGA routing as in (WOOD, 1998).

At this point, after all the mentioned application examples, we expect to have made clear that a SAT-based solution could be employed as basis for solving the stated problem of VCSAs logical capacity extraction discussed in Chapter 2. Such model is not available in scientific literature yet, and has been investigated in this thesis. It is described in the next chapter.

4 SAT MODEL FOR VCSA BLOCKS LOGICAL CAPACITY EVALUATION

This chapter starts listing all the requirements that a general model must attend in order to support user-defined VCSA templates, including the ones present in related literature and presented in previous chapters. Then, the development of a novel model is proposed.

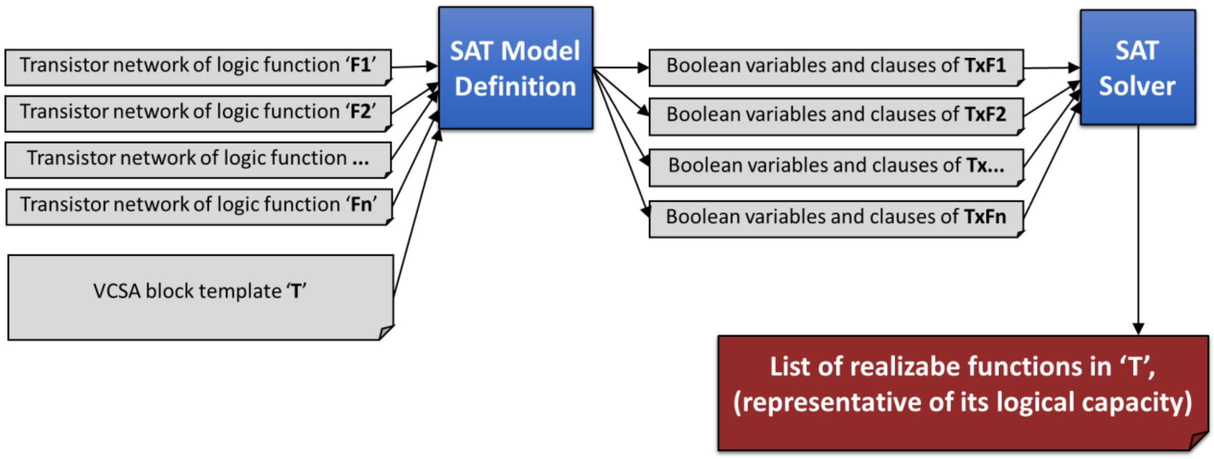
4.1 Requirements

The term "tie equivalent" of a transistor network is used herein to all the transistor networks that are formed by the addition of "tied" transistors (gate always at high or low logic level) to the original network, in a way that the logic function remains equivalent.

The SAT model presented in this chapter provides the input for a SAT solver to become able to answer the following question: "Can a given VCSA block template to express a given transistor network or any of its tie equivalents?". It can be interpreted as the fundamental step into a "fitting" test, in which any given transistor network can be checked against any given VCSA block template.

With an environment that is able to answer such question, a large set of transistor networks can pass through the environment's flow in order to build a library representing the logical capacity of the target VCSA block template, this way fulfilling the proposal in Chapter 2 of this thesis: to provide an embracing and efficient tool for extracting the logical capacity of any given VCSA block template. In this case, as larger is the input set of transistor networks, the most precisely evaluated is the template logical capacity. Figure 4.24 summarizes the described flow.

Figure 4.24 - SAT-based flow for VCSA block templates logical capacity evaluation.


Transistor network of logic function 'F1'
Transistor network of logic function 'F2'
Transistor network of logic function ...
Transistor network of logic function 'Fn'
VCSA block template 'T'
SAT Model Definition
Boolean variables and clauses of TxF1
Boolean variables and clauses of TxF2
Boolean variables and clauses of Tx...
Boolean variables and clauses of TxFn
SAT Solver
List of realizabe functions in 'T', (representative of its logical capacity)

The most basic requirement for this (and any other) SAT model is that the number of variables and equations must be small enough to be treatable by any SAT solver. A naïve model to the particular problem approached in this thesis does not achieve this requirement.

The mentioned “fitting” problem can be roughly broken into two sub-problems: transistors placement and intra-cell routing. The first one consists in finding an ordering for the logic transistors to be mapped in the layout transistors. The second one is to find out, for a specific transistors placement (or ordering), if all the node connections present in the transistor network can be performed in the layout, taking in consideration its available resources as metal lines and its restrictions as diffusion abutments.

Using only the intra-cell routing problem (a subset of the target problem to solve) is already enough for illustrating the modeling difficulty. The simplest solution for modeling the intra-cell routing problem is to consider every route among two layout diffusions as a Boolean variable, then ensuring the connection by a clause in which at least one of these variables must be TRUE, representing that, at least, one connection path exists. Another clause forbidding the TRUE value in two routes that share the same layout resource (*i.e.*, metal line) finalizes this naïve model. Now assume as input for the model the VCSA block template called VCTA, depicted in Figure 2.10. The amount of possible routes between two diffusion nodes is superior to 10^{20}, becoming the modeling computationally unfeasible, configuring an even worst scenario when regarding that this amount of variables would be repeated for every connection, and that a single (maybe wrong) transistor placement is assumed from the begin.

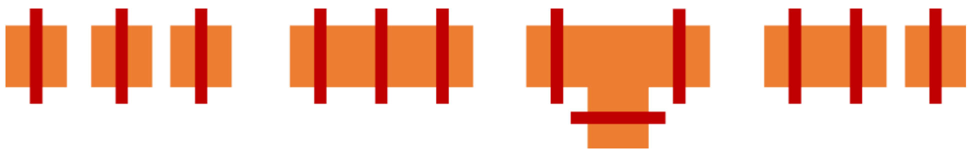
The above described naïve model, similar to the formulation presented in (RYZHENKO, 2012), can be classified as “tree-based” according to (CORTADELLA, 2014). The lack of efficiency in “tree-based” models for modeling a wire routing problem with large number of routes induced the creation of “segment-based” models, as described in (CORTADELLA, 2014) itself and in (TAYLOR, 2005), summarized in previous chapter. As already stated in the previous chapter explanation, the “segment-based” model consider all metal segments in the layout as Boolean variables, and makes them associated to vectors indicating the terminal nodes to which they are connected. This approach is much more efficient for the scenario studied in this thesis, and this model could be adapted to solve intra-cell routing in VCSA block templates, after exchanging metal line segments by vias as Boolean variables.

However, this adaption would not be enough yet for modeling the problem approached in this thesis, because the mentioned intra-cell routing models assume that the transistors placement is already performed, and this assumption is not valid here. In this case, an equivalent “tree-based” sub-model would be required for each possible transistors placement, increasing prohibitively the number of variables and equations.

In summary, the needed SAT model must present a list of characteristics that no other published model reunites yet:

- **Unrestricted input transistor network.** Some related work present a restriction of treating only series-parallel network, being an undesired characteristic as it may hide some logical capacity of the target VCTA template (UEHARA, 1981).
- **Unrestricted number of layout metal layers.** A limitation on the number of possible layout metal layers, as in (IIZUKA, 2004), would diminish the goal of being a general model.
- **Simultaneous transistors placement and intra-cell routing.** Both tasks are required for determining if a transistor network can be expressed into a VCSA block template, previewing even the possibility of different number of P-type and N-type transistors. Modeling only the wire routing would require a repetition of the solution of each possible placement or cutting part of the universe of solutions.
- **Diffusion abutments and breaks.** The user-defined VCSA block template may contain diffusion abutments for two or more transistors of the same type. The model must be prepared to deal with diffusion abutments, diffusion breaks and any kind of combination among them. Figure 4.25 illustrates some examples on possible diffusion layouts.

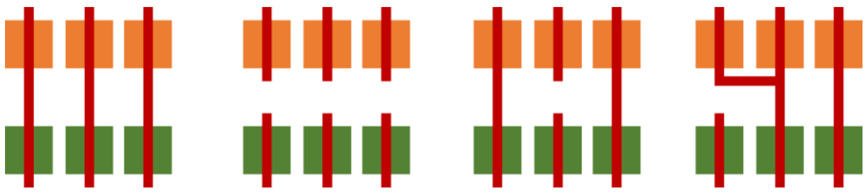
Figure 4.25 - Examples of possible diffusion layouts.



- **Poly matching.** Similar to the diffusions, any possible arrangement of poly matching among P-type and N-type transistors must be accepted. Figure 4.26 illustrates some possibilities of poly layout on VCSA block templates.
- **Tied-high and tied-low transistors.** As the number of transistors in a given transistor network may be lower than the number of available transistors on the VCSA block template, the model must make the use of tied gates as leaving them floating that would cause both logical and electrical issues. Beyond that, explicit

connections of transistor gates to supply vialines may be made in the transistors network.

Figure 4.26 - Examples of possible poly matching among P- and N- planes.



- **Possible configurable Contact layer.** As certain VCSA block templates may present a configurable contact layer beyond the via layer, as discussed in (PONS, 2010), this possibility must be taken into account in the model.
- **Supply lines restriction.** Usually, the supply lines are not intended to be present in any metal line of a block or cell layout, without criteria. Instead, it is common to have pre-defined supply lines, and the model must support such pre-definition.
- **Routing to poly.** As no transistor network restriction is imposed, the user may provide a multi-stage network, in which the output of a stage, found in a metal line, must be connected to the following state input, *i.e.*, a poly layer. The model must support this scenario.

Respecting the entire set of mentioned requirements would be enough for dealing with all VCSA block templates described in Chapter 2. However, if any unpredicted VCSA characteristic arises in future works, the model herein may be expanded to support it.

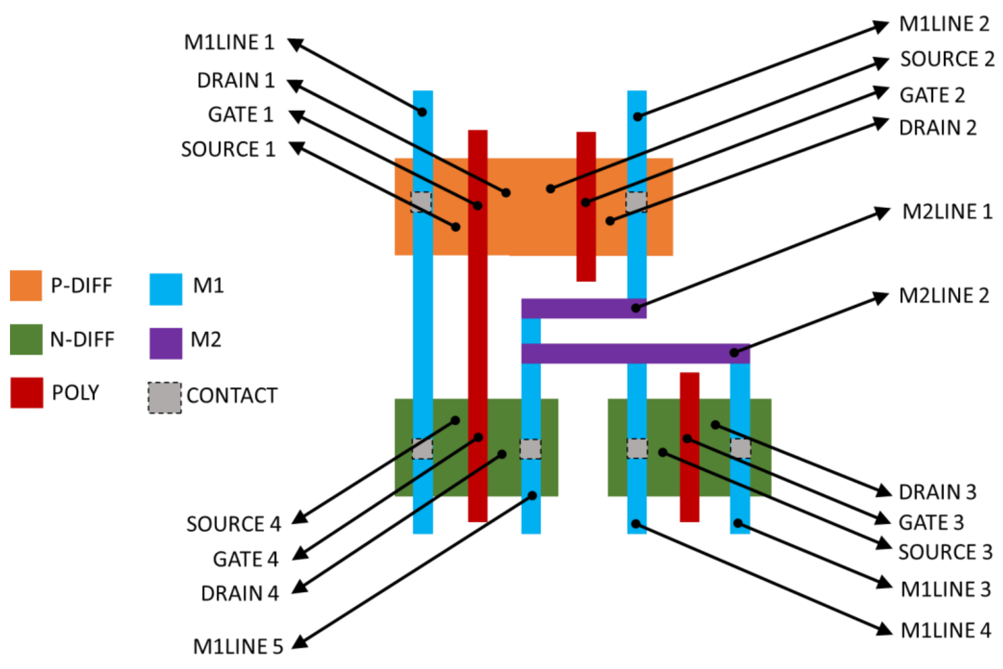
4.2 SAT Model Construction

The SAT model described in this thesis is fundamentally based on the idea of establishing virtual connections among the given layout elements of VCSA block template and the nodes of the given transistors network to be expressed on that template. For the completeness of the previous statement, definitions for the "layout elements" and for the given transistor "network nodes" are provided:

Definition of Layout Elements (e): gate, source and drain terminals of each transistor and also all metal lines of a given layout.

It does not matter if the transistor gate lines are shared or if their source/drain diffusions are in abutment with other diffusions, they are still counted individually. Figure 4.27 depicts an arbitrary example of VCSA template layout, highlighting all its “layout elements” taken into account for this modeling.

Figure 4.27 - Layout elements taken into account in a sample layout.



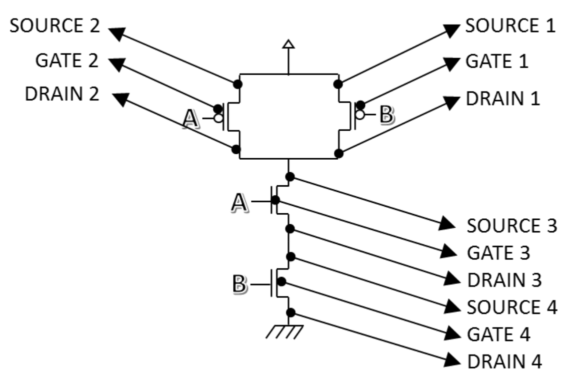
Definition of Network Nodes (n): gate, source and drain terminals of each transistor of a given transistors network.

Again, it does not matter if two of these nodes are connected (share the same name), they are counted and named individually. Figure 4.28 depicts a sample transistor network, highlighting all the “network nodes” defined for this modeling.

This model, as in many references, will be described in terms of Boolean variables, rules for these variables attributions and the enforcing clause (*i.e.*, Boolean equations) that ensure the rules implementation into a language accepted by a SAT solver. The first variable of the developed model is:

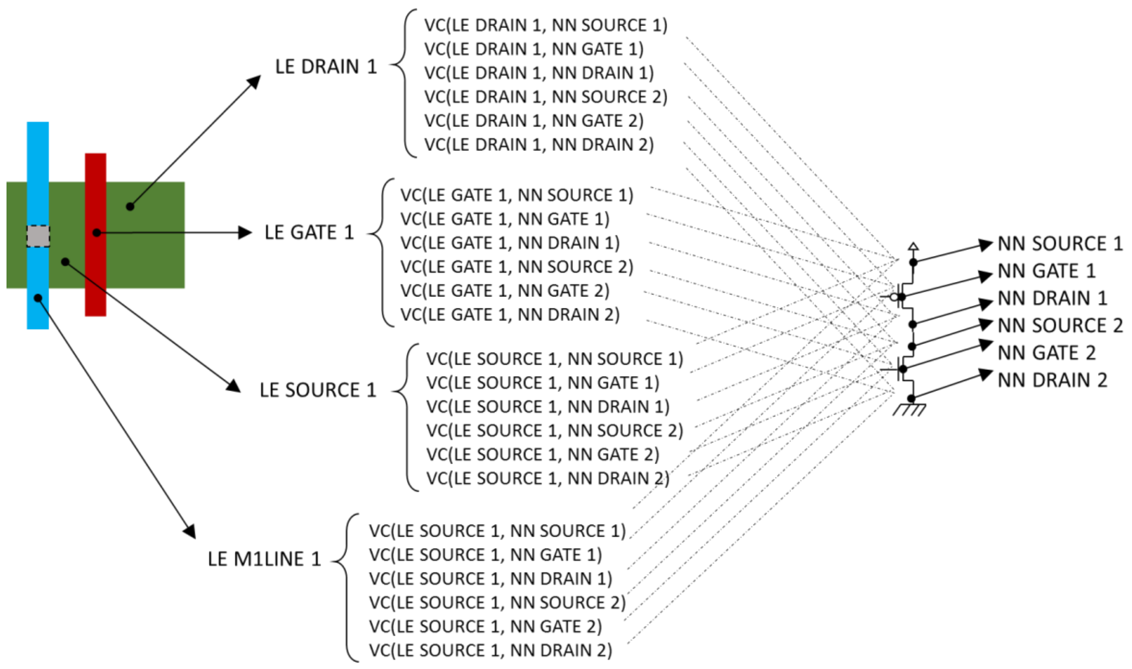
Variable VC(e,n): This variable represents the virtual connection between the layout element ‘e’ and the transistor network node ‘n’.

Figure 4.28 - Nodes taken into account in a sample transistor network.



Observe that, according to the VC variable definition, each layout element ‘e’ must present a vector of Boolean values, indicating its virtual connection to each network node, as illustrated in Figure 4.29.

Figure 4.29 - VC variables crossing layout elements with network nodes.


LE DRAIN 1
VC(LE DRAIN 1, NN SOURCE 1)
VC(LE DRAIN 1, NN GATE 1)
VC(LE DRAIN 1, NN DRAIN 1)
VC(LE DRAIN 1, NN SOURCE 2)
VC(LE DRAIN 1, NN GATE 2)
VC(LE DRAIN 1, NN DRAIN 2)
LE GATE 1
VC(LE GATE 1, NN SOURCE 1)
VC(LE GATE 1, NN GATE 1)
VC(LE GATE 1, NN DRAIN 1)
VC(LE GATE 1, NN SOURCE 2)
VC(LE GATE 1, NN GATE 2)
VC(LE GATE 1, NN DRAIN 2)
LE SOURCE 1
VC(LE SOURCE 1, NN SOURCE 1)
VC(LE SOURCE 1, NN GATE 1)
VC(LE SOURCE 1, NN DRAIN 1)
VC(LE SOURCE 1, NN SOURCE 2)
VC(LE SOURCE 1, NN GATE 2)
VC(LE SOURCE 1, NN DRAIN 2)
LE M1LINE 1
VC(LE SOURCE 1, NN SOURCE 1)
VC(LE SOURCE 1, NN GATE 1)
VC(LE SOURCE 1, NN DRAIN 1)
VC(LE SOURCE 1, NN SOURCE 2)
VC(LE SOURCE 1, NN GATE 2)
VC(LE SOURCE 1, NN DRAIN 2)
NN SOURCE 1
NN GATE 1
NN DRAIN 1
NN SOURCE 2
NN GATE 2
NN DRAIN 2

In other words, there must have a Boolean variable for every “layout element vs. network node” crossing, and if its value is TRUE then the layout element is considered virtually connected to the network node, while a FALSE value indicates the absence of such connection.

The example in Figure 4.30 depicts a very simple layout template, composed only by two transistors, and also a very simple transistor network, composed by a single transistor. This illustrative case is simple. We know, intuitively, that the given transistor network can be represented in the layout template (*i.e.*, it "fits") and there are four possible solutions to this fitting, as shows Figure 4.31.

Figure 4.30 - Very simple example of layout template and transistor network.

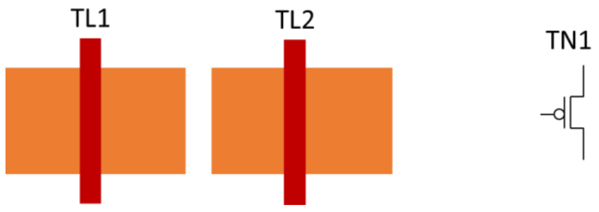
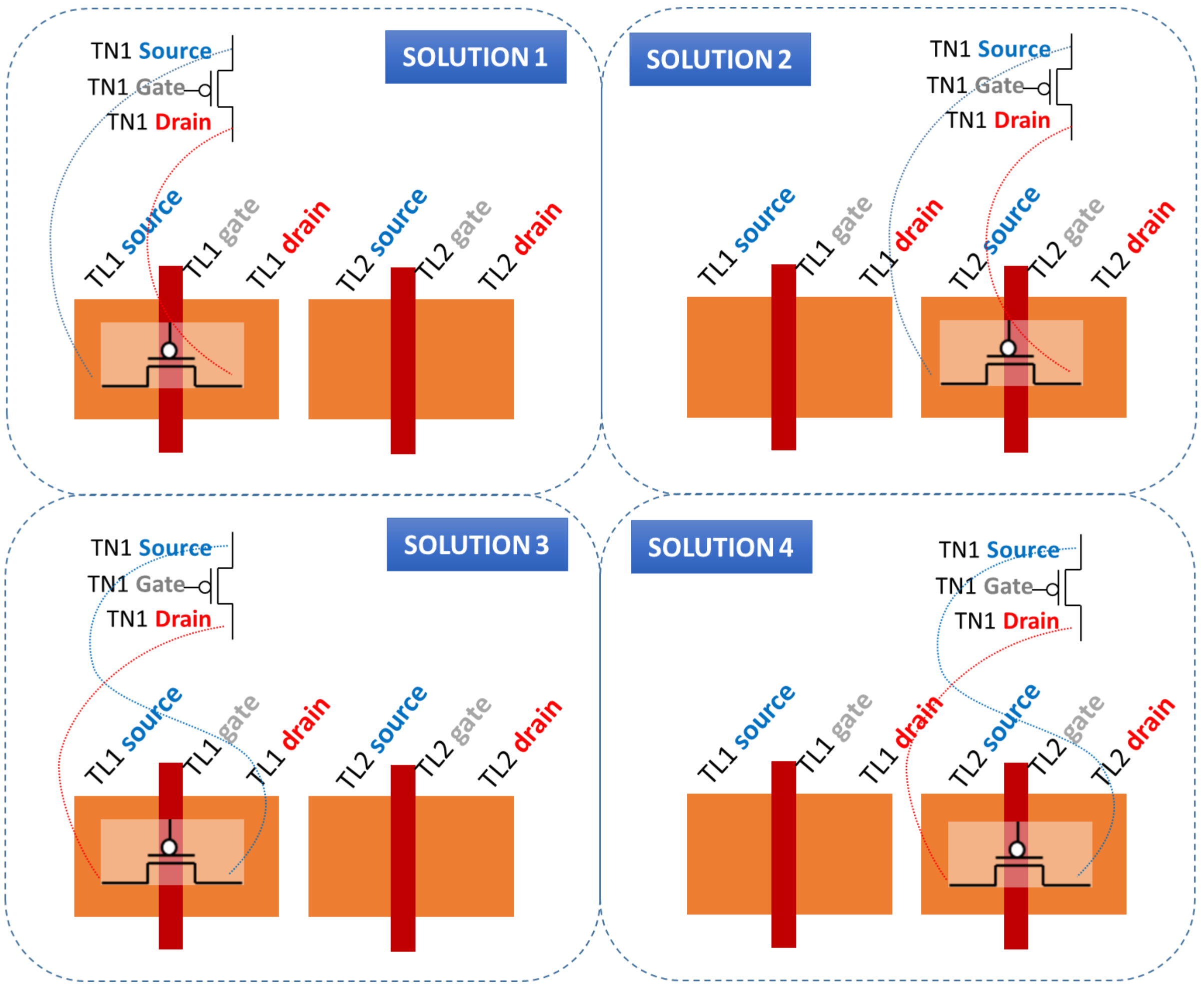
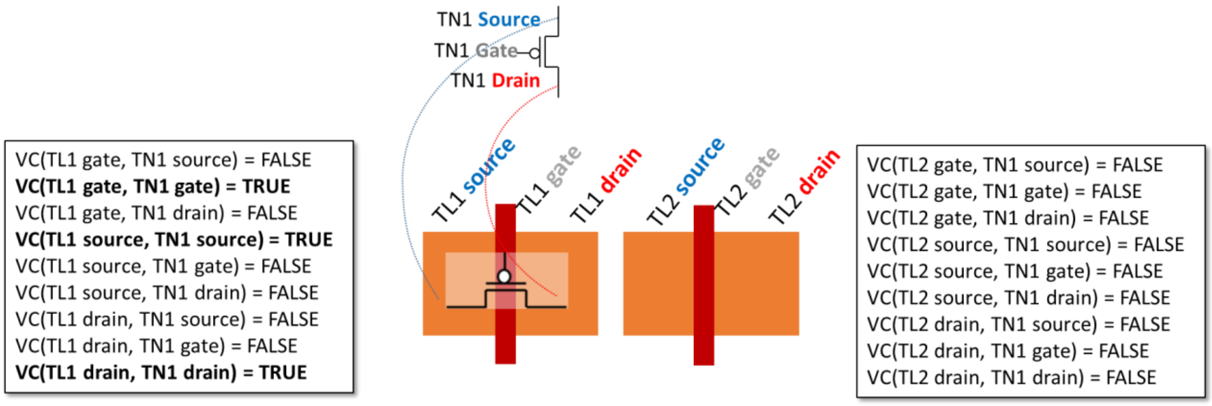


Figure 4.31 - Fitting solution possibilities on given example.



As the VC variable concept is to represent a virtual connection among layout elements and transistor network nodes, for each of the four illustrated solutions, the VC variables would assume different values, depending on where the transistor is "placed over" the layout, as demonstrated in Figure 4.32, in Figure 4.33, in Figure 4.34 and in Figure 4.35.

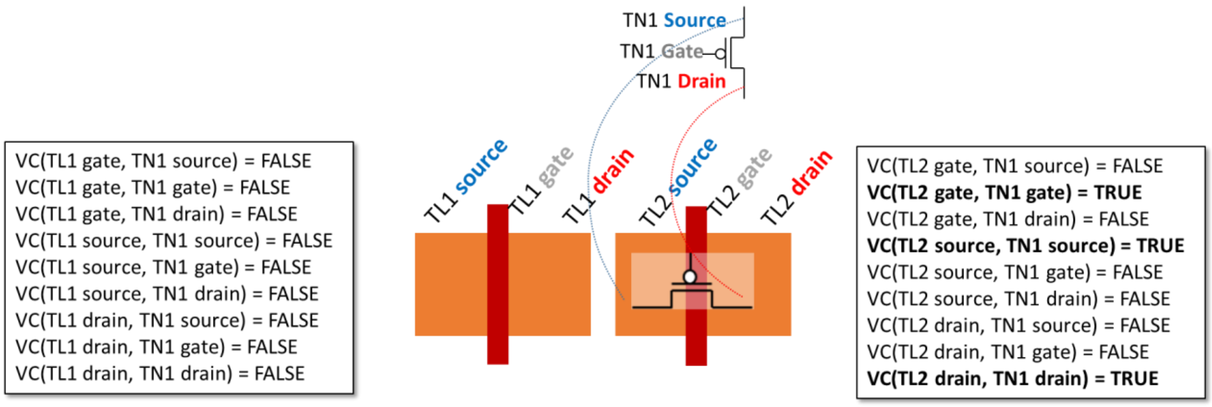
Figure 4.32 - Expected values on VC variable of solution 1.



VC(TL1 gate, TN1 source) = FALSE
VC(TL1 gate, TN1 gate) = TRUE
VC(TL1 gate, TN1 drain) = FALSE
VC(TL1 source, TN1 source) = TRUE
VC(TL1 source, TN1 gate) = FALSE
VC(TL1 source, TN1 drain) = FALSE
VC(TL1 drain, TN1 source) = FALSE
VC(TL1 drain, TN1 gate) = FALSE
VC(TL1 drain, TN1 drain) = TRUE

VC(TL2 gate, TN1 source) = FALSE
VC(TL2 gate, TN1 gate) = FALSE
VC(TL2 gate, TN1 drain) = FALSE
VC(TL2 source, TN1 source) = FALSE
VC(TL2 source, TN1 gate) = FALSE
VC(TL2 source, TN1 drain) = FALSE
VC(TL2 drain, TN1 source) = FALSE
VC(TL2 drain, TN1 gate) = FALSE
VC(TL2 drain, TN1 drain) = FALSE

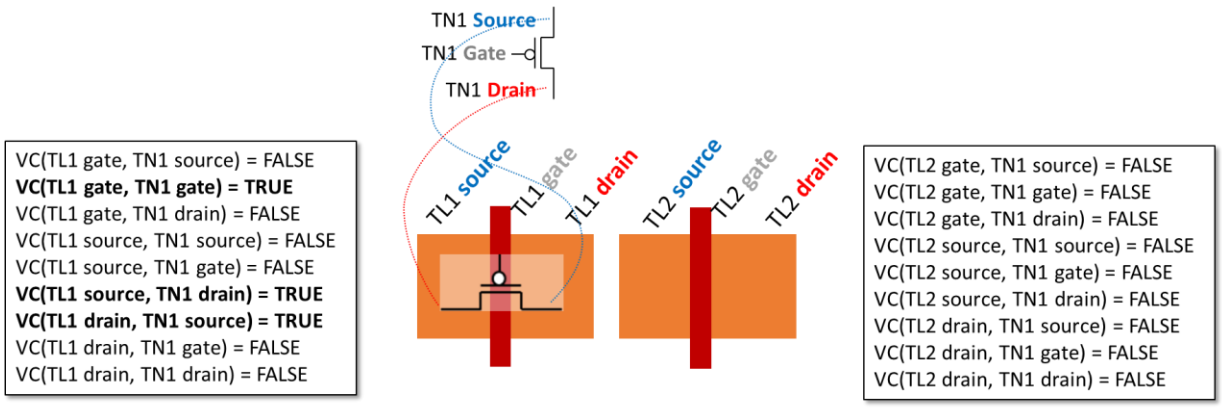
Figure 4.33 - Expected values on VC variable of solution 2.



VC(TL1 gate, TN1 source) = FALSE
VC(TL1 gate, TN1 gate) = FALSE
VC(TL1 gate, TN1 drain) = FALSE
VC(TL1 source, TN1 source) = FALSE
VC(TL1 source, TN1 gate) = FALSE
VC(TL1 source, TN1 drain) = FALSE
VC(TL1 drain, TN1 source) = FALSE
VC(TL1 drain, TN1 gate) = FALSE
VC(TL1 drain, TN1 drain) = FALSE

VC(TL2 gate, TN1 source) = FALSE
VC(TL2 gate, TN1 gate) = TRUE
VC(TL2 gate, TN1 drain) = FALSE
VC(TL2 source, TN1 source) = TRUE
VC(TL2 source, TN1 gate) = FALSE
VC(TL2 source, TN1 drain) = FALSE
VC(TL2 drain, TN1 source) = FALSE
VC(TL2 drain, TN1 gate) = FALSE
VC(TL2 drain, TN1 drain) = TRUE

Figure 4.34 - Expected values on VC variable of solution 3.

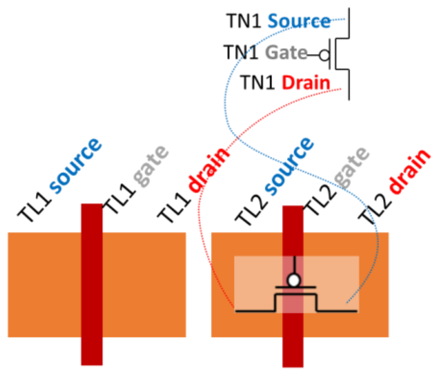


VC(TL1 gate, TN1 source) = FALSE
VC(TL1 gate, TN1 gate) = TRUE
VC(TL1 gate, TN1 drain) = FALSE
VC(TL1 source, TN1 source) = FALSE
VC(TL1 source, TN1 gate) = FALSE
VC(TL1 source, TN1 drain) = TRUE
VC(TL1 drain, TN1 source) = TRUE
VC(TL1 drain, TN1 gate) = FALSE
VC(TL1 drain, TN1 drain) = FALSE

VC(TL2 gate, TN1 source) = FALSE
VC(TL2 gate, TN1 gate) = FALSE
VC(TL2 gate, TN1 drain) = FALSE
VC(TL2 source, TN1 source) = FALSE
VC(TL2 source, TN1 gate) = FALSE
VC(TL2 source, TN1 drain) = FALSE
VC(TL2 drain, TN1 source) = FALSE
VC(TL2 drain, TN1 gate) = FALSE
VC(TL2 drain, TN1 drain) = FALSE

Figure 4.35 - Expected values on VC variable of solution 4.

VC(TL1 gate, TN1 source) = FALSE
VC(TL1 gate, TN1 gate) = FALSE
VC(TL1 gate, TN1 drain) = FALSE
VC(TL1 source, TN1 source) = FALSE
VC(TL1 source, TN1 gate) = FALSE
VC(TL1 source, TN1 drain) = FALSE
VC(TL1 drain, TN1 source) = FALSE
VC(TL1 drain, TN1 gate) = FALSE
VC(TL1 drain, TN1 drain) = FALSE

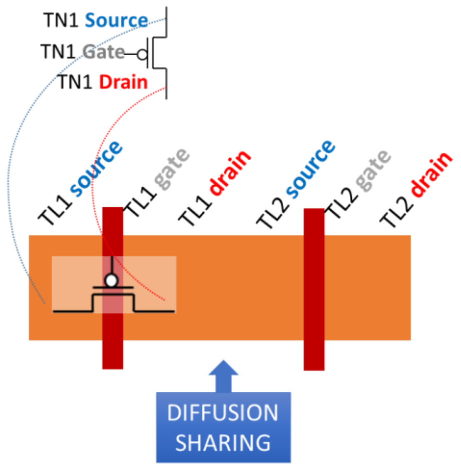


VC(TL2 gate, TN1 source) = FALSE
VC(TL2 gate, TN1 gate) = TRUE
VC(TL2 gate, TN1 drain) = FALSE
VC(TL2 source, TN1 source) = FALSE
VC(TL2 source, TN1 gate) = FALSE
VC(TL2 source, TN1 drain) = TRUE
VC(TL2 drain, TN1 source) = TRUE
VC(TL2 drain, TN1 gate) = FALSE
VC(TL2 drain, TN1 drain) = FALSE

Advancing in the elaboration of "virtual connections" concept, it seems natural that, if two layout elements are connected by a conductive path, then the virtual connections of the first element should "spread" to the second one. Figure 4.36 exemplifies this concept refinement, in which a transistors source is connected to another transistor drain through diffusion sharing, making the TRUE value in VC "to be conducted" among those layout elements.

Figure 4.36 - Diffusion sharing case: TRUE value in VC "spreads" over connected elements.

VC(TL1 gate, TN1 source) = FALSE
VC(TL1 gate, TN1 gate) = TRUE
VC(TL1 gate, TN1 drain) = FALSE
VC(TL1 source, TN1 source) = TRUE
VC(TL1 source, TN1 gate) = FALSE
VC(TL1 source, TN1 drain) = FALSE
VC(TL1 drain, TN1 source) = FALSE
VC(TL1 drain, TN1 gate) = FALSE
VC(TL1 drain, TN1 drain) = TRUE



VC(TL2 gate, TN1 source) = FALSE
VC(TL2 gate, TN1 gate) = FALSE
VC(TL2 gate, TN1 drain) = FALSE
VC(TL2 source, TN1 source) = FALSE
VC(TL2 source, TN1 gate) = FALSE
VC(TL2 source, TN1 drain) = TRUE
VC(TL2 drain, TN1 source) = FALSE
VC(TL2 drain, TN1 gate) = FALSE
VC(TL2 drain, TN1 drain) = FALSE

The rules and mechanisms that will define the TRUE or FALSE value in each VC variable will be formalized gradually in this text, and are the basis of the entire SAT formulation proposed in this thesis. However, with the partial concept of virtual connection already presented, it is possible to start formulating how transistor networks connections would be ensured or avoided in terms of VC variable values. The assumption imposed in this model is that if a layout element present the value TRUE simultaneously in two VC variables, then some mechanism in the layout (diffusion abutment, metal lines, or other, all will be detailed later on) ensured the connection of these two network nodes. Naturally, for a transistor network to be implemented in a layout, all connections among transistor network nodes must be ensured in the layout, and all absent node connections must be avoided, leading us to the first set of rules for this model:

Rule CONN1: If two network transistor nodes ‘n_1’ and ‘n_2’ are connected (present the same name in the input transistor network textual description) then for at least one layout element ‘e’ both VC(e, n_1) and VC(e, n_2) must be TRUE.

Assuming the amount of ‘k’ layout elements, the enforcing clause for CONN1 rule follows this format: $(VC(e_1, n_1) \land VC(e_1, n_2)) \lor (VC(e_2, n_1) \land VC(e_2, n_2)) \lor \ldots \lor (VC(e_{k-1}, n_1) \land VC(e_{k-1}, n_2)) \lor (VC(e_k, n_1) \land VC(e_k, n_2))$

Rule CONN2: If two network transistor nodes ‘n_1’ and ‘n_2’ are not connected (does not present the same name in the input transistor network textual description), then for **every** layout element ‘e’ at most one of VC(e, n_1) and VC(e, n_2) can be true, never both.

Assuming the amount of ‘k’ layout elements, the enforcing clause for CONN2 rule follows this format: $\neg(VC(e_1, n_1) \land VC(e_1, n_2)) \land \neg(VC(e_2, n_1) \land VC(e_2, n_2)) \land \ldots \land \neg(VC(e_{k-1}, n_1) \land VC(e_{k-1}, n_2)) \land \neg(VC(e_k, n_1) \land VC(e_k, n_2))$

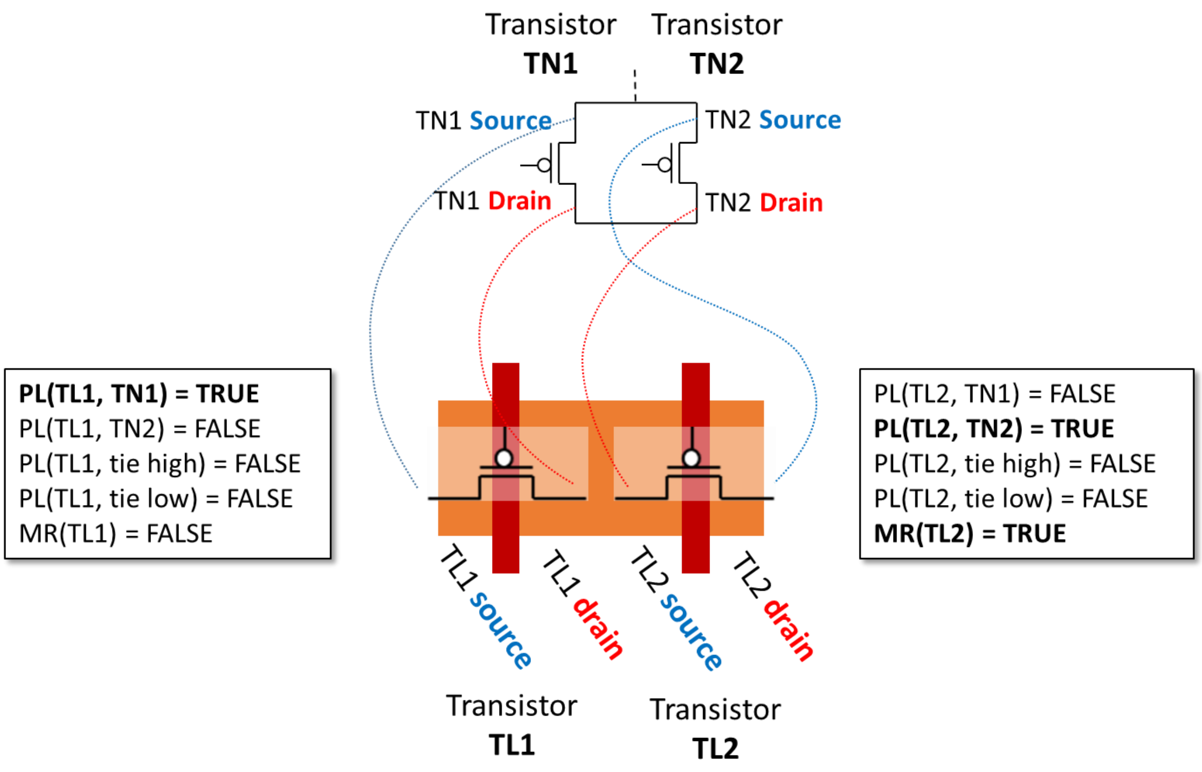
Notice that the rule CONN1 ensures the presence of desired node connections while the rule CONN2 ensures the avoidance of undesired connections. These rules and Boolean equations form the top level of the developed SAT model. However, the model is yet not complete as VC(e, n) variable should not be freely attributed to TRUE and FALSE, and as roughly induced in previous examples, such attributions must depend on several layout aspects, as the transistors placement and presence of conductive paths (through diffusion sharing, always-on transistors, metal lines and contacts/vias), turning imperative another set of rules. This second set of rules are layer-specific, and can be divided into “direct” and “indirect”. However, before presenting the rules, we must define their variables:

Variable PL(tl,tn): indicates if the layout transistor ‘tl’ is representing the network transistor ‘tn’. It can be interpreted as a sort of placement variable, informing if transistor ‘tn’ is virtually "placed over" the transistor 'tl'. The 'tn' field may assume two extra values beyond the identification of each transistor on transistor network, which are: 'tie high' and 'tie low'.

Variable MR(tl): indicates if the layout transistor 'tl' is "mirrored", *i.e.*, assuming a case in which the transistor 'tl' is representing the transistor network 'tn' then the source of the transistor 'tl' is actually representing the drain of transistor 'tn' instead of its source.

Figure 4.37 represents a visual interpretation for the variables PL and MR.

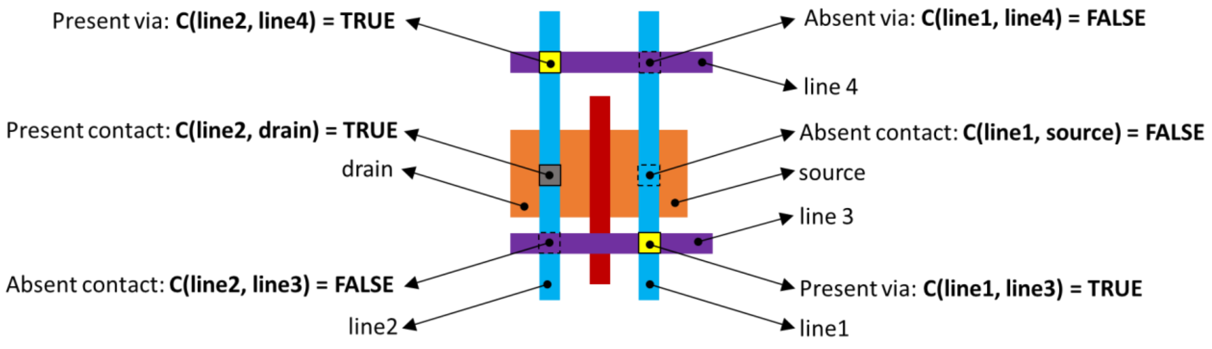
Figure 4.37 - Variables PL and MR attribution and its visual "placement" interpretation.



Variable C(e$_1$,e$_2$): These variables represent the presence of a connector (via or contact) among two layout elements 'e$_1$' and 'e$_2$'. The candidate layout elements for these variables must respect the restrictions of a traditional CMOS layout (*e.g.* vias can only connect metals of adjacent levels, contacts can only connect metal 1 to lower layers).

Figure 4.38 illustrates an example of expected values in variable C. Observe that potential places for vias and contacts makes C false if the contact/via is absent, while the placed contacts and vias causes the TRUE values in C variable.

Figure 4.38 - Variable C expected values example.


Present via: **C(line2, line4) = TRUE**
Absent via: **C(line1, line4) = FALSE**
line 4
Present contact: **C(line2, drain) = TRUE**
Absent contact: **C(line1, source) = FALSE**
drain
source
line 3
Absent contact: **C(line2, line3) = FALSE**
Present via: **C(line1, line3) = TRUE**
line2
line1

With the presented variables, it is possible to define a set of "direct" rules, or rules based on what we called in this text the "direct conditions" of a virtual connection.

Rule DIRPOLY1: Considering a layout element 'e' that is the gate of a transistor on the given VCSA block template layout. VC(e, n) is FALSE for any transistor network node 'n' that is not a transistor gate.

Rule DIRPOLY2: Considering a layout element 'e' that is the gate of a transistor 'tl' on the given VCSA block template layout. If the node 'n' is the gate of transistor 'tn' in the given transistor network, then VC(e, n) is TRUE if PL(tl, tn) is TRUE.

In a less formal definition, this last rule can be interpreted as: if the transistor of the transistor network is "placed over" the layout transistor then their gates are virtually connected. Observe that this rule is not exclusive, it does not list the only condition to make the referred variable TRUE, but one of these conditions.

Source and drain diffusions behave similarly, with extra care for the transistor "mirroring" variables, as follows:

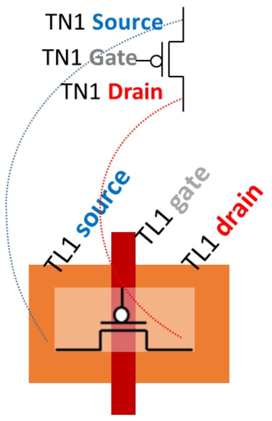
Rule DIRDIFF1: Considering a layout element 'e' that is the source or drain of the transistor 'tl' in the given VCSA block template layout. VC(e, n) is FALSE for any node 'n' that is not a transistor source or drain of a given transistor network.

Rule DIRDIFF2: Considering a layout element ‘e’ that is the source of a transistor ‘tl’ of a given VCSA block template layout. If the node ‘n’ is the source of the transistor ‘tn’ of the given transistor network, then VC(e, n) is TRUE if PL(tl, tn) is TRUE and MR(tl) is FALSE. If the node ‘n’ is the drain of a transistor ‘tn’ of a given transistor network, then VC(e, n) is TRUE if PL(tl, tn) is TRUE and MR(tl) is TRUE.

Rule DIRDIFF3: Considering a layout element ‘e’ that is the drain of a transistor ‘tl’ of a given VCSA block template layout. If the node ‘n’ is the source of the transistor ‘tn’ of a given transistor network, then VC(e, n) is TRUE if PL(tl, tn) is TRUE and MR(tl) is TRUE. If the node ‘n’ is the drain of a transistor ‘tn’ of a given transistor network, then VC(e, n) is TRUE if PL(tl, tn) is TRUE and MR(tl) is FALSE.

Figure 4.39 illustrates the presented diffusion-related rules.

Figure 4.39 - Example of poly and diffusion rules’ application.



Assume as example:

PL(TL1, TN1) = TRUE
MR(TL1) = FALSE

Then:

VC(TL1 gate, TN1 source) = FALSE	[by rule DIRPOLY1]
VC(TL1 gate, TN1 gate) = PL(TL1, TN1) = **TRUE**	[by rule DIRPOLY2]
VC(TL1 gate, TN1 drain) = FALSE	[by rule DIRPOLY1]
VC(TL1 source, TN1 source) = PL(TL1, TN1) & !MR(TL1) = **TRUE**	[by rule DIRDIFF2]
VC(TL1 source, TN1 gate) = FALSE	[by rule DIRDIFF1]
VC(TL1 source, TN1 drain) = PL(TL1, TN1) & MR(TL1) = FALSE	[by rule DIRDIFF3]
VC(TL1 drain, TN1 source) = PL(TL1, TN1) & MR(TL1) = FALSE	[by rule DIRDIFF3]
VC(TL1 drain, TN1 gate) = FALSE	[by rule DIRDIFF1]
VC(TL1 drain, TN1 drain) = PL(TL1, TN1) & !MR(TL1) = **TRUE**	[by rule DIRDIFF2]

Rule DIRMETAL: Considering a layout element ‘e_H’ that is a metal line and a layout element ‘e_L’ a layout element in the immediate lower layer than ‘e_H’ layer, then VC(e_H,n) is TRUE if C(e_H, e_L) is TRUE and VC(e_L,n) is TRUE.

Observe that the DIRMETAL rule induces the propagation of the TRUE virtual connection (VC) variables from lower to higher layout layers, but not on the contrary way (from the higher to the lower ones). This property will be called as “**unidirectional upward propagation**”, and will be used later again on this text. Figure 4.40 highlights this concept.

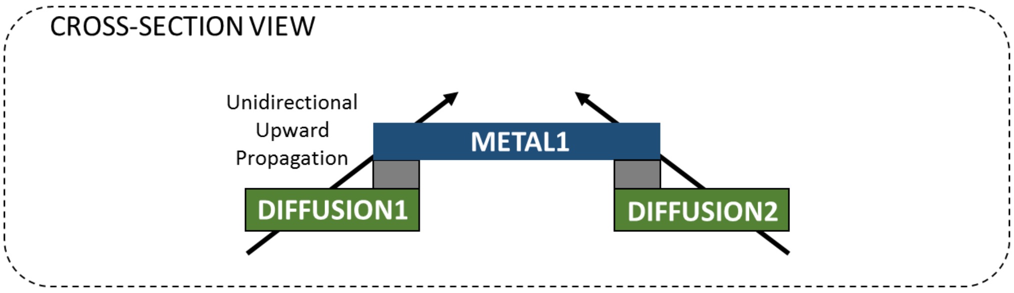
Figure 4.40 - Unidirectional upward propagation concept illustration.

Assume as example:

C(METAL1, DIFFUSION1) = TRUE C(METAL1, DIFFUSION2) = TRUE	Means that the contacts among metal line and diffusions are present

And also:

VC(DIFFUSION1, na) = TRUE	Means that DIFFUSION1 is virtually connected to an arbitrary node ‘na’ due some other direct condition



Then:

VC(METAL1, na) = VC(DIFFUSION1, na) | VC(DIFFUSION2, na) = TRUE [by rule DIRMETAL]

Forming the Boolean enforcing clauses from the direct rules description, as well as from the next rules of this chapter, it is straightforward (simply translating “and” / “or” words on rules’ descriptions to their equivalent logic operators) and so omitted here.

The “indirect” rules, or the rules based on what is called herein “indirect” conditions, consider a scenario in which a layout element is connected to another layout element of the same kind (same layout layer) that, in its turn, is connected to the target transistor network node through its direct conditions.

The concept in the previous paragraph brings the risk of creating “loop dependencies” that must be avoided as they constitute logical error in any Boolean model. An example of loop dependency is: layout lines ‘la’ and ‘lb’ are connected by a via; by the indirect condition concept, we may assume for a node ‘n’ that VC(la,n) is TRUE if VC(lb,n) is TRUE; similarly, VC(lb,n) is TRUE if VC(lb,n) is TRUE; in this case, a loop is created, in which both variables may assume the value TRUE, supporting each other independently of any other conditions. In order to avoid this error, observe how the next rules take special care for making VC variables dependent on the direct conditions of other layout elements, and not directly on their VC variable value.

Rule INDIR1: Considering a layout element 'e_1' that is the gate of a layout transistor 'tl_1' and a layout element 'e_2' that is the gate of a layout transistor 'tl_2', in a case that both 'e_1' and 'e_2' share the same (poly) line. If the node 'n' is the gate of the transistor 'tn' of the given transistors network, then $VC(e_1,n)$ is TRUE if $PL(tl_2,tn)$ is TRUE.

If two layout elements of the same layer present a connection path that passes through higher layers, then this connection can be ignored for determining the rules based on indirect conditions of that connected elements. This simplification is possible due to the already mentioned "unidirectional upward propagation" property, which will make the higher layer to always present a superset of the TRUE values of lower layers VC variables, satisfying the CONN1 rule by itself.

It means that, when building the rules based on indirect conditions, only underneath connections among layout elements of each layer must be taken into account, and this is why rule INDIR1 is so simple: it assumes that there is no way to connect two poly gates using lower layers (diffusion). Thus, two gates can only be connected without using higher layers (metal) if they share the same poly line. However, for diffusion and metal layers the complexity is increased: diffusions also cannot present underneath connections, but they can connect to each other through abutment or trough always-on transistors, possibly forming large connection chains; meanwhile, metal lines may present several possibilities for connection routes using lower layers.

To take in consideration all possible underneath routes for connecting two metal line segments as Boolean variables in the model, it arises again the problem of turning the model prohibitively large. If a routing scheme is built just for defining which layout lines are connected, and then integrate that to the main model the complexity also becomes a bottleneck. In order to avoid the stated problems, a new scheme for diffusions and metal lines were built, and it requires the understanding of the concepts of "1-step neighbors" and "virtual layout layers" beyond the already explained "unidirectional upward propagation".

Definition of 1-step neighbors: this definition is layer-dependent.

- If two layout elements are both diffusions and are connected by abutment they are considered 1-step neighbors with step condition always TRUE.
- If two layout elements are both diffusions, and they are the source and drain of the same layout transistor 'tl' , then they are considered 1-step neighbors with step-condition of 'tl' being always-on, *i.e.*, tied-high if N-type and tied-low if P-type.

- Two metal lines of the same layout layer are considered 1-step neighbors if they are underneath connected by the same layout element, *i.e.*, the step condition is both being connected to the same layout element of the immediate lower layer.

Definition of virtual layout layers: copies of the original layout layers, presenting the same amount of layout elements, used for internal manipulation of variables in the SAT model. The virtual layout layers behave as if placed below the original layer and are in maximum number equal to the number of layout elements on the original layer that they imitate.

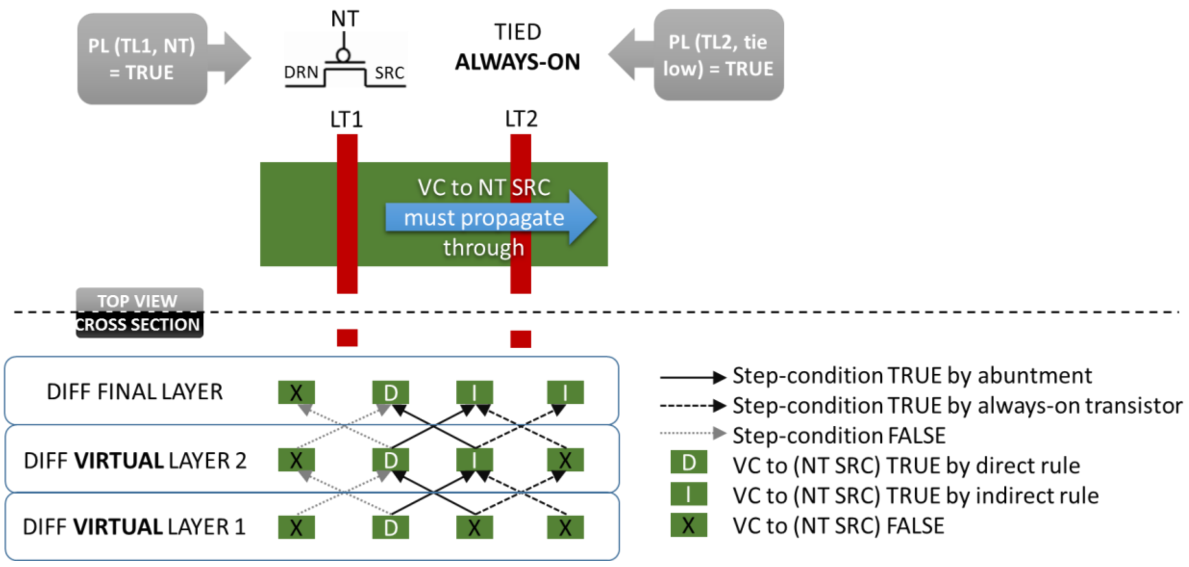
The main idea on virtual layers concept is to emulate an iterative propagation of TRUE values in VC variables. For example, if two metal lines of the same level ‘la’ and ‘lb’ have an underneath connection path and one of the lines present a TRUE value in a VC variable then, by the indirect conditions, the second line should present this variable TRUE too. In other words, if the line ‘la’ is virtually connected to the network node ‘n’ and the line ‘la’ is connected, in the layout, to the line ‘lb’, then ‘lb’ must also present a virtual connection to the node ‘n’. However, it was already stated that it may become not feasible to find/use all paths among two layout elements, but it is not needed to find all such paths if, someway, there is an iterative propagation of the desired information from ‘la’ to all its 1-step neighbors, and from them to their 1-step neighbors, and so on, until the line ‘lb’ and all other connected line segments are reached. A difficulty, in this case, is that there is no “for” or “while” loops to implement such iteration in a SAT model, only Boolean variables and equations. This way, a solution is to replicate the number of variables in the number of iterations needed, forming ‘k’ groups of semi-identical (related but uniquely identified) variables. Then, Boolean equations may rule the values transferring from the ‘(k-1)’th group to the ‘k’th group, in a way that the final group presents the same values that an iterative algorithm would cause. Figure 4.41 illustrates an example based on this explanation.

With the developed concepts of “1-step neighbors” and “virtual layers” it is possible to state the following rules:

Rule INDIR2: Considering two layout element ‘e_1’ and ‘e_2’ that are both diffusions or metal lines of the same layer. Considering also that ‘e_{1_k}’ and ‘e_{2_k}’ are the elements corresponding to ‘e_1’ and ‘e_2’ on the kth virtual layer. For any network node ‘n’, VC(e_{1_k},n) is

TRUE if ‘e_1’ and ‘e_2’ are 1-step neighbors and their step condition is TRUE and VC($e_{2\_(k-1)}$,n) is TRUE.

Figure 4.41 - VC variable TRUE value propagating upward through virtual layers.



The rules INDIR1 and INDIR2 depend on PL and C variables. As C represents vias and contacts, it can be freely attributed to TRUE or FALSE, representing if that connector is present on the layout or not (excepting for VCSA block templates that fix the contacts layer, in which all contacts are then TRUE). PL variables, in turn, must have rules to avoid cases of overlapping or missing transistors, and for this reason the third set of rules is presented:

Rule PLACE1: Considering ‘tl’ as a transistor in the VCSA block template layout. PL(tl,tn) must be TRUE for at least and at most one value of ‘tn’. The set of possible ‘tn’ values comprises all transistors in the transistor network plus two extra values: tie high and tie low.

Rule PLACE2: Considering ‘tn’ as a transistor of the transistor network. PL(tl,tn) must be TRUE for at least and at most one value of ‘tl’. The set of possible ‘tl’ values comprises all transistors in the VCSA block template layout.

With these two transistors placement or ordering rules, every transistor of the transistors network will be represented by exactly one transistor in the VCSA block template layout, while the exceeding layout transistors (if there is any) assume ‘tie high’ or ‘tie low’ values.

Another missing rule refers to layout transistors that share the same poly line, as they must represent transistors on the network that are controlled by the same signal:

Rule PLACE3: Considering ‘tl_1’ and ‘tl_2’ as transistors that share the same gate poly line in the VCSA block template layout. If PL(tl_1,tn_1) is TRUE then PL(tl_2,tn_2) can only be true if ‘tn_1’ and ‘tn_2’ are controlled by the same signal in the input transistors network.

The final rules are related to supply lines. It is usual to have pre-defined which metal lines in a VCSA block template layout will present power supply and ground nodes. A virtual node for power supply and another for ground are created in the software internal representation of the transistors network, they will be called ‘intVdd’ and ‘intGnd’, and virtual connections between the layout elements and these nodes are ruled by:

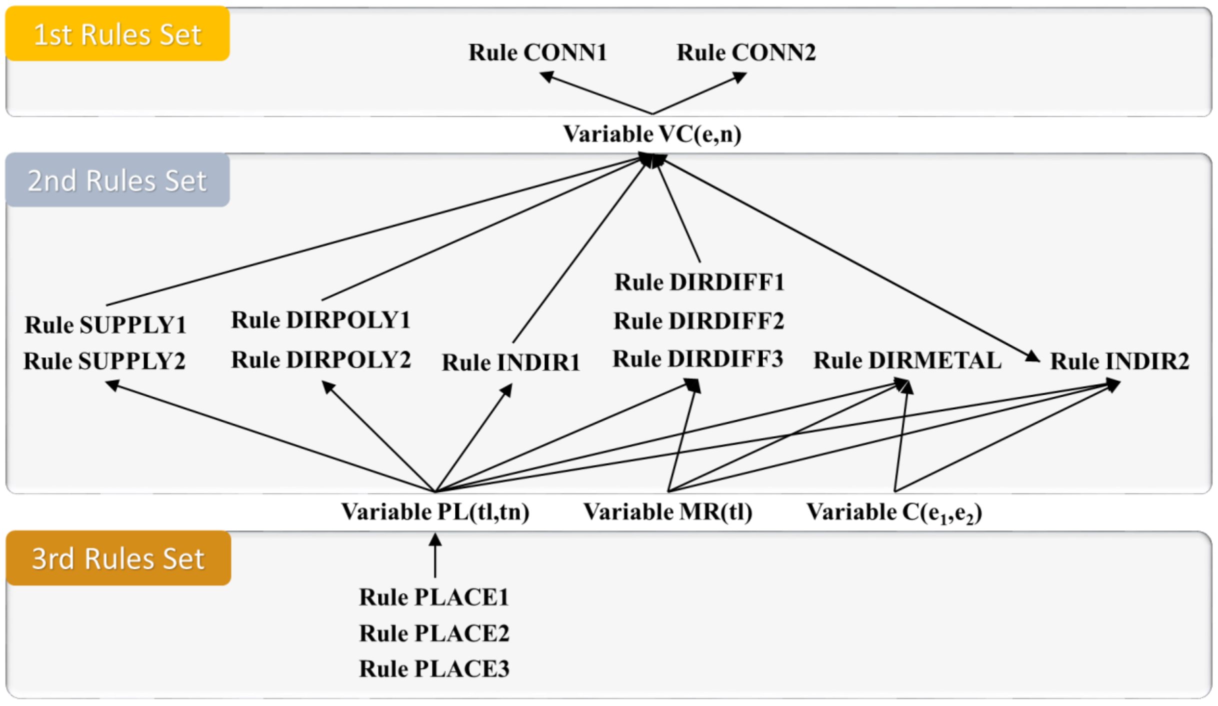
Rule SUPPLY1: VC(e,intVdd) is TRUE if ‘e’ is a pre-defined metal line containing power supply or is the gate of a transistor ‘t’ for which PL(t,tie high) is TRUE.

Rule SUPPLY2: VC(e,intGnd) is TRUE if ‘e’ is a pre-defined metal line containing ground or is the gate of a transistor ‘t’ for which PL(t,tie low) is TRUE.

The propagation of the TRUE values created by the conditions in SUPPLY1 and SUPPLY2 rules are automatically performed by the previous rules. And the connection of the transistor network nodes with specific pre-defined power supply node names to these created virtual nodes are ensured by the rule CONN1.

Figure 4.42 depicts the hierarchy scheme comprising all the stated variables, rules, and their dependencies.

Figure 4.42 - Hierarchy and dependencies of variables and rules of developed SAT model.



5 EXPERIMENTAL RESULTS

The current implementation of the proposed environment includes a software developed in Java language that receives two inputs, a VCSA block template description and a transistors network, and generates a textual file containing the SAT model that, when provided to a SAT solver, is able to decide if the given transistor network can be implemented in a given VCSA template. The software implementation contains currently 17 classes and their diagram can be found in Appendix A. The VCSA template layout is described in textual format, following a syntax developed for this specific application, similar to a SPICE description but extended to include metal lines. Appendix B demonstrates several VCSA templates described in this textual format. The input transistor network can be in SPICE format, although some of its information (as transistors widths) are not currently used in the developed application.

- Functional capacity evaluation of VCSA templates

The following results are generated using the developed SAT model to cross a set of transistor networks against different VCSA templates. The chosen transistor networks set belongs to the Nangate FreePDK Open Cell Library (FREEPDK, 2012), while the chosen VCSA templates are the ones described in Chapter 2: a VCC variation, a VCLB variation and a VCTA variation. After the SAT model for each transistor network and VCSA template combination is created, it is provided to the SAT solver YICES (YICES, 2015), reaching the depicted result at each cell in Table 5.6[2].

It must be taken into account that 76 of the 127 networks are eliminated by simple transistors counting. Regarding the networks with valid transistors count, Table 5.6 evidences that, among the single block templates[3], VCTA6T is able to express the largest subset, including networks like XOR2 and XNOR2, and tristate inverter. The gain in area that this extra logical capacity achieves in the final circuit may depend on many variables (as the used mapping tool, the circuit constraints and its characteristics) and is beyond the scope of this thesis. However, this investigation is the natural progress of this work, as stated in future

[2] The largest networks in this library are omitted as they cannot fit the used templates by simple transistors count.

[3] The template in last table column (2x5VCC) is composed by two 5VCC instances, as detailed later on this chapter.

work section in the last chapter. Table 5.6 also makes evident that, for this set of transistor networks, SLVC5P diffusion breaks add no extra logical capacity when compared to 5VCC.

Another evidence on these results is the VCTA6T routing scheme acting as a bottleneck for its logic capability, as AND4 and OR4 fit in both 5VCC and VCLB5P, but not in VCTA, even this last one presenting more transistors.

Table 5.6 - Library transistor networks SAT results for several VCSA templates.

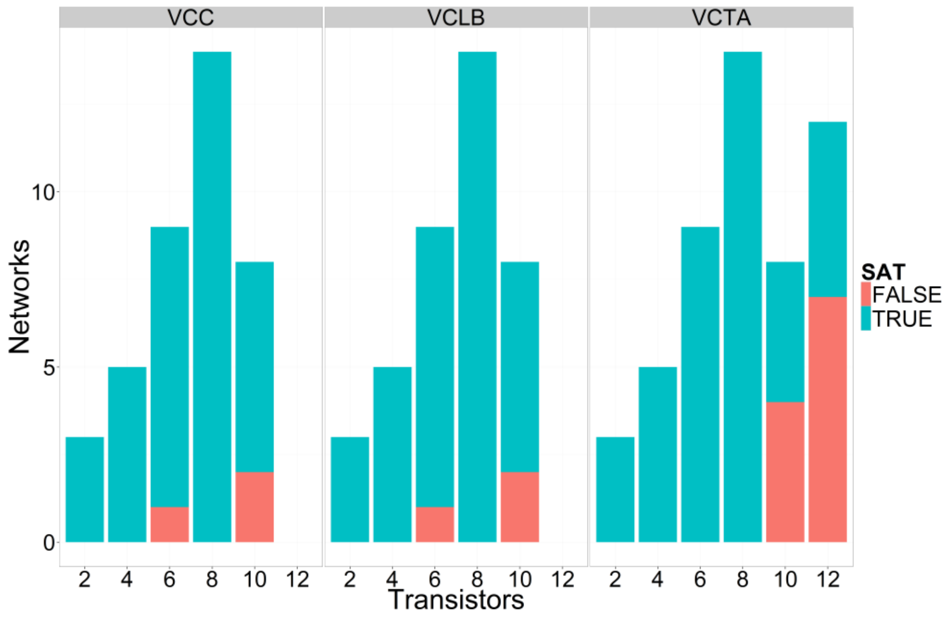
	VCTA6T	**SLVC5P**	**5VCC**	**2x5VCC**
AND2_X1	sat	sat	sat	sat
AND2_X2	sat	sat	sat	sat
AND2_X4	unsat	unsat	unsat	sat
AND3_X1	sat	sat	sat	sat
AND3_X2	sat	sat	sat	sat
AND3_X4	unsat	unsat	unsat	unsat
AND4_X1	unsat	sat	sat	sat
AND4_X2	unsat	unsat	unsat	sat
AND4_X4	unsat	unsat	unsat	unsat
AOI211_X1	sat	sat	sat	sat
AOI211_X2	unsat	unsat	unsat	sat
AOI211_X4	unsat	unsat	unsat	sat
AOI21_X1	sat	sat	sat	sat
AOI21_X2	sat	unsat	unsat	sat
AOI21_X4	unsat	unsat	unsat	unsat
AOI221_X1	unsat	sat	sat	sat
AOI221_X2	unsat	unsat	unsat	unsat
AOI221_X4	unsat	unsat	unsat	unsat
AOI222_X1	unsat	unsat	unsat	sat
AOI222_X2	unsat	unsat	unsat	unsat
AOI222_X4	unsat	unsat	unsat	unsat
AOI22_X1	sat	sat	sat	sat
AOI22_X2	unsat	unsat	unsat	sat
BUF_X1	sat	sat	sat	sat
BUF_X2	sat	sat	sat	sat
BUF_X4	sat	unsat	unsat	sat
BUF_X8	unsat	unsat	unsat	unsat
CLKBUF_X1	sat	sat	sat	sat
CLKBUF_X2	sat	sat	sat	sat
CLKBUF_X3	sat	sat	sat	sat
CLKGATETST_X1	unsat	unsat	unsat	unsat
CLKGATE_X1	unsat	unsat	unsat	unsat
CLKGATE_X2	unsat	unsat	unsat	unsat
DLH_X1	unsat	unsat	unsat	unsat

DLH_X2	unsat	unsat	unsat	unsat
DLL_X1	unsat	unsat	unsat	unsat
DLL_X2	unsat	unsat	unsat	unsat
HA_X1	unsat	unsat	unsat	unsat
INV_X1	sat	sat	sat	sat
INV_X2	sat	sat	sat	sat
INV_X4	sat	sat	sat	sat
INV_X8	unsat	unsat	unsat	sat
LOGIC0_X1	sat	sat	sat	sat
LOGIC1_X1	sat	sat	sat	sat
MUX2_X1	unsat	unsat	unsat	sat
MUX2_X2	unsat	unsat	unsat	sat
NAND2_X1	sat	sat	sat	sat
NAND2_X2	sat	sat	sat	sat
NAND2_X4	unsat	unsat	unsat	sat
NAND3_X1	sat	sat	sat	sat
NAND3_X2	sat	unsat	unsat	sat
NAND3_X4	unsat	unsat	unsat	unsat
NAND4_X1	sat	sat	sat	sat
NAND4_X2	unsat	unsat	unsat	sat
NOR2_X1	sat	sat	sat	sat
NOR2_X2	sat	sat	sat	sat
NOR2_X4	unsat	unsat	unsat	sat
NOR3_X1	sat	sat	sat	sat
NOR3_X2	sat	unsat	unsat	sat
NOR3_X4	unsat	unsat	unsat	unsat
NOR4_X1	sat	sat	sat	sat
NOR4_X2	unsat	unsat	unsat	sat
OAI211_X1	sat	sat	sat	sat
OAI211_X2	unsat	unsat	unsat	sat
OAI21_X1	sat	sat	sat	sat
OAI21_X2	sat	unsat	unsat	sat
OAI21_X4	unsat	unsat	unsat	unsat
OAI221_X1	unsat	sat	sat	sat
OAI221_X2	unsat	unsat	unsat	unsat
OAI221_X4	unsat	unsat	unsat	unsat
OAI222_X1	unsat	unsat	unsat	sat
OAI222_X2	unsat	unsat	unsat	unsat
OAI222_X4	unsat	unsat	unsat	unsat
OAI22_X1	sat	sat	sat	sat
OAI22_X2	unsat	unsat	unsat	sat
OAI33_X1	unsat	unsat	unsat	sat
OR2_X1	sat	sat	sat	sat
OR2_X2	sat	sat	sat	sat
OR2_X4	unsat	unsat	unsat	sat
OR3_X1	sat	sat	sat	sat

OR3_X2	sat	sat	sat	sat
OR3_X4	unsat	unsat	unsat	unsat
OR4_X1	unsat	sat	sat	sat
OR4_X2	unsat	unsat	unsat	sat
OR4_X4	unsat	unsat	unsat	unsat
TBUF_X1	unsat	unsat	unsat	unsat
TBUF_X2	unsat	unsat	unsat	unsat
TBUF_X4	unsat	unsat	unsat	unsat
TINV_X1	sat	unsat	unsat	unsat
TLAT_X1	unsat	unsat	unsat	unsat
XNOR2_X1	sat	unsat	unsat	sat
XNOR2_X2	unsat	unsat	unsat	sat
XOR2_X1	sat	unsat	unsat	sat
XOR2_X2	unsat	unsat	unsat	sat
TOTAL SAT	**40**	**36**	**36**	**64**

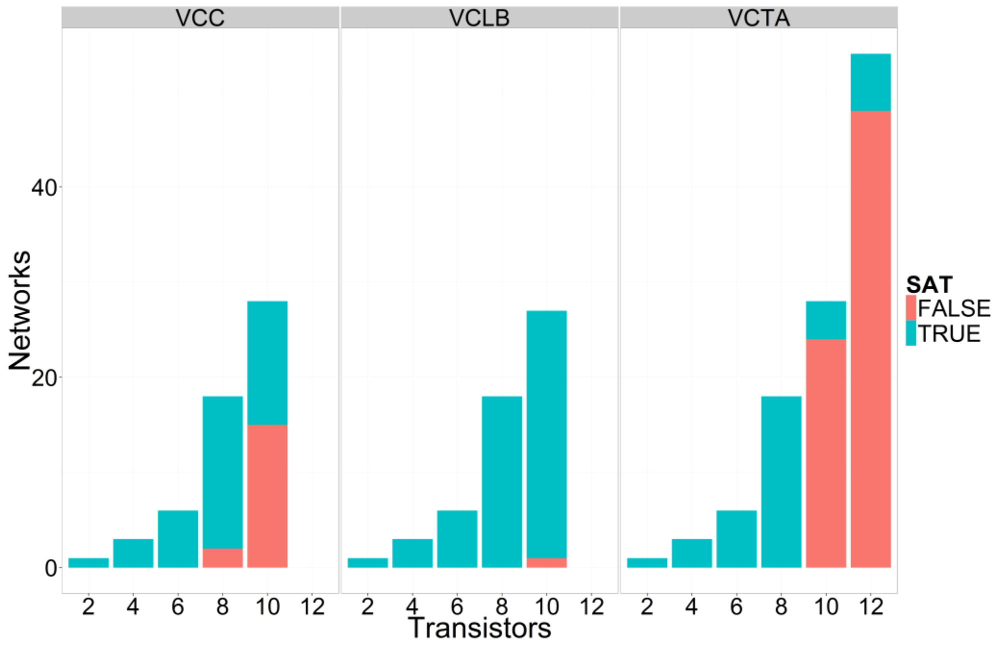
Table 5.6 is summarized in Figure 5.43, where the amount of realizable functions are grouped according to the transistor count in their respective transistor network.

Figure 5.43 - Realizable networks of Nangate Open Cell Library in three VCSA templates.



Expanding this analysis, another transistor networks set were evaluated against the same VCSA templates. This set encompasses all possible functions with up to 4 inputs, excluding the ones that are equivalent by inputs permutation, named P-class[4] (SASAO, 1999). The transistor networks for each of these logic functions (a total of 3982) are created automatically, using a method which focuses on reducing transistors count (POSSANI, 2015). Among the 3982 networks, only 111 presented 12 transistors or less, it means, only this subset must be evaluated on VCTA templates as the largest one present 12 transistors. The results for this second set of transistor networks are depicted in Figure 5.44.

Figure 5.44 - Realizable networks of 4-inputs PClass in three VCSA templates.



For the 4-input P-Class set, VCTA6T is able to express 38 of the evaluated networks, while VCLB5P expresses 54 and 5CC expresses 39 of them. This new set of results refuses the previous achievement of VCTA as the highest logical capacity template, while exposes the dependency of the input functions set for this measurement.

- Results consistency check

The results in Table 5.6 can be manually checked against the described functional capacity on the papers proposing the used templates (RAN, 2006a; PONS, 2010; TUNG,

[4] In this case, “P” stands for “permutation”. Do not mistake this P-class with the algorithms complexity P-class, in which the “P” stands for “polynomial”, that will be mentioned later on this text.

2011). However, only in (RAN, 2006a), the author brings an exhaustive list of its template realizable functions, being many of them absent in our input library, turning this a poor method for validating the developed environment. Manual checks are also error prone, leading us to implement an automated checker.

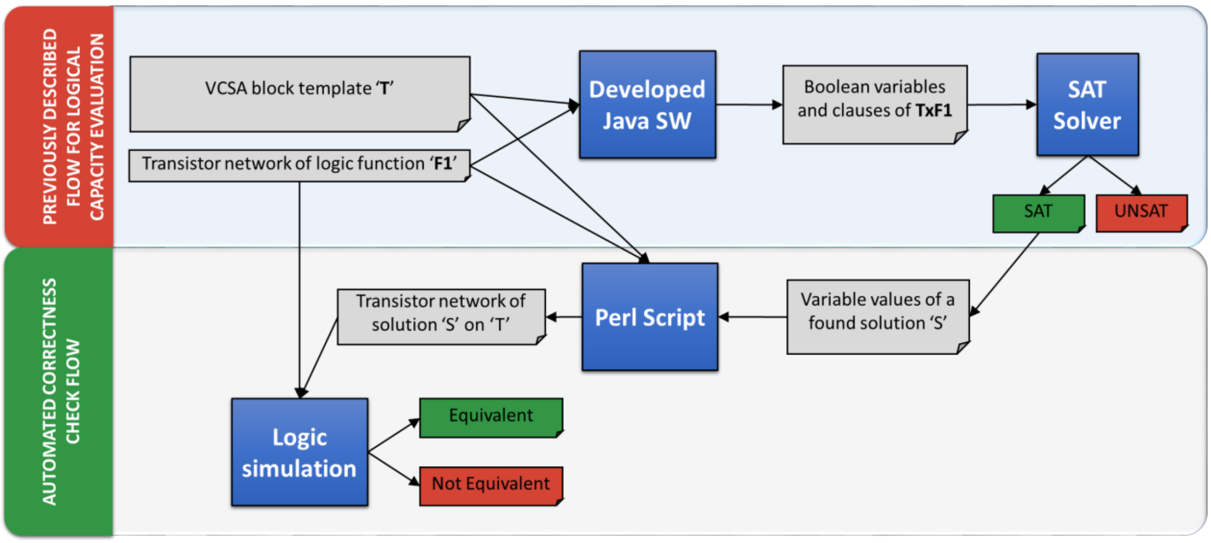
If the SAT solver concludes that a given model is satisfiable, then it can provide at least a solution that satisfies the input model. This solution describes the vias/contacts configuration in the given VCSA block template, as they are Boolean variables in the developed model. With the vias/contacts configuration combined to the input VCSA template description, it is possible to build the reached transistor network by that configuration in the given template.

A software in Perl language was developed to consume the output solution of the SAT solver and to generate the netlist resulting of that solution applied to the given VCSA block template. This software is much simpler than the one that generates the SAT equations to be provided to the SAT solver. A high abstraction level pseudo-code of this software is below.

```
1. Parse input VCSA block template
2. Parse input transistor network
3. Parse SAT solver result
4. Compose a connectivity matrix of layout diffusions based on SAT solver result
5. Compose a connectivity matrix of layout polys based on SAT solver result
6. Compose database with vias and contacts whose value is TRUE in SAT solver result
7. Store pointers to higher and lower layout elements in all these TRUE vias/contacts
8. Attach a transistor network node name to each layout diffusion and poly based on SAT
   solver result
9. Repeat while any change is realized:
   9.1. For each layout layer:
        9.1.1. Spread attached node names to connected layout elements of the same layer
        9.1.2. Spread attached node names to connected layout elements of higher layer
        9.1.3. Spread attached node names to connected layout elements of lower layer
```

An automatic logic equivalence check can be performed between the reached transistor network by configuring vias in the VCSA block against the original given transistor network. In the exercised flow, a logic simulator were used to stress all input possibilities in both these networks, proving their equivalence. Figure 5.45 allows better understanding of the described check process.

Figure 5.45 - Implemented flow for checking correctness of reached SAT results.

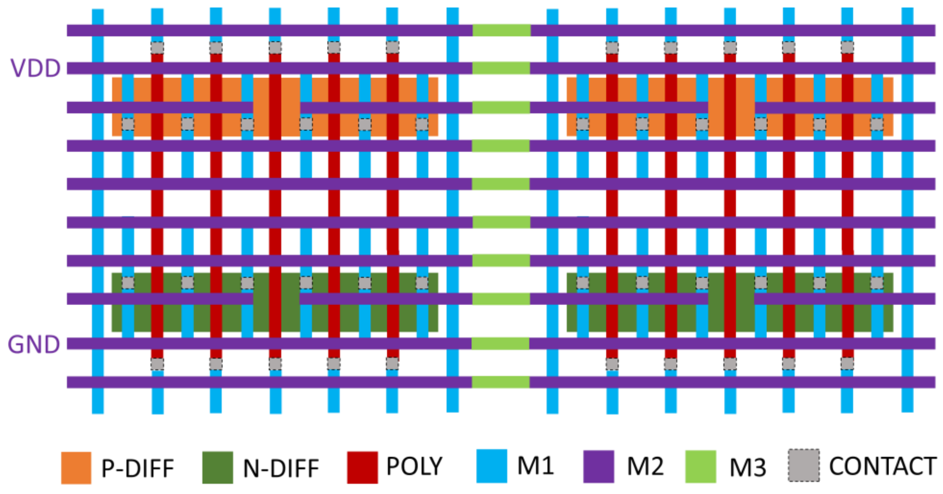

PREVIOUSLY DESCRIBED FLOW FOR LOGICAL CAPACITY EVALUATION
AUTOMATED CORRECTNESS CHECK FLOW
VCSA block template 'T'
Transistor network of logic function 'F1'
Developed Java SW
Boolean variables and clauses of TxF1
SAT Solver
SAT
UNSAT
Variable values of a found solution 'S'
Perl Script
Transistor network of solution 'S' on 'T'
Logic simulation
Equivalent
Not Equivalent

- Exploring the flexibility of the developed SAT model

One of the main features of the implemented model is to support any given VCSA block template. This feature allows the investigation of which characteristics on a VCSA block template significantly impacts its logical capacity, determining trade-offs between the template area and the number of transistor networks that is able to express, enlightening guidelines for an effective design of a VCSA block template.

Another use for the mentioned feature is that the user can “trick” the tool by describing mode than one VCSA block template as one input. This way, it is possible to obtain insights about how larger functions can be implemented using more than a single block. An example is the last column in Table 5.6, build by feeding the developed environment two instances of the 5VCC template described as one, connected by M3 lines, as depicted in Figure 5.46.

Figure 5.46 - Two instances of 5VCC block template described as a new template.



- Scalability analysis

Some SAT models, as in (CORTADELLA, 2014), intentionally reduce the solutions search universe in order to become computationally treatable, inserting this way the possibility of considering Unsatisfiable a problem that, if entirely described, is originally Satisfiable.

The model described and tested in this thesis does not present this problem for its target application. However, it is important to study how far from reaching “its limit of use” the developed model is, or how fast the model variables and clauses increase as the system inputs grow.

As the SAT model definition depends on many variables, the Boolean variables and equations will vary with several VCSA template characteristics and input transistor network aspects. However, using varied sizes of the known templates and usual transistor networks can provide an overview on the “scalability” of the developed SAT model, as the following experiments demonstrate.

To the measurements shown in Figure 5.47, a transistor network of a NAND3 was used as input, while the number of transistors at each plan of input VCSA templates varied. VCC and VCTA templates were chosen due to their well-defined layout patterns, making it possible to build variations with different transistor count.

To the measurements shown in Figure 5.48, the main variations of VCC and VCTA were used, while the input transistor network varied from a NAND3 to a NAND20.

Figure 5.47 - Developed SAT model variables, clauses and literals count behavior as template transistor count increases.

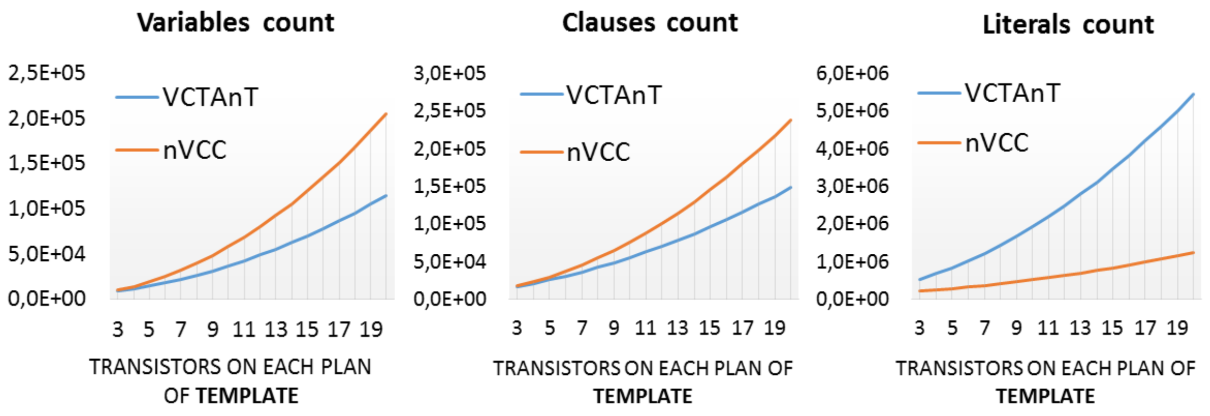
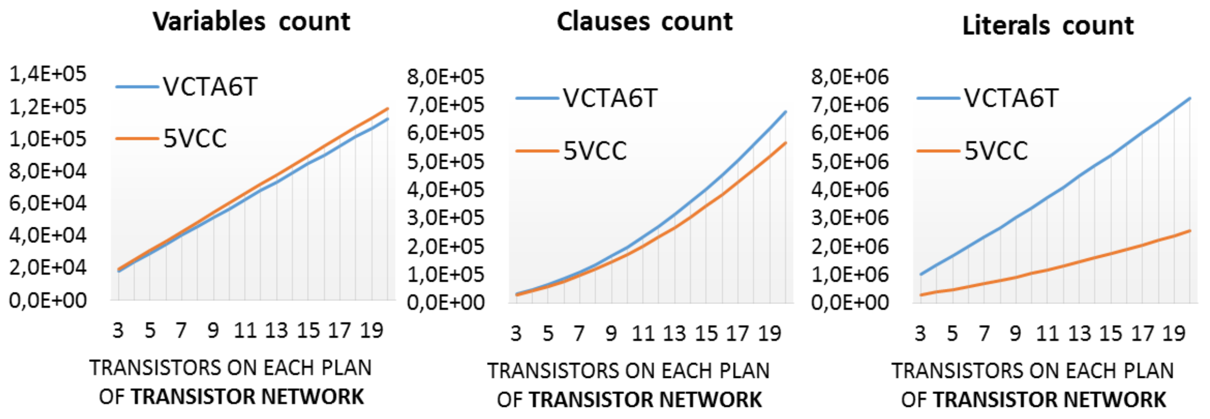


Figure 5.48 - Developed SAT model variables, clauses and literals count behavior as transistor networks increase.



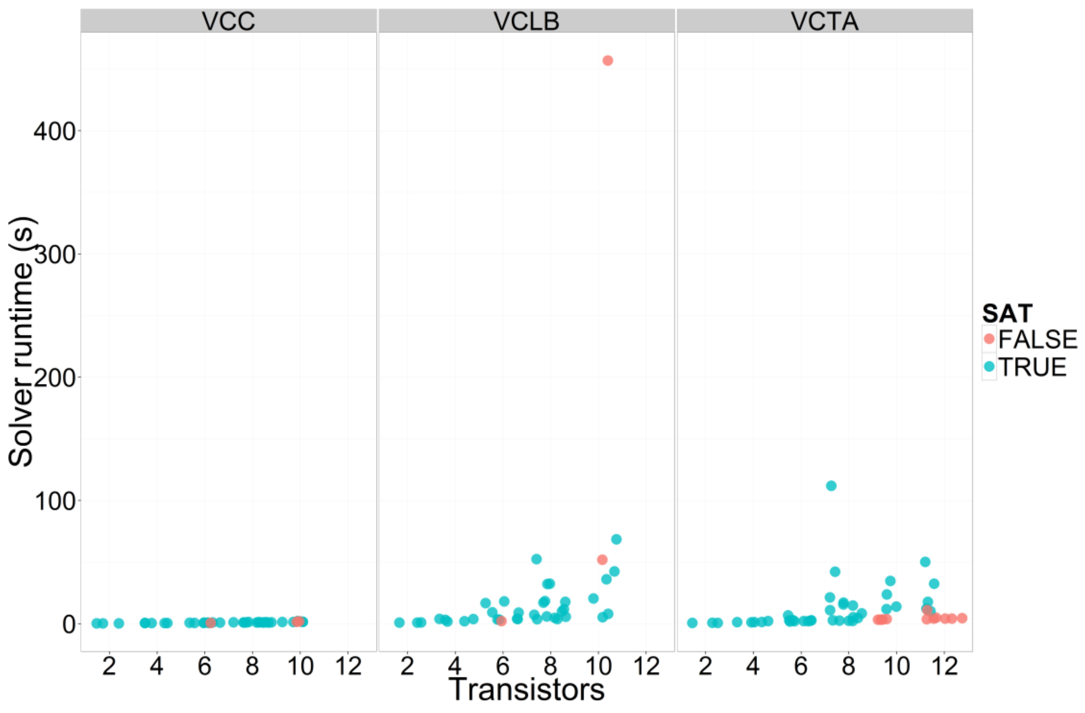
It is possible to observe in Figure 5.47 and in Figure 5.48 that more segmented templates, as VCC, tend to present more variables and clauses. However, templates with more paths among their elements, as VCTA, require much more literals, becoming ultimately disadvantageous.

It is also evident in Figure 5.47 and in Figure 5.48 that the model size usually increases with a slight polynomial behavior, close to linear. Regarding that the model is an input for a NP-complete problem, this “almost-linearity” does not suffice in to keep the required computational resources manageable for wide range of cases. This fact reassures the disadvantage of strongly connected VCSA templates, which present harder inclination on clauses increase graph of the developed SAT model.

Execution time is a measurement twisted by the computer hardware characteristics, operational system and chosen SAT solver. However, as an experimental insight, we state that the results for the tree first columns in Table 5.6 (VCSA template single blocks) where achieved within few minutes, already including the time required by the SAT solver, in a personal computer with processor Intel Core I5 and 6GB of RAM. The last column (2x5VCC) took few minutes for the equation files building, but the SAT solver used 10.5 hours to solve them all (without multi-threading). This is an indication that the limit of use for the developed model, in a personal computer, lies around the size of two blocks of a usual VCSA template. Meanwhile, it presents very high-performance for the scenario it is designed for (single block templates).

Detailed runtime on both Nangate Library set and 4-input P-Class set can be observed in Figure 5.49 and Figure 5.50.

Figure 5.49 - SAT solver runtime for Nangate Library set.



Observe that the worst cases runtime do not follow a pattern, happening to SAT inputs as well as to UNSAT inputs. Also, it is evident a strong dependence on the input layout template characteristics, highlighted in Table 5.7. In general, the solving time can be considered efficient, as it is not prohibitive even when using a standard personal computer, and regarding the possibility of parallel running for each evaluated network.

Figure 5.50 - SAT solver runtime for 4-input P-Class.

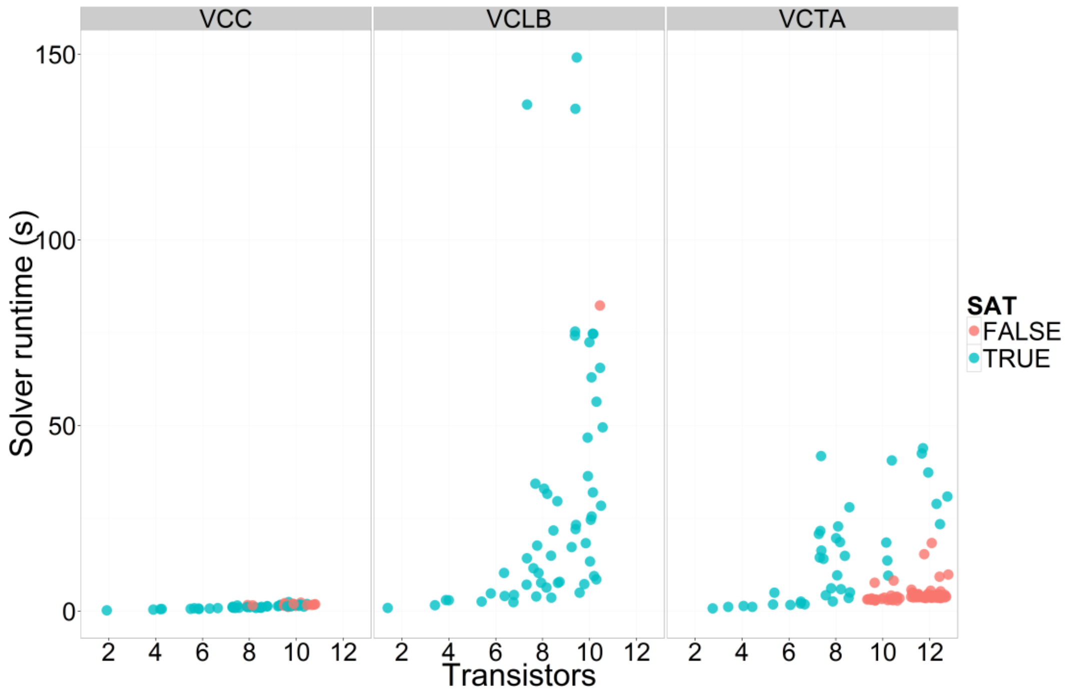


Table 5.7 - Summary of runtime (in seconds) of solver run on Nangate Library networks set.

Template	AVG (μ)	STD. Deviation (σ)	μ+3σ	Worst case
VCTA	10	19	67	112
VCC	1	0.4	2.25	2
VCLB	26	72	241	457

6 CONCLUSIONS AND FUTURE WORK

This thesis accomplishes its main proposal by describing an efficient tool for evaluation of VCSA block templates logical capacity.

Not only the task performed by the developed tool is novel, but its structures and methods: its concept is based on virtual connections between block layout and input transistor network, solving simultaneously transistors ordering and intra-cell wire routing, and including power supply lines treatment.

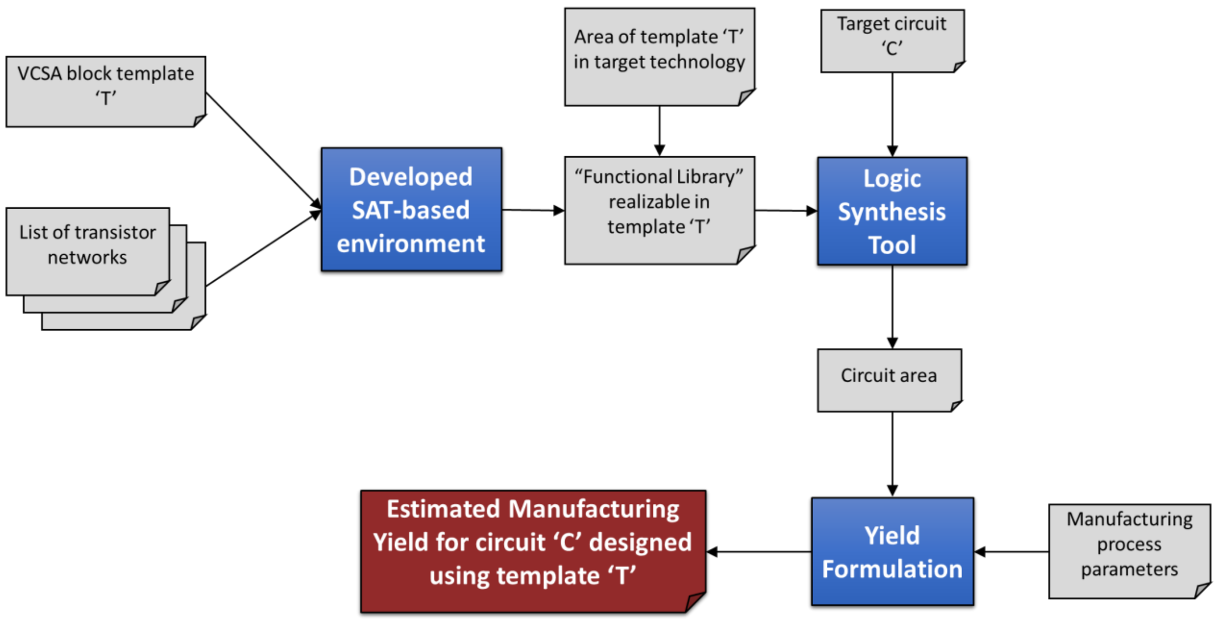
More than being compatible to the current main VCSA block templates on literature, the developed tool is also general as no other published, embracing any user-defined input block templates and input transistors network. This characteristic allows the user to test template variations and reach its own design guidelines for a target scenario.

A very intuitive application for the developed tool and suggested herein as future work is to implement a yield estimation environment combining the templates logical capacity information to the target circuit characteristics. Such environment flow could start with several input transistor networks and VCSA block templates, provided to the developed software that builds the SAT model described for each combination, that then will feed a SAT solver and determine the possible "functional libraries" for each VCSA block template. With these functional libraries and the area of each VCSA block template in a desired technology node, a liberty file can be built and provided to a mapper tool, together with a target circuit, finally reaching an area estimation. With the circuits area estimation and some manufacturing parameters several yield estimation formulas (KANO, 1998) can be applied. Figure 6.51 summarizes the described yield estimation flow.

Even if the target circuit is not available yet, making impossible the yield estimation described in the previous paragraph, the extraction method of logical capacity of VCSA templates provided herein can be useful, for instance, when balanced with electrical and lithographic performance for the composition of a quality metric to rank the available VCSA block templates for choice.

Still another application of the developed environment is to use it together with transistor networks generation algorithms. In such a scenario, the developed SAT environment can be source or target for transistor networks generation algorithms. It can be target in as the larger is the number of transistor networks provided for the fitting tests against a target VCSA block template the more precise is its logical capacity estimation. It can be source as exhaustive transistor network generation algorithms can be very expensive computationally, and the developed environment can provide a feedback to stop the networks generation when they are no longer useful for the target VCSA template.

Figure 6.51 – Manufacturing yield estimation flow for a given VCSA template and target circuit.


VCSA block template 'T'
List of transistor networks
Developed SAT-based environment
Area of template 'T' in target technology
"Functional Library" realizable in template 'T'
Target circuit 'C'
Logic Synthesis Tool
Circuit area
Estimated Manufacturing Yield for circuit 'C' designed using template 'T'
Yield Formulation
Manufacturing process parameters

All mentioned applications will be explored in future work, as well as the developed SAT model optimizations, expansions and adaptions. One very promising adaptation comes from changing the model targeting FPGA instead of VCSA, as it seems do not exist yet a published SAT model able to treat, simultaneously, place and route as this target.

REFERENCES

BOBBA, S.; DE MICHELI, G. Layout Technique for Double-Gate Silicon Nanowire FETs with an Efficient Sea-of-Tiles Architecture. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, vol. 23, no. 10, pp. 2103-2115, Oct. 2015.

BORKAR, S.; KARNIK, T.; NARENDRA, S.; TSCHANZ, J.; KESHAVARZI, A.; DE, V. Parameter variations and impact on circuits and microarchitecture. **Proceedings of Design Automation Conference (DAC)**, pp. 338-342, June 2003.

BOVET, D. P.; CRESCENZI, P. **Introduction to the Theory of Complexity**. New York: Prentice Hall, 1994.

BUTZEN, P. F; DAL BEM, V.; REIS, A. I.; RIBAS, R. P. Transistor network restructuring against NBTI degradation. **Microelectronics Reliability**, vol. 50, issues 9–11, pp. 1298-1303, Sept.-Nov. 2010.

BUTZEN, P. F; DAL BEM, V.; REIS, A. I.; RIBAS, R. P. Design of CMOS logic gates with enhanced robustness against aging degradation. **Microelectronics Reliability**, vol 52, issues 9–10, pp 1822-1826, Sept.-Oct. 2012.

F. BRGLEZ, F.; FUJIWARA, H. A neutral netlist of 10 combinational benchmark circuits and a target simulator in Fortran. **Proceedings of International Symposium on Circuits and Systems (ISCAS)**, pp. 695-698, June 1985.

CHEN, Y.C; PANG, H. Y.; LIN, K. W.; LIN, R. B.; TUNG, H. H.; SU, S. C. Via configurable three-input lookup-tables for structured ASICs. **Proceedings of the 20th symposium on Great lakes symposium on VLSI (GLSVLSI '10)**, pp. 49-54, May 2010.

CHIANG, C; KAWA, J. **Design for Manufacturability and Yield for Nano-Scale CMOS**. Secaucus: Springer-Verlag New York, Inc., 2007.

COOK, S. A. The complexity of theorem-proving procedures. **Proceedings of the third annual ACM symposium on Theory of computing (STOC '71)**, pp.151-158, May 1971.

CORTADELLA, J.; PETIT, J.; GOMEZ, S.; MOLL, F., A Boolean Rule-Based Approach for Manufacturability-Aware Cell Routing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol.33, no.3, pp.409-422, March 2014.

DAL BEM, V.; BUTZEN, P. F.; CLOCK, C. E.; CALLEGARO, V.; REIS, A. I.; RIBAS, R. P. Area impact analysis of via-configurable regular fabric for digital integrated circuit design. **Proceedings of the 24th symposium on Integrated circuits and systems design (SBCCI '11)**, pp. 103-108, Aug. 2011a.

DAL BEM, V.; BUTZEN, P.; MARRANGHELLO, F.S.; REIS, A.I.; RIBAS, R.P., Impact and optimization of lithography-aware regular layout in digital circuit design. **IEEE 29th International Conference on Computer Design (ICCD)**, issues 9-12, pp.279,284, Oct. 2011b.

DAL BEM, V.; REIS, A. I.; RIBAS, R. P. Lithography analysis of via-configurable transistor-array fabrics. **Proceedings of NORCHIP 2012**, pp. 1-4, Nov. 2012.

DONG, S.K.; SUN, Y.; SATO, S.; LIU, C.L. Two channel routing algorithm for quickly customized logic. **Proceedings of IEEE European Conference on Design Automation**, pp. 122-126, Feb. 1993.

FREEPDK. FreePDK of Nangate's Open Cell Library, user version of 2012. Available at: http://www.eda.ncsu.edu/. Accessed in: Jan. 2015.

GU, J.; PURI, R. Asynchronous circuit synthesis with Boolean satisfiability. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol.14, no.8, pp.961-973, Aug. 1995.

HE, F.; HUNG, W.N.N.; SONG, X.; GU, M; SUN, J. Segmented channel routing with pin rearrangements via satisfiability. **Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS 2005)**, vol. 6, pp. 6248-6251, May 2005.

HU, Y.; SHIH, V.; MAJUMDAR, R.; HE, L. Exploiting symmetry in SAT-based boolean matching for heterogeneous FPGA technology mapping. **Proceedings of IEEE/ACM International Conference on Computer-Aided Design (ICCAD 2007)**, pp. 350-353, Nov. 2007.

IIZUKA, T.; IKEDA, M.; ASADA, K. High speed layout synthesis for minimum-width CMOS logic cells via Boolean satisfiability. **Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC 2004)**, pp.149-154, Jan. 2004.

JHAVERI, T. K.; ROVNER, V.; PILEGGI, L.; STROJWAS, A. J.; MOTIANI, D.; KHETERPAL, V.; TONG, K. Y.; HERSAN, T.; PANDINI, D. Maximization of layout printability/manufacturability by extreme layout regularity. **Journal of Micro/Nanolithography, MEMS, and MOEMS**, vol. 6, issue 3, July 2007.

JHAVERI, T.; ROVNER, V.; LIEBMANN, L.; PILEGGI, L.; STROJWAS, A.J.; HIBBELER, J.D. Co-Optimization of Circuits, Layout and Lithography for Predictive Technology Scaling Beyond Gratings. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol. 29, no. 4, pp.509-527, April 2010.

KANO, K. **Semiconductor Devices**. Upper Saddle River: Prentice Hall, 1998.

KHOMENKO, V.; KOUTNY, M.; YAKOVLEV, A. Detecting state coding conflicts in STG unfoldings using SAT, **Proceedings of Third International Conference on Application of Concurrency to System Design**, pp. 51-60, June 2003.

KHOMENKO, V.; KOUTNY, M.; YAKOVLEV, A. Logic synthesis for asynchronous circuits based on Petri net unfoldings and incremental SAT. **Proceedings of Fourth International Conference on Application of Concurrency to System Design (ACSD 2004)**, pp. 16-25, June 2004.

KUHN, K.J. Reducing Variation in Advanced Logic Technologies: Approaches to Process and Design for Manufacturability of Nanoscale CMOS. **Proceedings of IEEE International Electron Devices Meeting (IEDM)**. pp. 471-474, Dec. 2007.

LI, M. C.; TUNG, H. H.; LAI, C. C.; LIN, R. B. Standard Cell Like Via-Configurable Logic Block for Structured ASICs. **IEEE Computer Society Annual Symposium on VLSI 2008 (ISVLSI '08)**, pp. 381-386, April 2008.

LIMA, F.; JOHANN, M.; GÜNTZEL, J.; D'AVILA, E.; CARRO, L.; REIS, R. Designing a Mask Programmable Matrix for Sequential Circuits. **VLSI: Systems on a Chip**, Boston: Springer US, 2000. pp. 439-446.

LIN, Y. W.; SADOWSKA, M. M.; MALY, W. Transistor-level layout of high-density regular circuits. **Proceedings of the 2009 international symposium on Physical design (ISPD '09)**, pp. 83-90, March 2009.

LING, A.; SINGH, D.P.; BROWN, S.D. FPGA technology mapping: a study of optimality. **Proceedings of 42nd Design Automation Conference (DAC)**, pp. 427-432, June 2005.

MARRANGHELLO, F. S.; DAL BEM, V.; REIS, A. I.; MOLL, F; RIBAS, R. P. Transistor sizing in lithography-aware regular fabrics. **Proceedings of the 24th symposium on Integrated circuits and systems design (SBCCI '11)**, pp. 97-102, Aug. 2011.

MITRA, J.; PENG YU; PAN, D.Z. RADAR: RET-aware detailed routing using fast lithography simulations. **Proceedings of 42nd Design Automation Conference (DAC)**. 369-372, June 2005.

MUKHERJEE, S.; ROY, S. SAT Based Multi Pin Net Detailed Routing for FPGA. **Proceedings of International Symposium on Electronic System Design (ISED)**, pp.141-146, Dec. 2010.

NAM, G. J.; SAKALLAH, K.A.; RUTENBAR, R.A. Satisfiability-based detailed FPGA routing. **Proceeding of Twelfth International Conference on VLSI Design**, pp. 574-577, Jan. 1999a.

NAM, G. J.; SAKALLAH, K.A.; RUTENBAR, R.A. Satisfiability-based layout revisited: detailed routing of complex FPGAs via search-based Boolean SAT. **Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays (FPGA '99)**, pp. 167-175, Feb. 1999b.

NAM, G. J.; SAKALLAH, K.A.; RUTENBAR, R.A. A new FPGA detailed routing approach via search-based Boolean satisfiability. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, vol. 21, no. 6, pp. 674-684, June 2002.

NOURY, L.; DUPUIS, S.; FEL, N. A reference low-complexity structured ASIC. **IEEE International Symposium on Circuits and Systems (ISCAS)**, pp. 2709-2712, May 2012.

ORSHANSKY, M.; NASSIF, S.; BONING, D. **Design for Manufacturability and Statistical Design - A Constructive Approach**. 1st ed. Heidelberg: Springer Publishing Company, Inc., 2008.

PONS, M.; MOLL, F.; RUBIO, A.; ABELLA, J.; VERA, X.; GONZALEZ, A. VCTA: A Via-Configurable Transistor Array regular fabric. **Proceedings of VLSI System on 18th IEEE/IFIP Chip Conference (VLSI-SoC 2010)**, pp. 335-340, Sept. 2010.

PONS, M.; BARAJAS, E.; MATEO, D.; GONZALEZ, J.L.; MOLL, F.; RUBIO, A.; ABELLA, J.; VERA, X.; GONZALEZ, A. Fast time-to-market with via-configurable transistor array regular fabric: A delay-locked loop design case study. **Proceedings of the 6th International Conference on Design & Technology of Integrated Systems in Nanoscale Era (DTIS),** pp. 1-6, April 2011.

PONS, M.; MORGAN, M.; PIGUET, C. Fixed origin corner square inspection layout regularity metric. **Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)**, pp. 1397-1402, March 2012.

POSSANI, V.N.; CALLEGARO, V.; REIS, A.I.; RIBAS, R.P.; DE SOUZA MARQUES, F.; DA ROSA, L.S. Graph-Based Transistor Network Generation Method for Supergate Design. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, vol. 24, no. 2, pp. 692-705, Feb. 2016.

QUINTANA, J. M.; AVEDILLO, M. J.; PARRA, M. P.; HUERTAS, J. L. Optimum PLA folding through boolean satisfiability. **Proceedings of the 1995 Asia and South Pacific Design Automation Conference (ASP-DAC '95)**, pp. 289-293, Aug. 1995.

RAN, Y.; MAREK-SADOWSKA, M. On designing via-configurable cell blocks for regular fabrics. **Proceeding of 41st Design Automation Conference (DAC)**, pp. 198-203, July 2004a.

RAN, Y.; MAREK-SADOWSKA, M. The magic of a via-configurable regular fabric. **Proceedings of IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD)**, pp. 338-343, Oct. 2004b.

RAN, Y.; MAREK-SADOWSKA, M. Designing a via-configurable regular fabric, **Proceedings of the IEEE Custom Integrated Circuits Conference (CICC)**, pp. 423-426, Oct. 2004c.

RAN, Y.; MAREK-SADOWSKA, M. Designing via-configurable logic blocks for regular fabric. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, vol. 14, no. 1, pp. 1-14, Jan. 2006a.

RAN, Y.; MAREK-SADOWSKA, M. Via-Configurable Routing Architectures and Fast Design Mappability Estimation for Regular Fabrics. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, vol. 14, no. 9, pp. 998-1009, Sept. 2006b.

ROY, S.; CHAKRABARTI, P.P.; DASGUPTA, P. Bounded Delay Timing Analysis Using Boolean Satisfiability. **Proceedings of 20th International Conference on VLSI Design**, pp. 295-302, Jan. 2007.

RYZHENKO, N.; BURNS, S. Standard cell routing via Boolean satisfiability. **Proceedings of the 49th Design Automation Conference (DAC)**, pp. 603-612, June 2012.

SAFARPOUR, S.; VENERIS, A.; BAECKLER, G.; YUAN, R. Efficient SAT-based Boolean matching for FPGA technology mapping. **Proceedings of 43rd Design Automation Conference (DAC)**, pp. 466-471, July 2006.

SASAO, T. Equivalence Classes of Logic Functions. **Switching Theory for Logic Synthesis**. Dordrecht: Kluwer Academic Publishers, 1999. pp. 106-116.

SENTOVICH, E. M.; SINGH, K. J.; LAVAGNO, L.; MOON, C.; MURGAI, R.; SALDANHA, A.; SAVOJ, H.; STEPHAN, P. R.; BRAYTON, R. K.; SANGIOVANNI-VINCENTELLI, A. L. **SIS: a system for sequential circuit synthesis**, Tech. Rep. UCB/ERL M92/41. EECS Department, University of California, Berkeley, 1992.

SUTO, G. Rule agnostic routing by using design fabrics. **Proceedings of the 49th Design Automation Conference (DAC)**, pp. 471-475, June 2012.

SRINIVASAN, N. Tutorial: Current state of P vs NP problem. **Proceedings of International Conference on Recent Trends in Information Technology (ICRTIT)**, pp. 1-1, June 2011.

TAYLOR, B. **Automated Layout of Regular Fabric Bricks**. 2005. 29 f. Master's thesis. Electrical and Computer Engineering Department, Carnegie Mellon University. Pittsburg PA, Dec. 2005.

TAYLOR, B.; PILEGGI, L. Exact Combinatorial Optimization Methods for Physical Design of Regular Logic Bricks. **Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC '07)**, pp. 344-349, June 2007.

TRAKHTENBROT, B.A. A Survey of Russian Approaches to Perebor (Brute-Force Searches) Algorithms. **Annals of the History of Computing**, vol. 6, no. 4, pp. 384-400, Oct.-Dec. 1984.

TSAI, H. P.; LIN, R. B.; LAI, L. C. Design and analysis of via-configurable routing fabrics for structured ASICs. **Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)**, pp.1479-1482, March 2012.

TUNG, H. H.; CHEN, Y. C.; HSU, D. W.; HSU, S. J.; CHEN, S. Y; LIN, R. B. Via-configurable logic block architectures for standard cell like structured ASICs. **Proceedings of the 12th International Symposium on Integrated Circuits (ISIC '09)**, pp. 17-20, Dec. 2009.

TUNG, H. H.; LIN, R. B.; LI, M. C.; HEISH T. H. Standard Cell Like Via-Configurable Logic Blocks for Structured ASIC in an Industrial Design Flow. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, vol. 20, no. 12, pp. 2184-2197, Dec. 2012.

UEHARA, T.; VANCLEEMPUT, W. M. Optimal Layout of CMOS Functional Arrays. **IEEE Transactions on Computers**, vol. 30, no. 5, pp. 305-312, May 1981.

VAN NOIJE, W.A.; DECLERCK, G.J. Advanced CMOS gate array architecture combining 'gate isolation' and programmable routing channels. **IEEE Journal of Solid-State Circuits**, vol. 20, no. 2, pp. 469-480, April 1985.

VATTIKONDA, R.; WENPING WANG; YU CAO. Modeling and minimization of PMOS NBTI effect for robust nanometer design. **Proceedings of 43rd Design Automation Conference (DAC)**, pp. 1047-1052, July 2006.

WILDERMANN, S.; TEICH, J.; ZIENER, D. Unifying Partitioning and Placement for SAT-Based Exploration of Heterogeneous Reconfigurable SoCs. **Proceedings of International Conference on Field Programmable Logic and Applications (FPL)**, pp. 429-434, Sept. 2011.

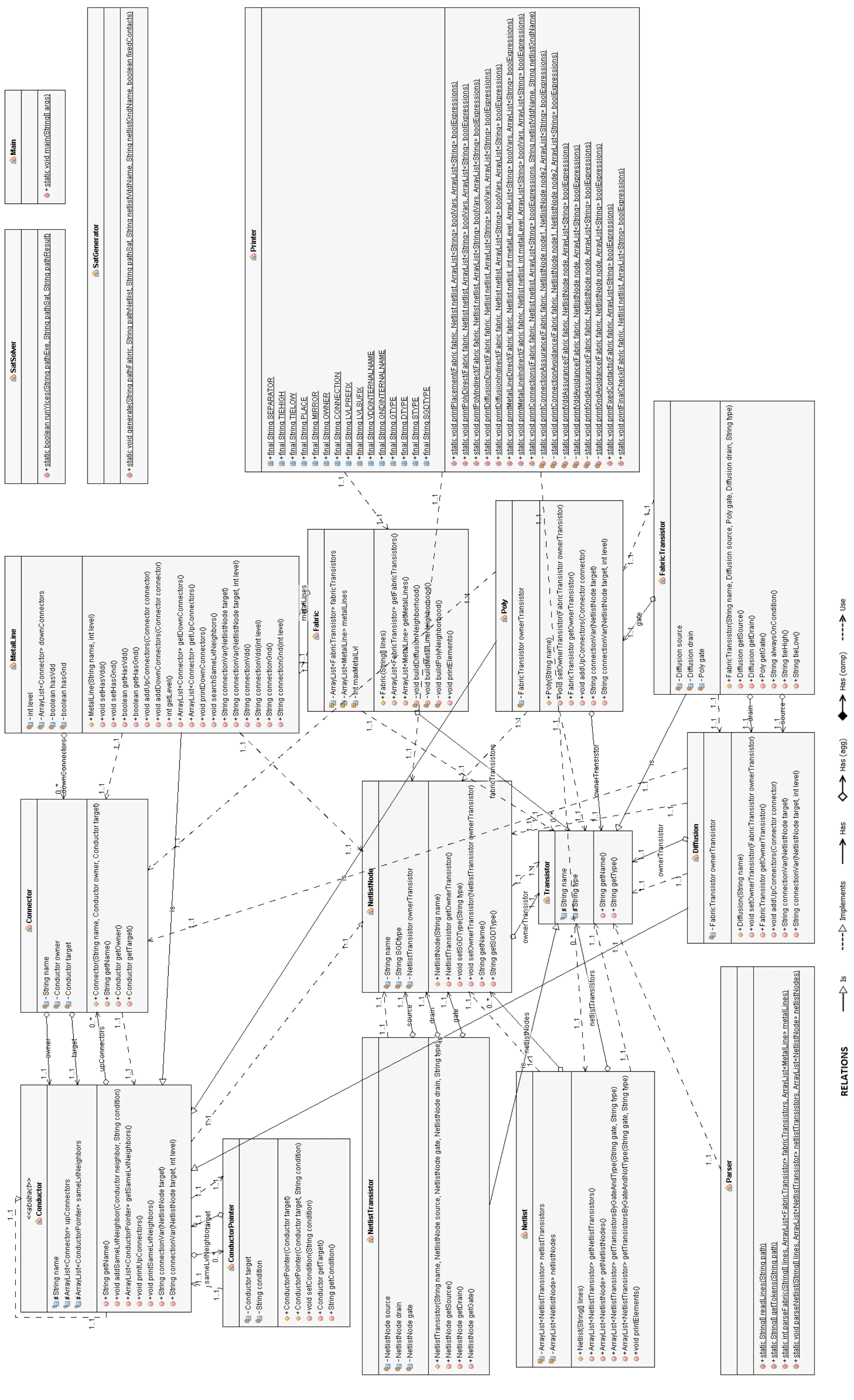
WONG, B. P.; MITTAL, A.; STARR, G. W.; ZACH, F.; MOROZ, V.; KAHNG, A. **Nano-CMOS Design for Manufacturability: Robust Circuit and Physical Design for Sub-65nm Technology Nodes**. New York: Wiley-Interscience, 2008.

WOOD, R.G.; RUTENBAR, R.A. FPGA routing and routability estimation via Boolean satisfiability. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, vol.6, no. 2, pp. 222-231, June 1998.

YANG, F.; CAI, Y.; ZHOU, Q.; HU, J. SAT based multi-net rip-up-and-reroute for manufacturing hotspot removal. **Proceedings of Design, Automation & Test in Europe Conference & Exhibition (DATE)**, pp. 1369-1372, March 2010.

YICES. Sat solver, 2015 version. Available at: http://yices.csl.sri.com/. Accessed in: Jan. 2015.

APPENDIX A – CLASS DIAGRAM OF IMPLEMENTED SOFTWARE



APPENDIX B – TEXTUAL DESCRIPTION OF VCSA BLOCK TEMPLATES

Syntax description:

- Transistors: m[name] [source name] [gate name] [drain name] p|n
    - p: p-type transistor
    - n: n-type transistor
- Metal lines: l|g|v[name] [connector to lower layer] [connector target] ... [metal level]
    - l: ordinary metal line
    - v: metal line containing VDD
    - g: metal line containing GND

Template 5VCC:

```
mpa pasrc pya padrn p
mpb padrn pyb pbdrn p
mpc pbdrn pyc pcdrn p
mpd pcdrn pyd pddrn p
mpe pddrn pye pedrn p
mna nasrc pya nadrn n
mnb nadrn pyb nbdrn n
mnc nbdrn pyc ncdrn n
mnd ncdrn pyd nddrn n
mne nddrn pye nedrn n
l1_aup cont_aup pya 1
l1_bup cont_bup pyb 1
l1_cup cont_cup pyc 1
l1_dup cont_dup pyd 1
l1_eup cont_eup pye 1
l1_adn cont_adn pya 1
l1_bdn cont_bdn pyb 1
l1_cdn cont_cdn pyc 1
l1_ddn cont_ddn pyd 1
l1_edn cont_edn pye 1
l1_u1 cont_u1 pasrc 1
```

l1_u2 cont_u2 padrn 1

l1_u3 cont_u3 pbdrn 1

l1_u4 cont_u4 pcdrn 1

l1_u5 cont_u5 pddrn 1

l1_u6 cont_u6 pedrn 1

l1_d1 cont_d1 nasrc 1

l1_d2 cont_d2 nadrn 1

l1_d3 cont_d3 nbdrn 1

l1_d4 cont_d4 ncdrn 1

l1_d5 cont_d5 nddrn 1

l1_d6 cont_d6 nedrn 1

l1_c1 1

l1_c2 1

l2_f1 via_f1_c1 l1_c1 via_f1_aup l1_aup via_f1_bup l1_bup via_f1_cup l1_cup via_f1_dup l1_dup via_f1_eup l1_eup via_f1_c2 l1_c2 2

v2_vdd via_vdd_c1 l1_c1 via_vdd_aup l1_aup via_vdd_bup l1_bup via_vdd_cup l1_cup via_vdd_dup l1_dup via_vdd_eup l1_eup via_vdd_c2 l1_c2 via_vdd_u1 l1_u1 via_vdd_u2 l1_u2 via_vdd_u3 l1_u3 via_vdd_u4 l1_u4 via_vdd_u5 l1_u5 via_vdd_u6 l1_u6 2

l2_w1 via_w1_c1 l1_c1 via_w1_u1 l1_u1 via_w1_u2 l1_u2 via_w1_u3 l1_u3 2

l2_w2 via_w2_c2 l1_c2 via_w2_u4 l1_u4 via_w2_u5 l1_u5 via_w2_u6 l1_u6 2

l2_w3 via_w3_c1 l1_c1 via_w3_u1 l1_u1 via_w3_u2 l1_u2 via_w3_u3 l1_u3 via_w3_u4 l1_u4 via_w3_u5 l1_u5 via_w3_u6 l1_u6 via_w3_c2 l1_c2 2

l2_w4 via_w4_c1 l1_c1 via_w4_u1 l1_u1 via_w4_u2 l1_u2 via_w4_u3 l1_u3 via_w4_u4 l1_u4 via_w4_u5 l1_u5 via_w4_u6 l1_u6 via_w4_c2 l1_c2 2

l2_w5 via_w5_c1 l1_c1 via_w5_d1 l1_d1 via_w5_d2 l1_d2 via_w5_d3 l1_d3 via_w5_d4 l1_d4 via_w5_d5 l1_d5 via_w5_d6 l1_d6 via_w5_c2 l1_c2 2

l2_w6 via_w6_c1 l1_c1 via_w6_d1 l1_d1 via_w6_d2 l1_d2 via_w6_d3 l1_d3 via_w6_d4 l1_d4 via_w6_d5 l1_d5 via_w6_d6 l1_d6 via_w6_c2 l1_c2 2

l2_w7 via_w7_c1 l1_c1 via_w7_d1 l1_d1 via_w7_d2 l1_d2 via_w7_d3 l1_d3 2

l2_w8 via_w8_c2 l1_c2 via_28_d4 l1_d4 via_w8_d5 l1_d5 via_w8_d6 l1_d6 2

g2_gnd via_gnd_c1 l1_c1 via_gnd_adn l1_adn via_gnd_bdn l1_bdn via_gnd_cdn l1_cdn via_gnd_ddn l1_ddn via_gnd_edn l1_edn via_gnd_c2 l1_c2 via_gnd_d1 l1_d1 via_gnd_d2 l1_d2 via_gnd_d3 l1_d3 via_gnd_d4 l1_d4 via_gnd_d5 l1_d5 via_gnd_d6 l1_d6 2 l2_f2 via_f2_c1 l1_c1 via_f2_adn l1_adn via_f2_bdn l1_bdn via_f2_cdn l1_cdn via_f2_ddn l1_ddn via_f2_edn l1_edn via_f2_c2 l1_c2 2

Template VCTA6T:

mpa pasrc pya padrn p

mpb padrn pyb pbdrn p

mpc pbdrn pyc pcdrn p

mpd pcdrn pyd pddrn p

mpe pddrn pye pedrn p

mpf pedrn pyf pfdrn p

mna nasrc nya nadrn n

mnb nadrn nyb nbdrn n

mnc nbdrn nyc ncdrn n

mnd ncdrn nyd nddrn n

mne nddrn nye nedrn n

mnf nedrn nyf nfdrn n

v1_1 1

l1_2 cont_l1_2_pya pya cont_l1_2_pyb pyb cont_l1_2_pyc pyc cont_l1_2_pyd pyd cont_l1_2_pye pye cont_l1_2_pyf pyf cont_l1_2_nya nya cont_l1_2_nyb nyb cont_l1_2_nyc nyc cont_l1_2_nyd nyd cont_l1_2_nye nye cont_l1_2_nyf nyf 1

l1_3 cont_l1_3_pya pya cont_l1_3_pyb pyb cont_l1_3_pyc pyc cont_l1_3_pyd pyd cont_l1_3_pye pye cont_l1_3_pyf pyf cont_l1_3_nya nya cont_l1_3_nyb nyb cont_l1_3_nyc nyc cont_l1_3_nyd nyd cont_l1_3_nye nye cont_l1_3_nyf nyf 1

l1_4 cont_l1_4_pya pya cont_l1_4_pyb pyb cont_l1_4_pyc pyc cont_l1_4_pyd pyd cont_l1_4_pye pye cont_l1_4_pyf pyf cont_l1_4_nya nya cont_l1_4_nyb nyb cont_l1_4_nyc nyc cont_l1_4_nyd nyd cont_l1_4_nye nye cont_l1_4_nyf nyf 1

l1_5 cont_l1_5_pya pya cont_l1_5_pyb pyb cont_l1_5_pyc pyc cont_l1_5_pyd pyd cont_l1_5_pye pye cont_l1_5_pyf pyf cont_l1_5_nya nya cont_l1_5_nyb nyb cont_l1_5_nyc nyc cont_l1_5_nyd nyd cont_l1_5_nye nye cont_l1_5_nyf nyf 1

g1_6 1

l1_pasrc cont_pasrc pasrc 1

l1_padrn cont_padrn padrn 1

l1_pbdrn cont_pbdrn pbdrn 1

```
l1_pcdrn cont_pcdrn pcdrn 1
l1_pddrn cont_pddrn pddrn 1
l1_pedrn cont_pedrn pedrn 1
l1_pfdrn cont_pfdrn pfdrn 1
l1_nasrc cont_nasrc nasrc 1
l1_nadrn cont_nadrn nadrn 1
l1_nbdrn cont_nbdrn nbdrn 1
l1_ncdrn cont_ncdrn ncdrn 1
l1_nddrn cont_nddrn nddrn 1
l1_nedrn cont_nedrn nedrn 1
l1_nfdrn cont_nfdrn nfdrn 1
v2_1 via_v2_1_v1_1 v1_1 via_v2_1_l1_2 l1_2 via_v2_1_l1_3 l1_3 via_v2_1_l1_4 l1_4
via_v2_1_l1_5 l1_5 via_v2_1_g1_6 g1_6 2
l2_2 via_l2_2_v1_1 v1_1 via_l2_2_l1_2 l1_2 via_l2_2_l1_3 l1_3 via_l2_2_l1_4 l1_4
via_l2_2_l1_5 l1_5 via_l2_2_g1_6 g1_6 2
l2_3 via_l2_3_v1_1 v1_1 via_l2_3_l1_2 l1_2 via_l2_3_l1_3 l1_3 via_l2_3_l1_4 l1_4
via_l2_3_l1_5 l1_5 via_l2_3_g1_6 g1_6 2
l2_13 via_l2_13_v1_1 v1_1 via_l2_13_l1_2 l1_2 via_l2_13_l1_3 l1_3 via_l2_13_l1_4 l1_4
via_l2_13_l1_5 l1_5 via_l2_13_g1_6 g1_6 2
l2_23 via_l2_23_v1_1 v1_1 via_l2_23_l1_2 l1_2 via_l2_23_l1_3 l1_3 via_l2_23_l1_4 l1_4
via_l2_23_l1_5 l1_5 via_l2_23_g1_6 g1_6 2
l2_24 via_l2_24_v1_1 v1_1 via_l2_24_l1_2 l1_2 via_l2_24_l1_3 l1_3 via_l2_24_l1_4 l1_4
via_l2_24_l1_5 l1_5 via_l2_24_g1_6 g1_6 2
g2_25 via_g2_25_v1_1 v1_1 via_g2_25_l1_2 l1_2 via_g2_25_l1_3 l1_3 via_g2_25_l1_4
l1_4 via_g2_25_l1_5 l1_5 via_g2_25_g1_6 g1_6 2
l2_4 via_l2_4_v1_1 v1_1 via_l2_4_l1_2 l1_2 via_l2_4_l1_3 l1_3 via_l2_4_l1_4 l1_4
via_l2_4_l1_5 l1_5 via_l2_4_g1_6 g1_6 via_l2_4_l1_pasrc l1_pasrc 2
l2_5 via_l2_5_v1_1 v1_1 via_l2_5_l1_2 l1_2 via_l2_5_l1_3 l1_3 via_l2_5_l1_4 l1_4
via_l2_5_l1_5 l1_5 via_l2_5_g1_6 g1_6 via_l2_5_l1_padrn l1_padrn 2
l2_6 via_l2_6_v1_1 v1_1 via_l2_6_l1_2 l1_2 via_l2_6_l1_3 l1_3 via_l2_6_l1_4 l1_4
via_l2_6_l1_5 l1_5 via_l2_6_g1_6 g1_6 via_l2_6_l1_pbdrn l1_pbdrn 2
l2_7 via_l2_7_v1_1 v1_1 via_l2_7_l1_2 l1_2 via_l2_7_l1_3 l1_3 via_l2_7_l1_4 l1_4
via_l2_7_l1_5 l1_5 via_l2_7_g1_6 g1_6 via_l2_7_l1_pcdrn l1_pcdrn 2
```

l2_8 via_l2_8_v1_1 v1_1 via_l2_8_l1_2 l1_2 via_l2_8_l1_3 l1_3 via_l2_8_l1_4 l1_4
via_l2_8_l1_5 l1_5 via_l2_8_g1_6 g1_6 via_l2_8_l1_pddrn l1_pddrn 2
l2_9 via_l2_9_v1_1 v1_1 via_l2_9_l1_2 l1_2 via_l2_9_l1_3 l1_3 via_l2_9_l1_4 l1_4
via_l2_9_l1_5 l1_5 via_l2_9_g1_6 g1_6 via_l2_9_l1_pedrn l1_pedrn 2
l2_10 via_l2_10_v1_1 v1_1 via_l2_10_l1_2 l1_2 via_l2_10_l1_3 l1_3 via_l2_10_l1_4 l1_4
via_l2_10_l1_5 l1_5 via_l2_10_g1_6 g1_6 via_l2_10_l1_pfdrn l1_pfdrn 2
l2_11 via_l2_11_v1_1 v1_1 via_l2_11_l1_2 l1_2 via_l2_11_l1_3 l1_3 via_l2_11_l1_4 l1_4
via_l2_11_l1_5 l1_5 via_l2_11_g1_6 g1_6 2
l2_12 via_l2_12_v1_1 v1_1 via_l2_12_l1_2 l1_2 via_l2_12_l1_3 l1_3 via_l2_12_l1_4 l1_4
via_l2_12_l1_5 l1_5 via_l2_12_g1_6 g1_6 2
l2_14 via_l2_14_v1_1 v1_1 via_l2_14_l1_2 l1_2 via_l2_14_l1_3 l1_3 via_l2_14_l1_4 l1_4
via_l2_14_l1_5 l1_5 via_l2_14_g1_6 g1_6 via_l2_14_l1_nasrc l1_nasrc 2
l2_15 via_l2_15_v1_1 v1_1 via_l2_15_l1_2 l1_2 via_l2_15_l1_3 l1_3 via_l2_15_l1_4 l1_4
via_l2_15_l1_5 l1_5 via_l2_15_g1_6 g1_6 via_l2_15_l1_nadrn l1_nadrn 2
l2_16 via_l2_16_v1_1 v1_1 via_l2_16_l1_2 l1_2 via_l2_16_l1_3 l1_3 via_l2_16_l1_4 l1_4
via_l2_16_l1_5 l1_5 via_l2_16_g1_6 g1_6 via_l2_16_l1_nbdrn l1_nbdrn 2
l2_17 via_l2_17_v1_1 v1_1 via_l2_17_l1_2 l1_2 via_l2_17_l1_3 l1_3 via_l2_17_l1_4 l1_4
via_l2_17_l1_5 l1_5 via_l2_17_g1_6 g1_6 via_l2_17_l1_ncdrn l1_ncdrn 2
l2_18 via_l2_18_v1_1 v1_1 via_l2_18_l1_2 l1_2 via_l2_18_l1_3 l1_3 via_l2_18_l1_4 l1_4
via_l2_18_l1_5 l1_5 via_l2_18_g1_6 g1_6 via_l2_18_l1_nddrn l1_nddrn 2
l2_19 via_l2_19_v1_1 v1_1 via_l2_19_l1_2 l1_2 via_l2_19_l1_3 l1_3 via_l2_19_l1_4 l1_4
via_l2_19_l1_5 l1_5 via_l2_19_g1_6 g1_6 via_l2_19_l1_nedrn l1_nedrn 2
l2_20 via_l2_20_v1_1 v1_1 via_l2_20_l1_2 l1_2 via_l2_20_l1_3 l1_3 via_l2_20_l1_4 l1_4
via_l2_20_l1_5 l1_5 via_l2_20_g1_6 g1_6 via_l2_20_l1_nfdrn l1_nfdrn 2
l2_21 via_l2_21_v1_1 v1_1 via_l2_21_l1_2 l1_2 via_l2_21_l1_3 l1_3 via_l2_21_l1_4 l1_4
via_l2_21_l1_5 l1_5 via_l2_21_g1_6 g1_6 2
l2_22 via_l2_22_v1_1 v1_1 via_l2_22_l1_2 l1_2 via_l2_22_l1_3 l1_3 via_l2_22_l1_4 l1_4
via_l2_22_l1_5 l1_5 via_l2_22_g1_6 g1_6 2
l3_1 via_l3_1_v2_1 v2_1 via_l3_1_l2_2 l2_2 via_l3_1_l2_3 l2_3 via_l3_1_l2_4 l2_4
via_l3_1_l2_5 l2_5 via_l3_1_l2_6 l2_6 via_l3_1_l2_7 l2_7 via_l3_1_l2_8 l2_8
via_l3_1_l2_9 l2_9 via_l3_1_l2_10 l2_10 via_l3_1_l2_11 l2_11 via_l3_1_l2_12 l2_12
via_l3_1_l2_13 l2_13 via_l3_1_l2_14 l2_14 via_l3_1_l2_15 l2_15 via_l3_1_l2_16 l2_16
via_l3_1_l2_17 l2_17 via_l3_1_l2_18 l2_18 via_l3_1_l2_19 l2_19 via_l3_1_l2_20 l2_20

via_l3_1_l2_21 l2_21 via_l3_1_l2_22 l2_22 via_l3_1_l2_23 l2_23 via_l3_1_l2_24 l2_24 via_l3_1_g2_25 g2_25 3

l3_2 via_l3_2_v2_1 v2_1 via_l3_2_l2_2 l2_2 via_l3_2_l2_3 l2_3 via_l3_2_l2_4 l2_4 via_l3_2_l2_5 l2_5 via_l3_2_l2_6 l2_6 via_l3_2_l2_7 l2_7 via_l3_2_l2_8 l2_8 via_l3_2_l2_9 l2_9 via_l3_2_l2_10 l2_10 via_l3_2_l2_11 l2_11 via_l3_2_l2_12 l2_12 via_l3_2_l2_13 l2_13 via_l3_2_l2_14 l2_14 via_l3_2_l2_15 l2_15 via_l3_2_l2_16 l2_16 via_l3_2_l2_17 l2_17 via_l3_2_l2_18 l2_18 via_l3_2_l2_19 l2_19 via_l3_2_l2_20 l2_20 via_l3_2_l2_21 l2_21 via_l3_2_l2_22 l2_22 via_l3_2_l2_23 l2_23 via_l3_2_l2_24 l2_24 via_l3_2_g2_25 g2_25 3

l3_3 via_l3_3_v2_1 v2_1 via_l3_3_l2_2 l2_2 via_l3_3_l2_3 l2_3 via_l3_3_l2_4 l2_4 via_l3_3_l2_5 l2_5 via_l3_3_l2_6 l2_6 via_l3_3_l2_7 l2_7 via_l3_3_l2_8 l2_8 via_l3_3_l2_9 l2_9 via_l3_3_l2_10 l2_10 via_l3_3_l2_11 l2_11 via_l3_3_l2_12 l2_12 via_l3_3_l2_13 l2_13 via_l3_3_l2_14 l2_14 via_l3_3_l2_15 l2_15 via_l3_3_l2_16 l2_16 via_l3_3_l2_17 l2_17 via_l3_3_l2_18 l2_18 via_l3_3_l2_19 l2_19 via_l3_3_l2_20 l2_20 via_l3_3_l2_21 l2_21 via_l3_3_l2_22 l2_22 via_l3_3_l2_23 l2_23 via_l3_3_l2_24 l2_24 via_l3_3_g2_25 g2_25 3

l3_4 via_l3_4_v2_1 v2_1 via_l3_4_l2_2 l2_2 via_l3_4_l2_3 l2_3 via_l3_4_l2_4 l2_4 via_l3_4_l2_5 l2_5 via_l3_4_l2_6 l2_6 via_l3_4_l2_7 l2_7 via_l3_4_l2_8 l2_8 via_l3_4_l2_9 l2_9 via_l3_4_l2_10 l2_10 via_l3_4_l2_11 l2_11 via_l3_4_l2_12 l2_12 via_l3_4_l2_13 l2_13 via_l3_4_l2_14 l2_14 via_l3_4_l2_15 l2_15 via_l3_4_l2_16 l2_16 via_l3_4_l2_17 l2_17 via_l3_4_l2_18 l2_18 via_l3_4_l2_19 l2_19 via_l3_4_l2_20 l2_20 via_l3_4_l2_21 l2_21 via_l3_4_l2_22 l2_22 via_l3_4_l2_23 l2_23 via_l3_4_l2_24 l2_24 via_l3_4_g2_25 g2_25 3

l3_5 via_l3_5_v2_1 v2_1 via_l3_5_l2_2 l2_2 via_l3_5_l2_3 l2_3 via_l3_5_l2_4 l2_4 via_l3_5_l2_5 l2_5 via_l3_5_l2_6 l2_6 via_l3_5_l2_7 l2_7 via_l3_5_l2_8 l2_8 via_l3_5_l2_9 l2_9 via_l3_5_l2_10 l2_10 via_l3_5_l2_11 l2_11 via_l3_5_l2_12 l2_12 via_l3_5_l2_13 l2_13 via_l3_5_l2_14 l2_14 via_l3_5_l2_15 l2_15 via_l3_5_l2_16 l2_16 via_l3_5_l2_17 l2_17 via_l3_5_l2_18 l2_18 via_l3_5_l2_19 l2_19 via_l3_5_l2_20 l2_20 via_l3_5_l2_21 l2_21 via_l3_5_l2_22 l2_22 via_l3_5_l2_23 l2_23 via_l3_5_l2_24 l2_24 via_l3_5_g2_25 g2_25 3

l3_6 via_l3_6_v2_1 v2_1 via_l3_6_l2_2 l2_2 via_l3_6_l2_3 l2_3 via_l3_6_l2_4 l2_4 via_l3_6_l2_5 l2_5 via_l3_6_l2_6 l2_6 via_l3_6_l2_7 l2_7 via_l3_6_l2_8 l2_8 via_l3_6_l2_9 l2_9 via_l3_6_l2_10 l2_10 via_l3_6_l2_11 l2_11 via_l3_6_l2_12 l2_12 via_l3_6_l2_13 l2_13 via_l3_6_l2_14 l2_14 via_l3_6_l2_15 l2_15 via_l3_6_l2_16 l2_16 via_l3_6_l2_17 l2_17 via_l3_6_l2_18 l2_18 via_l3_6_l2_19 l2_19 via_l3_6_l2_20 l2_20 via_l3_6_l2_21 l2_21 via_l3_6_l2_22 l2_22 via_l3_6_l2_23 l2_23 via_l3_6_l2_24 l2_24 via_l3_6_g2_25 g2_25 3

Template SLVC5P:

```
mpa pasrc ya padrn p
mpb padrn yb pbdrn p
mpc pbdrn yc pcdrn p
mpd pdsrc yd pddrn p
mpe pesrc ye pedrn p
mna nasrc ya nadrn n
mnb nadrn yb nbdrn n
mnc nbdrn yc ncdrn n
mnd ndsrc yd nddrn n
mne nesrc ye nedrn n
l1_1 1
l1_2 cont_l1_2_pasrc pasrc 1
l1_3 cont_l1_3_padrn padrn 1
l1_4 cont_l1_4_pbdrn pbdrn 1
l1_5 cont_l1_5_pcdrn pcdrn 1
l1_6 cont_l1_6_ya ya 1
l1_7 cont_l1_7_yb yb 1
l1_8 cont_l1_8_yc yc 1
l1_9 cont_l1_9_nasrc nasrc 1
l1_10 cont_l1_10_nadrn nadrn 1
l1_11 cont_l1_11_nbdrn nbdrn 1
l1_12 cont_l1_12_ncdrn ncdrn 1
l1_13 1
l1_14 1
l1_15 cont_l1_15_pdsrc pdsrc 1
l1_16 cont_l1_16_yd yd 1
l1_17 cont_l1_17_pddrn pddrn 1
l1_18 cont_l1_18_pesrc pesrc 1
```

l1_19 cont_l1_19_ye ye 1

l1_20 cont_l1_20_pedrn pedrn 1

l1_21 cont_l1_21_ndsrc ndsrc 1

l1_22 cont_l1_22_nddrn nddrn 1

l1_23 cont_l1_23_nesrc nesrc 1

l1_24 cont_l1_24_nedrn nedrn 1

l1_25 1

v2_vdd via_l2_vdd_l1_2 l1_2 via_l2_vdd_l1_3 l1_3 via_l2_vdd_l1_4 l1_4 via_l2_vdd_l1_5 l1_5 via_l2_vdd_l1_15 l1_15 via_l2_vdd_l1_16 l1_16 via_l2_vdd_l1_17 l1_17 via_l2_vdd_l1_18 l1_18 via_l2_vdd_l1_19 l1_19 via_l2_vdd_l1_20 l1_20 2

l2_1 via_l2_1_l1_2 l1_2 via_l2_1_l1_3 l1_3 via_l2_1_l1_4 l1_4 via_l2_1_l1_5 l1_5 via_l2_1_l1_13 l1_13 via_l2_1_l1_15 l1_15 via_l2_1_l1_16 l1_16 via_l2_1_l1_17 l1_17 via_l2_1_l1_18 l1_18 via_l2_1_l1_19 l1_19 via_l2_1_l1_20 l1_20 2

l2_2 via_l2_2_l1_1 l1_1 via_l2_2_l1_2 l1_2 via_l2_2_l1_3 l1_3 2

l2_3 via_l2_3_l1_4 l1_4 via_l2_3_l1_5 l1_5 via_l2_3_l1_13 l1_13 2

l2_4 via_l2_4_l1_1 l1_1 via_l2_4_l1_2 l1_2 via_l2_4_l1_3 l1_3 via_l2_4_l1_4 l1_4 via_l2_4_l1_5 l1_5 via_l2_4_l1_13 l1_13 2

l2_5 via_l2_5_l1_14 l1_14 via_l2_5_l1_15 l1_15 2

l2_6 via_l2_6_l1_20 l1_20 via_l2_6_l1_25 l1_25 2

l2_7 via_l2_7_l1_1 l1_1 via_l2_7_l1_2 l1_2 via_l2_7_l1_3 l1_3 via_l2_7_l1_4 l1_4 via_l2_7_l1_5 l1_5 via_l2_7_l1_13 l1_13 2

l2_8 via_l2_8_l1_14 l1_14 via_l2_8_l1_15 l1_15 via_l2_8_l1_16 l1_16 via_l2_8_l1_17 l1_17 via_l2_8_l1_18 l1_18 via_l2_8_l1_19 l1_19 via_l2_8_l1_20 l1_20 via_l2_8_l1_25 l1_25 2

l2_9 via_l2_9_l1_1 l1_1 via_l2_9_l1_6 l1_6 via_l2_9_l1_7 l1_7 via_l2_9_l1_8 l1_8 via_l2_9_l1_13 l1_13 via_l2_9_l1_14 l1_14 via_l2_9_l1_15 l1_15 via_l2_9_l1_16 l1_16 via_l2_9_l1_17 l1_17 via_l2_9_l1_18 l1_18 via_l2_9_l1_19 l1_19 via_l2_9_l1_20 l1_20 via_l2_9_l1_25 l1_25 2

l2_10 via_l2_10_l1_17 l1_17 via_l2_10_l1_22 l1_22 2

l2_11 via_l2_11_l1_18 l1_18 via_l2_11_l1_23 l1_23 2

l2_12 via_l2_12_l1_6 l1_6 via_l2_12_l1_7 l1_7 via_l2_12_l1_8 l1_8 via_l2_12_l1_13 l1_13 via_l2_12_l1_14 l1_14 via_l2_12_l1_21 l1_21 via_l2_12_l1_16 l1_16 via_l2_12_l1_22 l1_22 via_l2_12_l1_23 l1_23 via_l2_12_l1_19 l1_19 via_l2_12_l1_24 l1_24 via_l2_12_l1_25 l1_25 2 l2_13 via_l2_13_l1_1 l1_1 via_l2_13_l1_9 l1_9 via_l2_13_l1_10 l1_10 via_l2_13_l1_11 l1_11 via_l2_13_l1_12 l1_12 via_l2_13_l1_13 l1_13 via_l2_13_l1_14 l1_14 via_l2_13_l1_21 l1_21 via_l2_13_l1_16 l1_16 via_l2_13_l1_22 l1_22 via_l2_13_l1_23 l1_23 via_l2_13_l1_19 l1_19 via_l2_13_l1_24 l1_24 via_l2_13_l1_25 l1_25 2

l2_14 via_l2_14_l1_1 l1_1 via_l2_14_l1_9 l1_9 via_l2_14_l1_10 l1_10 via_l2_14_l1_11 l1_11 via_l2_14_l1_12 l1_12 2

l2_15 via_l2_15_l1_14 l1_14 via_l2_15_l1_21 l1_21 via_l2_15_l1_16 l1_16 via_l2_15_l1_22 l1_22 via_l2_15_l1_23 l1_23 via_l2_15_l1_19 l1_19 via_l2_15_l1_24 l1_24 via_l2_15_l1_25 l1_25 2

l2_16 via_l2_16_l1_1 l1_1 via_l2_16_l1_9 l1_9 via_l2_16_l1_10 l1_10 2

l2_17 via_l2_17_l1_11 l1_11 via_l2_17_l1_12 l1_12 via_l2_17_l1_13 l1_13 2

l2_18 via_l2_18_l1_14 l1_14 via_l2_18_l1_21 l1_21 via_l2_18_l1_16 l1_16 via_l2_18_l1_22 l1_22 via_l2_18_l1_23 l1_23 via_l2_18_l1_19 l1_19 via_l2_18_l1_24 l1_24 via_l2_18_l1_25 l1_25 2

l2_19 via_l2_19_l1_9 l1_9 via_l2_19_l1_10 l1_10 via_l2_19_l1_11 l1_11 via_l2_19_l1_12 l1_12 via_l2_19_l1_13 l1_13 via_l2_19_l1_21 l1_21 via_l2_19_l1_16 l1_16 via_l2_19_l1_22 l1_22 via_l2_19_l1_23 l1_23 via_l2_19_l1_19 l1_19 via_l2_19_l1_24 l1_24 2

g2_gnd via_l2_gnd_l1_9 l1_9 via_l2_gnd_l1_10 l1_10 via_l2_gnd_l1_11 l1_11 via_l2_gnd_l1_12 l1_12 via_l2_gnd_l1_21 l1_21 via_l2_gnd_l1_16 l1_16 via_l2_gnd_l1_22 l1_22 via_l2_gnd_l1_23 l1_23 via_l2_gnd_l1_19 l1_19 via_l2_gnd_l1_24 l1_24 2